Dipl.-Ing. Lukas Esterle

Autonomous Distributed Tracking in Networks of Self-organised Smart Cameras

Dissertation
zum Erlangen des akademischen Grades
Doktor der technischen Wissenschaften (Dr. techn.)

ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

FAKULTÄT FÜR TECHNISCHE WISSENSCHAFTEN

1. Begutachter: Univ.-Prof. DI Dr. BERNHARD RINNER
Institut für Vernetzte und Eingebettete Systeme
Alpen-Adria Universität Klagenfurt

2. Begutachter: Prof. Dr. JANUSZ KONRAD
Department of Electrical and Computer Engineering
Boston University

July 2014

Abstract

This thesis presents an approach to address a handover problem in the distributed tracking of objects in networks of autonomous smart cameras. In contrast to multi-camera tracking, distributed tracking assigns tracking responsibility of an object only to a single camera at a time. This requires the network to decide autonomously which camera is responsible for tracking an object at what time.

In typical handover approaches either a centralised control is utilised, *a priori* knowledge for each camera about their environment and neighbourhood relationships is provided, or the employed cameras are calibrated. In contrast, we did not rely on any *a priori* knowledge at all and assume uncertainty regarding location and orientation of the cameras as well as the movement patterns of the objects of interest. Additionally, we did not calibrate our cameras and do not facilitate a central server to control our system. Inspired by market-mechanisms, we implemented a single-sealed bid auction mechanism on each camera and enable it to trade tracking responsibilities for objects of interest among other cameras in the network. Getting to know its trading partners, each camera is further induced with the capability of learning its local neighbourhood, the so-called *vision graph.* This allows each camera in the system to reduce its own communication overhead, and hence the communication effort of the entire system. Making use of biology-inspired foraging mechanisms, we implemented artificial pheromones to build up the vision graph and simultaneously enable the cameras to forget about neighbours where the response rate to advertised auctions drained over time.

To advertise auctions within the network of smart cameras to prospective buyers, we described six different strategies which are able to exploit the vision graph. Each of these strategies gives rise to one out of two objectives: minimising network-wide communication or maximising system-wide tracking performance. The selection of an appropriate configuration, using a variety of strategies in the smart camera network, turns out to require knowledge of the camera setup, the environment as well as the movement patterns of the objects of interest. To neglect *a priori* knowledge of these parameters, we implemented multi-armed bandit problem solvers in every camera of the network. This enables the cameras to learn on their own which strategy fits them best, given a priority on either minimising communication or maximising tracking performance. In conclusion, we were able to show that cameras are able to improve their network-wide performance when learning their own strategy during runtime compared to obstinately assigning homogeneous strategies.

Ehrenwörtliche Erklärung zur Dissertation

Ich erkläre ehrenwörtlich, dass ich die vorliegende wissenschaftliche Arbeit selbstständig angefertigt und die mit ihr unmittelbar verbundenen Tätigkeiten selbst erbracht habe. Ich erkläre weiters, dass ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle ausgedruckten, ungedruckten oder dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte sind gemäß den Regeln für wissenschaftliche Arbeiten zitiert und durch Fußnoten bzw. durch andere genaue Quellenangaben gekennzeichnet.

Die während des Arbeitsvorganges gewährte Unterstützung einschließlich signifikanter Betreuungshinweise ist vollständig angegeben.

Die wissenschaftliche Arbeit ist noch keiner anderen Prüfungsbehörde vorgelegt worden. Diese Arbeit wurde in gedruckter und elektronischer Form abgegeben. Ich bestätige, dass der Inhalt der digitalen Version voll- ständig mit dem der gedruckten Version übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Klagenfurt am Wörthersee, June 26, 2014

(Lukas Esterle) (Ort, Datum)

Acknowledgements

I would like to express my sincere gratitude to many people who travelled along with me on this fantastic journey. First of all, I would like to thank Professor Bernhard Rinner, my supervisor and mentor, for giving me the freedom to work on my own ideas and his leadership, getting me back on track whenever I wandered off.

I would also like to thank Peter R. Lewis for the endless discussions giving me new perspectives on problems, possible solutions, but also fresh approaches when I got stuck. Thanks for the great collaboration in the last years.

The EPiCS project and all project partners providing me with a playground for my own ideas as well as helpful and interesting discussions. Especially, Professor Xin Yao from the University of Birmingham and Arjun Chandra from the University of Oslo for the very fruitful discussions, helpful feedback, and great collaboration.

To my colleagues at the Pervasive Computing Group and office mates over the years, Christian Hofbauer, Christoph Unterrieder, Bernhard Dieber, and Jennifer Simonjan as well as Thomas Winkler and Melanie Schranz. All of you made coming to the office every day a great pleasure. Thanks for great coffees and even greater discussions on current research as well as topics apart from daily business—even though this often lead to more work. To Heidelies Aschbacher for always having an open door and offering a helping hand.

Immense gratitude goes out to my fiancée Aislinn for enjoying all the great moments with me. But even more for being there and having the right words to keep me going when times were rough. I cannot express how much I love you for your patience and understanding.

Finally, to my family: to my brother Sebastian for showing me that life has to be taken easy sometimes; my sister Kristina for always being so proud of me; my Dad for always having an enormous interest in my work, providing me with unorthodox ideas and having a calm approach to everything; and my Mum for always being there for me and giving the correct outlooks on life at just the right time. Without the lifelong support and guidance of my parents this thesis would not have been possible. Thank you so much for everything!

There is more to learn... more to explore!

Publications arising from this Thesis

- **Lukas Esterle**, Peter R. Lewis, Marcin Bogdanski, Bernhard Rinner, and Xin Yao. *A Socio-Economic Approach to Online Vision Graph Generation and Handover in Distributed Smart Camera Networks.* In Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras. August 2011. [31]

- Bernhard Rinner, Bernhard Dieber, **Lukas Esterle**, Peter R. Lewis, and Xin Yao. *Resource-Aware Configuration in Smart Camera Networks.* In Proceedings of the IEEE Workshop on Camera Networks and Wide Area Scene Analysis. February 2012. [76]

- **Lukas Esterle**, Peter R. Lewis, Bernhard Rinner, and Xin Yao. *Improved Adaptivity and Robustness in Decentralised Multi-Camera Networks.* In Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras. October 2012. [32]

- Bernhard Dieber, **Lukas Esterle**, and Bernhard Rinner. *Distributed resource-aware task assignment for complex monitoring scenarios in visual sensorn networks.* In Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras. October 2012. [22]

- Peter R. Lewis, **Lukas Esterle**, Arjun Chandra, Bernhard Rinner, and Xin Yao. *Learning to be Different: Heterogeneity and Efficiency in Distributed Smart Camera Networks.* In Proceedings of the 7th IEEE Conference on Self-Adaptive and Self-Organizing Systems (SASO). September 2013. [53]

- **Lukas Esterle**, Horatio Caine, Peter R. Lewis, Xin Yao, Bernhard Rinner. *CamSim: A Smart Camera Network Simulator.* In Proceedings of the 7th IEEE Conference on Self-Adaptive and Self-Organizing Systems Demo (SASOD). September 2013. [30]

- **Lukas Esterle** *CamSim* (V1.0) [Software]. Release on GitHub. `https://github.com/EPiCS/Camsim`. September 2013. [29]

- **Lukas Esterle**, Peter R. Lewis, Xin Yao, and Bernhard Rinner. *A Socio-Economic Approach to Online Vision Graph Generation and Handover in Distributed Smart Camera Networks.* In Transaction on Sensor Networks. January 2014. [33]

- Kristof Van Moffaert, Tim Brys, Arjun Chandra, **Lukas Esterle**, Peter R. Lewis, Ann Nowé. *A Novel Adaptive Weight Selection Algorithm for Multi-Objective Multi-Agent Reinforcement Learning.* In Proceedings of the Annual International Joint Conference on Neural Networks (IJCNN). July 2014. [86]
- Peter R. Lewis, **Lukas Esterle**, Arjun Chandra, Bernhard Rinner, Jim Torresen, Xin Yao. *Static, Dynamic and Adaptive Heterogeneity in Socio-Economic Distributed Smart Camera Networks.* ACM Transactions on Autonomous and Adaptive Systems (TAAS). Accepted for publication. [52]

Contents

List of Figures

List of Tables

For Hemma

First Chapter

Introduction

With the improvements in technology, through cheaper production of electronic elements and better electric and networking infrastructure, cameras persistently advance into novel areas of use. Video, surveillance or security cameras are nowadays used for many reasons in diverse areas: monitoring motorways and highways to identify accidents, traffic jams or unlawful behaviour; surveillance of shopping areas to identify shoplifters; monitoring production lines for quality assurance, or; observation of public areas for crime prevention—to name just a few.

According to Gerrard and Thompson [43] there are about 1.8 million cameras in public areas used for surveillance in the UK. This number is based on an extrapolation of the available, public surveillance cameras in Cheshire. A clear trend towards increasing numbers of surveillance cameras is also noticeable in Germany. The number of CCTV cameras in public areas has increased from roughly 11,000 to more than 17,000 cameras in only five years [85] in Bavaria alone. The vast majority of these cameras are 'dumb' cameras which can only transmit raw image data to a central control or operations room. This data has then to be stored at a central server and either be processed by a computer or a human operator. Human operators are usually sitting in front of multiple monitors analysing images, identifying suspicious behaviour, and tracking people manually. The drawbacks of transmitting and storing imagery to a central point are on the one hand, a huge overhead in communication and needed storage capacity, and on the other hand, a single point of failure in the system as well as a single point of attack for individuals trying to get their hands on the video data. However, it has to be mentioned that the vast amount of digital imagery has also fostered a lot of research in computer vision. Detection of specific faces and objects [47, 65, 95], identification and analysis of situations, atypical events and poses [11, 64], and tracking of a person of interest [94] to automate the process often undertaken by human operators, are just three exemplary areas of research.

This chapter is structured as follows: Section 1.1 introduces the problem investigated in this thesis. Afterwards, the research questions are presented in Section 1.2. Section 1.3 summarises the contribution to the state of the art made in this thesis and finally, Section 1.4 contains the thesis outline.

1.1 Motivation

In recent years 'dumb' cameras have evolved into embedded smart cameras [77, 93], combining a processing unit with an image sensor on a single platform. These processing capabilities, even though limited, allow the smart cameras to pre-process video data on-site and transmit only aggregated information, or a complete analysis of a scene, instead of plain images. Modern smart cameras are even capable of accomplishing processing intensive tasks, such as object tracking. In object tracking, a description of the object of interest is initially provided to the camera. The camera thereafter attempts to re-identify this object in consecutive frames of its own field of view (FOV). There are various tracking algorithms to locate objects in each frame matching the given description with the highest probability. While we employ tracking algorithms to identify and locate moving objects, we do not elaborate on these fundamental tracking techniques in this thesis.

Soon enough, single smart cameras have been connected to distributed smart camera systems [75, 78]. Tracking objects in multi-camera systems can be approached in two different ways. The first approach uses all cameras to track various objects and the gathered information is fused at a central control. When tracking objects within multiple cameras concurrently, cameras need to align their FOVs to ensure the gathered data is coherent. To do so, a calibration process is employed to remove geometric distortions caused by the camera lens. Aligning FOVs using a calibration process needs extra work before the system can go online. This extra effort might be feasible with small numbers of cameras but could be highly problematic in larger systems with tens, hundreds or even thousands of cameras. Furthermore, in case one of the cameras' parameters is changed, new cameras are added or cameras are removed from the network, cameras might need to be re-calibrated to ensure proper functionality. We refer to the second approach as *distributed tracking*, where each object of interest is tracked by a dedicated camera. This requires the network to decide which camera is responsible to keep track of a specific object at any time. Furthermore, the tracking camera has to decide when and to which camera it should transfer the tracking responsibility to. This process of transferring a tracking responsibility is know as *handover*.

Knowledge about the neighbourhood of each camera, utilised in the handover process, allows us to further separate distributed tracking approaches into two groups: with and without *a priori* knowledge. Introducing *a priori* knowledge about the network topology, the environment or simply about neighbourhood relations between the cameras, can on the one hand improve tracking performance of the network tremendously. On the other hand however, this requires prior work of either a central component, such as a server, or a human operator. In contrast, we do not assume any *a priori* knowledge but induce our cameras with software agents, allowing them to act autonomously, learn about their neighbourhood and the environment they are embedded in, and enable the network of cameras to organise themselves to improve the network-wide performance. While distributed tracking requires extra effort in terms of coordination or *a priori* knowledge, the main advantage is the increased number of concurrently tracked objects in the network.

Even though every single smart camera has limited processing capabilities, in such a co-operative ensemble, the entire network can distribute the total workload of the system to all available cameras.

There is a plethora of behavioural strategies one can employ to distribute and assign tracking responsibilities. As the diverse approaches give rise to different, sometimes even opposing, benefits and/or drawbacks regarding required qualifications, assigning the same strategy to all cameras might not always be beneficial. This is due to the individual situation of each camera in the network. An operator, trying to adjust the system to fulfil all requirements as needed, has essentially three options. First, the operator could assign strategies homogeneously in the network, where all cameras have the same behaviour. This might result in cameras using a strategy not optimal in their given situation and hence achieving less performance than possible. Second, the operator could try to find a heterogeneous assignment, where at least two cameras behave differently. On the one hand, this can result in a more beneficial performance of the entire network, since cameras operate optimally according to their location, orientation and viewing angle. On the other hand, the operator is required to know about the individual state of each camera. The third option allows each camera to learn about its own situation. Based on this learned information, the camera could select the strategy optimal for its individual situation during runtime. Moreover, this allows each camera to deal with dynamics and changes in the network as well as object movements.

This thesis concentrates on cooperative ensembles of smart cameras, able to track objects autonomously in absence of central coordination. Moreover, the system is able to operate without any *a priori* knowledge, where cameras do not know anything about the environment in which they are deployed. Apart from enabling the cameras to organise themselves in terms of learning about their neighbourhood to optimise local as well as network wide performance, we additionally enable cameras to reason about their own behaviour. This allows them to select the best strategy from a repertoire of possible strategies for their current situation.

1.2 Research Questions

In this thesis the following research questions are addressed:

- How can a distributed smart camera network track an object consistently without a central control? More precisely, how can the network assign tracking responsibility to a single camera, instead of tracking the object with all cameras at the same time?

In addition, how can the network ensure to select the camera with the best view in terms of tracking confidence?

- If the tracking responsibility is assigned by an autonomous software agent on the camera using information from other cameras, how can this agent select the best camera without having to gather information from every single camera in the network?

- Even when considering static networks of smart cameras, the state of every camera in the network can change due to various uncertainties, such as changing their orientation or their location, or simple failure. The question arises: how can the network cope with such uncertainties and maintain continuous tracking of objects at the same time? Moreover, if neighbourhood relations have been learned, how can individual cameras 'forget' this information?

- When multiple approaches to assign tracking responsibility are at hand, where each approach focuses on one out of multiple objectives, how can a single camera decide which approach fits best for its current situation? Furthermore, how can the autonomous software agent in the camera make sure that the best approach is employed in a changing environment?

1.3 Contribution

The objective of this thesis is to enable distributed tracking in decentralised smart camera networks. In order to coordinate tracking responsibilities autonomously within a network, the cameras are induced with abilities to organise themselves as well as adapt to changed conditions in their environment. In addition, this allows them to optimise their local performance as well as the performance of the entire network. Moreover, cameras organise themselves to employ the strategy that fits best for their individual situation. The work presented in this thesis has been conducted in the EPiCS project (Engineering Proprioception in Computing Systems) and received funding from the European Union Seventh Framework Programme under grant agreement n^{o} 257906. Modelling of concepts has been done in close collaboration with our project partners at the University of Birmingham and University of Oslo. The real camera setup has been done in cooperation with the Austrian Institute of Technology (AIT). One of the trackers used in the real world experiments has also been supplied by AIT. The key aspects contributing most significantly to the current state-of-the-art are as follows:

Distributed Tracking and Autonomous Camera Control. In a network of smart cameras, either all cameras track all visible objects concurrently or the responsibility of tracking a specific object has to be handed over between cameras. In the latter case, it has to be decided which camera is best suited to track an object at a given time. In previous approaches, either central coordination had been used to assign responsibilities to specific cameras or cameras had predefined knowledge about the topology of the network, and hence knew which camera would be most suitable to continue tracking. In Chapter 3, we formulate a market-based approach to assign tracking responsibilities within a network of smart cameras in the absence of central coordination. A completely decentralised approach is used, employing auctions with single sealed bids, to transfer the tracking responsibility to the camera with the best view of the object at a given time step. This method handles objects that are to be tracked as goods, which can be traded by cameras. To define a 'price' for their goods, we developed a function that enables cameras to calculate the value of objects autonomously. The results of this research have been discussed in [31] and elaborated on in [33].

Self-Organisation of Networks of Autonomous Smart Cameras. Various distributed applications benefit from organising smart cameras into smaller groups. In this thesis, smart cameras learn about their immediate neighbourhood across the entire duration of their service. To achieve this, we introduce artificial pheromones to create links between cameras based on the trading behaviour of tracked objects. This allows the cameras to organise themselves within the network and build up the so-called vision graph in absence of a central coordination. Moreover, this knowledge allows each currently tracking camera to reduce its communication effort with other cameras in the network, while keeping the tracking performance of the entire ensemble at a high level. We describe two auction strategies in Section 3.2 which allow each camera to advertise objects of interest to other cameras in the network. Furthermore, we present three communication policies exploiting the private vision graph of each camera, in order to reduce the number of exchanged messages within the network. Using artificial ant-pheromones to indicate the spatial relationships between cameras allows the network to organise itself. Fading pheromone links increase the robustness to changes in this approach and allow individual cameras to 'forget' about previously learnt, but now changed neighbourhoods. In Section 3.4 we define various uncertainties regarding the cameras in a network, such as adding, removing and changing the location and orientation of a camera during runtime. Additionally, the section discusses the robustness of our socio-economic approach to these uncertainties and compares our method to a static approach where the neighbourhood is known *a priori*. The presented approaches and the resulting outcomes have been discussed in [31–33].

Self-Organisation in Smart Cameras. In the course of this work, various approaches for distributed camera control are presented. While a network can have a good overall performance when all cameras use the same approach, assigning heterogeneous approaches can increase the performance even further. In Section 3.5 we exhaustively analyse and compare homogeneously, as well as heterogeneously assigned strategies and policies, showing the benefits of heterogeneous assignments based on the unique neighbourhood relationships of each camera. The efficient heterogeneous assignment is highly dependent on the given scenario and setup of the cameras. Creating such a heterogeneous assignment requires knowledge of the environment as well as the movement patterns of the objects observed in the given area. To overcome the problem of *a priori* knowledge regarding orientation, location and neighbourhood relations of smart cameras, Section 3.5 presents an approach using a multi-arm bandit problem solver. This allows the cameras to self-organise their available algorithms during runtime without any supervision or central control, enabling each camera to select the strategy that fits its current situation best. The presented work on homogeneously and heterogeneously assigned approaches to all cameras as well as the work enabling cameras to learn which strategy fits them best has been published in [53].

Simulations and Real World Deployments. A simulation tool called *CamSim* has been developed for the purpose of testing our novel market-based approach. *CamSim* simulates an arbitrary number of autonomous smart cameras operating in a network with the goal of tracking objects. The number of simulated objects and cameras is only limited by the capabilities of the computer running the simulation tool. The simulation environment has been used to evaluate the presented approaches, policies and strategies discussed throughout this thesis. *CamSim* has been released open source in order to be available to a broader community and allow the prospective developers to extend and refine the existing simulator. Furthermore, we deployed our methods in two different smart camera networks in the real world. The first network consists of homogeneous smart cameras with and without overlapping fields of view (FOVs). The second network setup uses heterogeneous smart cameras in laboratory conditions consisting of four custom built SLR smart cameras[1] and two smart cameras built with

[1] `http://www.slr-engineering.at/`

off-the-shelf components. The simulation environment has been used in all publications related to this thesis in general and elaborated on in [30]. The real world deployment has been employed and discussed in [33, 53].

The presented concepts were developed and modelled in cooperation with partners from the EPiCS project. The authors genuine contributions are as follows:

- Definition of a function to allow cameras to calculate the value of objects autonomously for both, a simulation environment and a real world application.
- Description of two auction strategies which allow the cameras to exploit their unique neighbourhood relations with other cameras in the network.
- Presentation of three communication policies exploiting the private vision graph of each camera to reduce the number of exchanged messages within the network.
- A comparison of the communication policies combined with auction schedules, so-called strategies, showing the significant reduction in communication while keeping the utility high for the entire network.
- Implementation of the camera network simulation tool *CamSim* and the corresponding socio-economic concepts. This custom simulation environment allows for fast testing and evaluation of the presented approaches and policies.
- A definition of various uncertainties regarding the cameras in a network such as adding, removing and changing the location and orientation of a camera during runtime.
- An analysis of the robustness of the presented approaches regarding various uncertainties of cameras such as failure, resets, relocation or orientation changes.
- A comparison of homogeneously as well as heterogeneously assigned strategies and policies, showing the benefits of heterogeneous assignment of strategies based on the unique neighbourhood relationships of each camera.
- An outline of how the presented approach allows a network of smart cameras to initially assign tracking responsibilities for objects to the camera having the best view. This is done with a minimal number of messages and in a completely distributed fashion.

- Deployment of a heterogeneous smart-camera network in lab conditions consisting of four custom built SLR smart cameras and two smart cameras built with off-the-shelf components.

- Implementation of the application, including some of the employed tracking algorithms and the socio-economic based approach for handover, used in the real world deployment.

- Design and performance evaluation of all performed experiments in simulation as well as real world environment deployments except the Wilcoxon rank sum test which was done by Arjun Chandra at the University of Oslo.

1.4 Thesis Outline

The remaining chapters of this thesis are organised as follows:

Chapter 2 discusses the current state-of-the-art related to this thesis, namely multi-camera tracking and handover, market-mechanisms for resource allocation as well as heterogeneous assignment of algorithms in distributed sensor networks. Chapter 3 defines distributed tracking and the associated handover problem. Moreover, this chapter introduces the socio-economic approach for smart cameras to enable distributed tracking. This method allows cameras to act as self-interested agents which are able to host Vickrey auctions. Furthermore, the building of vision graphs based on a camera's trade behaviour as well as the exploitation of the learnt neighbourhood is explained. While Vickrey auctions work very well for the exchange of tracking responsibilities in a network, the presented market-based approach can also be used to assign initial tracking responsibilities at the time the system is started up. This idea is outlined in Section 3.3. As camera networks are prone to uncertainties such as changes and errors, the network has to be able to deal with such uncertainties during runtime. Furthermore, Chapter 3 presents different uncertainties and their impact on the overall performance of the network. Moreover, the robustness of the presented socio-economic approach is shown as well as its ability to deal with these changes and errors during runtime. Even though assigning the same algorithm to advertise objects to all cameras in the network allows the system as a whole to perform very well, assigning algorithms heterogeneously to individual cameras results in even better performance. This heterogeneous assignment, as well as an approach to enable cameras to learn the best strategy for their given situation, concludes Chapter 3.

In order to evaluate the presented approaches, a 2D simulation environment, called *CamSim*, has been used. *CamSim* is outlined in Chapter 4. This chapter gives an overview of the entire simulator, its usage, and the most important components. Additionally, all scenarios used for evaluation and their corresponding properties are discussed in detail.

Chapter 5 focusses on the evaluation of our approaches under various aspects in simulation. After establishing the feasibility of our approach, we show its robustness in the presence of different uncertainties. Furthermore, we present results of networks where strategies were assigned heterogeneously and homogeneously, as well as autonomously learnt during runtime by each individual camera. The chapter concludes with a summary and discussion of our evaluated approaches and their respective results.

Chapter 6 describes the smart camera networks deployed in our laboratories at Alpen-Adria Universität Klagenfurt. The employed heterogeneous hardware as well as the general setup is discussed. Additionally, experiments conducted for evaluating our approaches in a real setting are discussed. The resulting outcomes of these experiments are presented at the end of Chapter 6.

A summary of the presented work and the contributions made is given in Chapter 7. Moreover, the possible strains for future work are outlined.

Second Chapter

Related Work

This chapter discusses the current state of the art in the three main research areas relevant to this thesis. In Section 2.1 several approaches for multi-camera tracking and handover are examined and qualitatively analysed. Afterwards, Section 2.2 studies mechanisms to assign and allocate resources which have taken inspiration in real markets. In the course of this thesis, various techniques to hand over objects of interest from one camera to another are presented. Section 2.3 explores the benefits of heterogeneous assignment of different algorithms to nodes in networks leading to similar or even better results when compared to homogeneous assignment. Furthermore, various applications exploiting these benefits are discussed.

2.1 Multi-camera Tracking & Handover

In multi-camera tracking, a single fundamental decision has to be made: should all cameras try to track the object or person of interest, or should only a single camera with the best view be responsible for tracking? Using all cameras to track a single object or person of interest, allows on one hand to neglect otherwise mandatory coordination of the tracking responsibility. On the other hand requires fusion of the tracking results of all cameras at either a central control or at all tracking cameras in the network. Additionally, concurrent tracking requires vast amounts of resources, such as memory and processing power at the individual cameras. Furthermore, while tracking a single object might be feasible, following multiple objects at the same time within a single camera might overexert the available resources. Therefore, this thesis focuses on the assignment of tracking responsibility of a given object or person of interest to a single camera. To accomplish this, the fundamental tasks of single camera object detection and tracking must be expanded by a coordination mechanism, which is referred to as *handover*. The task of this handover is to find the successive camera with the best view on the target object, once it has left the FOV of the current camera [28]. Various mechanisms have been proposed to track objects in a multi-camera network. These mechanisms vary in terms of the required assumptions of the camera network, the distribution of data and processing, as well as the required resources [55]. We analyse the related work with a focus on the following characteristics:

- **Distributed:** Is the proposed system capable of performing the handover in a distributed fashion or is a central component, such as a server or a cluster of standard computers, required?
- **No *a priori* knowledge:** Is any *a priori* knowledge, which is predefined for all cameras manually or in *a priori* learning phase, required in order to allow for dis-

tributed tracking in the network (e.g., position of each camera or neighbourhood relations)?

- **Non-overlapping FOV:** Does the proposed technique operate with cameras having non-overlapping FOVs as well as with cameras having overlapping FOVs?
- **Uncalibrated:** Does the approach require calibrated cameras in the network? Even though calibration could be thought of as *a priori* knowledge, it is considered as a separate characteristic due to its significance in various techniques.
- **Static cameras:** Static cameras are not able to change their pose actively.
- **Asynchronous cameras:** Synchronized cameras create images from their FOV exactly at the same time. This allows to infer a precise relation between cameras based on the occurrence of objects.
- **Best view:** The network of cameras does not only track the object or person but tries to have the best view on the object or person. This can either be considered right after the handover or at any given time.

Considering these categories, networks of active cameras, which are able to change their FOV either autonomously or based on user input, are a special case. These cameras can transit from a network with overlapping FOVs to a network with non-overlapping FOVs and *vice versa*. Various approaches have concentrated on distributed tracking in networks of active cameras.

Erdem and Sclaroff [28] focus on the movement of objects being tracked in the network to learn which camera is used next for racking along the object's path. Song et al. [82] use a game theoretic approach, where the cameras negotiate with each other to reach a Nash equilibrium. This requires multiple negotiation cycles to reach the best possible global utility. The approach allows to track multiple objects but does not enable cameras to learn neighbourhood relations during runtime. Möller et al. [66] find the re-occurrence of a tracked object or person within all cameras in the network based on their colour histogram using a central server. This approach displays only little scalability as well as a single point of failure. The mechanism presented by Qureshi and Terzopoulos [74] allows to persistently surveil a simulated environment by assigning the tracking tasks to groups of cameras. A group is formed based on dedicated group leaders and via inter-camera negotiation, using an auction mechanism.

One of the first approaches on identifying neighbourhood relationships of static cameras to accomplish distributed tracking is presented by Javed et al. [48]. They use the co-occurrence of a person in two overlapping FOVs of cameras to calculate the rough epipolar geometry of the uncalibrated cameras. Black et al. [9] use a set of calibrated cameras and identify the common ground plain in overlapping FOVs. They employ Kalman filters in combination with homography mappings to concurrently track objects in multiple cameras. Ellis et al. [27] are the first ones to propose a completely distributed handover approach. They observe moving objects in a network of synchronised and calibrated cameras and use the gathered information to learn the topology over time. A similar approach is taken by Mandel et al. [61] and Kim et al. [50]. Even though neither of their approaches requires calibrated cameras, Mandel et al. assume only overlapping FOVs, while Kim et al. induce *a priori* knowledge about the monitored environment to the camera network. Makris et al. [60] also exploit temporal correlations in observations of object movements. They learn the entry and exit areas of every camera's FOV in the network by using Expectation-Maximisation from an extended dataset. Additionally, they use a Gaussian Mixture Model to represent the collection of entry/exit areas. Similarly, Loy et al. [59] propose a Cross Canonical Correlation Analysis to measure the interrelationships of activities in different FOVs. This allows them to infer the topology of the camera network. Cheng et al. [13] do not track people or objects but use the common background information of overlapping FOVs to identify neighbouring cameras. An approach presented by Detmold et al. [20] is based on simple exclusion. This means that if an object is within the FOV of a camera and not in the FOV of another camera, these two cameras cannot be monitoring the same space. To overcome the problem of non-overlapping FOVs they introduce temporal padding, which also overcomes clock skews between cameras. An implementation of a distributed handover on embedded smart cameras is presented by Quaritsch et al. [73]. Their technique does not require a central coordination, but uses predefined areas for the handover to pre-specified neighbouring cameras. Farrell and Davis [36] introduce a distributed approach based on sequential Bayesian estimation using a modified multinomial distribution in absence of *a priori* knowledge. While they are able to track objects within the network of cameras, they cannot ensure the best view of the object or person of interest. Cichowski et al. [14] track people in networks of calibrated and synchronized cameras. They segment each FOV in a pre-defined grid, where each cell is considered for overlap between two or more cameras. A fuzzy automaton based approach to select the camera for further tracking is presented by Morioka et al. [67]. They are able to determine the best camera out of all neighbouring cameras based on previously selected cameras. Li and Bhanu [56] present a game-theoretic framework which is able to select the best view based on user-provided criteria. Their server-based implementation synchronises the frames of a network of uncalibrated cameras with and without overlapping FOVs.

Table 2.1 summarises the related work presented in this section and indicates the fulfilment of the described properties of the various approaches.

	Distributed	**No *a priori* knowledge**	**Non-overlapping FOV**	**Uncalibrated**	**Static cameras**	**Asynchronous cameras**	**Best view**
Erdem & Sclaroff [28]	○	●	◐	●	○	●	◐
Song et al. [82]	●	◐	◐	○	○	●	●
Möller et al. [66]	○	○	◐	●	○	●	●
Qureshi & Terzopoulos [74]	●	●	◐	●	○	○	●
Javed et al. [48]	○	●	○	●	●	●	○
Black et al. [9]	◐	◐	●	○	●	○	○
Ellis et al. [27]	●	●	●	○	●	○	○
Makris et al. [60]	○	◐	●	●	●	●	○
Cheng et al. [13]	●	●	○	○	●	●	○
Mandel et al. [61]	●	●	○	●	●	○	○
Detmold et al. [20]	○	●	●	○	●	●	○
Quaritsch et al. [73]	●	○	●	●	●	●	○
Farrell & Davis [36]	●	●	●	●	●	○	○
Kim et al. [50]	◐	○	●	●	●	●	○
Loy et al. [59]	○	○	●	●	●	○	○
Cichowski et al. [14]	○	●	○	○	○	○	○
Morioka et al. [67]	●	◐	○	○	●	●	●
Li & Bhanu [56]	○	●	●	●	●	◐	●
Socio-economic approach	●	●	●	●	●	●	●

Table 2.1: Various approaches for multi-camera tracking, clustered by active and static cameras and ordered by their date of publication. ● represents a fulfilment of the described property, ◐ illustrates a partial compliance of the corresponding feature, and ○ describes the corresponding property as not fulfilled.

2.2 Market-mechanisms for Resource Allocation

Deciding which camera should be responsible to continue tracking can be considered a resource allocation problem. In object tracking, the known objects have to be allocated amongst the cameras, depending on their local knowledge, available resources and immediate objectives. Economics has long been an inspiration in allocating resources in computing systems. This existing family of techniques is known as market-based control, which applies economic principles to solve the resource allocation problem in distributed systems [15].

A typical system controlled by market-based mechanisms consists of autonomous software agents, which are able to make their own actions and decisions. Through the defined market mechanisms, these agents are able to interact with each other. On the one hand, buyer agents attempt to purchase resources from sellers with the objective to maximise their utility function. This function is based on the tasks or requirements at hand, or the individual objectives of the agents. On the other hand, seller agents offer resources and in response demand artificial or even real currency as payment. The charged price is usually dependent on either the agents strategy or the quality and/or quantity of the resource on offer. As in a real market, it is assumed that in a system controlled by market mechanisms, supply and demand will determine the price of offered goods. While scarce goods will be valued at higher prices, a low number of buyers might force the seller to lower the price on offered goods. The fundamental idea is that the competition for the same resources between the buyer agents leads to an efficient allocation of these resources. In addition, the allocation reflects the objectives and preferences of all agents in the system based on their very local utility function.

The market-based mechanisms and techniques used in these application domains can be categorised into the following three groups: (i) posted offer markets, (ii) bilateral bargaining and (iii) auctions.

The first group of approaches has been inspired by modern retail markets, where goods are offered at a certain price, and is hence referred to as *posted offer* or *posted price* [49, 71]. These posted offers are set by the seller, are visible to all market players, and have non-negotiable prices for certain goods. The potential buyers can then purchase desired quantities from the seller. The main benefit of this approach is the sellers ability to operate fully distributed without any central control to determine the price for the offered goods. Drawbacks of this approach are either complex strategies for sellers to adjust the prices according to demand or a negative impact for the seller when prices for goods do not change over time [8]. With scarce network bandwidth, Gupta et al. [45] propose to price network usage in contrast to limit the bandwidth usage for everyone. This pricing approach decentralises the resource allocation for the limited bandwidth. However, this means that bandwidth usage would need to be monitored and billed. They argue that efficient resource allocation is key in order to improve overall network performance.

Bilateral bargaining, the second family of approaches on market-based control, does not provide a certain price for a given good. Instead, both agents, the buyer and the seller, try to achieve an acceptable price for a certain good by bargaining. Moreover, this means the price for a good usually changes during the bargaining process. While bilateral bargaining is working well for distributed resource allocation and can result in equilibrium between sellers and buyer [17], it still requires bargaining strategies for both groups of agents. A selling agent may not set their asking price too high in order to have prospective buyers interested in their goods. But then again the price has to be high enough to allow the seller to make a profit. Similarly, the buyer agents should not cap their prices, they are willing to pay, in order to be able to purchase goods. At the same time, a buyer agent should not spend more money than they have to [17]. This approach has been realised in various systems such as AVALANCHE [35], and CATNET [4]. In multi-robot coordination, Dias et al. [21] analyse market-based approaches to allow robots to allocate resources in order to handle various tasks as a team. They point out the importance of the trade-off between solution quality and scalability of these coordination mechanisms. The goal of replicating the dynamics of human markets with complex cognitive agents is the subject of continuing research (e.g., [12, 42, 57]).

The third group of market-based control mechanisms is the large group of auctions [26, 51, 62]. In auctions, sales of commodities are managed by an auctioneer. The buyers submit bids to the auctioneers to express their interest in a good. These bids contain a valuation for the good in form of some kind of resource. This resource also acts as a comparative value for the auctioneer to select the winner of an auction. Submitting bids varies from auction to auction. In *sealed bid* auctions, participants of the auction do not know the value of bids submitted by other bidders. The buyers have to value a good on their own and can only speculate about other buyers valuation of the same commodity. Whereas in *open bid* auctions, every participants knows the submitted bids of all other bidders. Hence, bidders can reason about the valuation of other bidders based on their submitted bids for a certain good. Additionally, auctions either allow for submission of only a *single* or *multiple bids* by an individual bidder. In contrast to *posted offer* markets, where the price is selected by the seller, and *bilateral bargaining*, where the price is decided upon by the seller and the buyer, in *auctions* the price is eventually set by the buyers. Commonly known examples of auctions are

- **English auction:** The English auction is the most prominent auction mechanism. Prospective buyers name their prices publicly and can therefore reason about the behaviour of other participants. Accepted bids have to overbid the previous bids value. Therefore, this auction is also called *increasing multiple open bid auction.* The good is sold to the buyer making the last and highest bid.

- **Dutch auction:** While in English auctions the price starts low and is raised by the buyers, the Dutch auction starts with a high price, which is reduced continuously. The good is sold to the first bidder agreeing to the asked price. Even though the valuation of each participant is not directly revealed to other bidders, prospective buyers can speculate about the behaviour of others and change their own valuation during the auction process. This type of auction is also known as *decreasing single sealed bid auction.*

- **First-price sealed-bid auction:** In this auction, prospective buyers submit bids without revealing the offered amount for a certain good to other bidders. The auctioneer sells the good for the highest bid. In systems, where the true valuation is required, this strategy should be neglected since a buyer with enough resources could bid untruthful in order to outbid other participants.

- **Vickrey auction:** Vickrey auctions are also called *second-price sealed-bid auctions.* Similar to *first-price sealed-bid auctions*, the bids are not revealed to other prospective buyers. The good is sold to the highest bidder but for the second highest price. This approach enforces truthful bidding among the participating bidders [88]. Furthermore, this auction method has been generalised to sell internet advertising keywords [25].

- **Unique-bid auction:** Unique-bid auctions are *single sealed-bid auctions* as well. After all participants have submitted their bids, the good is sold to the highest (or lowest) bidder with a unique bid. A bid is considered unique, when no other bidder submitted an equally valued bid. When using unique-bid auctions, it is possible that the highest unique bid is the lowest submitted bid amongst all.

- **Double auction:** While in the previous auction mechanisms, the price asked by the seller did not directly influence the final price, double auctions do incorporate this factor. Both, the seller and the buyer, submit bids. The bid of the seller contains the minimum price asked and the bid of the buyer the offered price to be paid. The price for the good is the mean between asked and offered price. The good is only sold though, if the offered price exceeds the asked price.

While the presented auction mechanisms show the presence of a central control in form of an auctioneer, Cliff and Bruten [18] point out that this centralisation dismisses the core advantages of a market-based system such as its robustness, decentralisation and scalability. Introducing distributed auction mechanisms tries to mitigate this problem [34, 44, 46] by replacing a central auctioneer by a number of local ones, or by selecting an auctioneer using leader election algorithms. Waldspurger et al. [89] present one of the first examples of computational economy to allocate computational resources in a distributed system. In their system, called *Spawn*, agents offering resources also host second-price, sealed-bid auctions for other agents to acquire these offered resources. Agents face the need to decide which auctions they want to participate in and which ones they want to omit. If they only bid for the resources they need, the agent might be overbid and end up with too little resources. In case the agent bids for more resources than it needs, it has to decide what to do if it wins too many auctions. This dilemma has been the focus of various work such as that by Gerding et al. [40, 41]. However, this is only problematic if the buyer depends on a specific amount of a limited resource. In case of the presented problem in this thesis, buyers can continue their work regardless of the number of won auctions.

In Clearwater et al. [15] various application domains are proposed to be managed by market-based control, such as regulation of office temperatures [16], resource allocation in distributed systems [38], or adaptation of network bandwidth for individual agents [63]. Wellman et al. [90] use market techniques to determine schedules using different auction mechanisms. They investigate the existence of equilibria and are able to show that combinatorial auctions, where a buyer tries to acquire multiple goods with a single bid, support equilibria whereas single-good auctions do not. A detailed review of these three groups, including their benefits and drawbacks, is provided by Cliff [17] and Lewis [54].

2.3 Heterogeneous Assignment of Behavioural Strategies

Assigning the same behaviour to all components in a distributed system homogeneously, to accomplish a common task, might result in a satisfactory outcome. Having different situations for each component or agent in the system, or in case situations change over time, a single common behaviour might not be sufficient anymore. Inducing agents with a variation of different behaviours can result in increased stability and allows for more effective self-organisation, and therefore leads to even better local and system-wide outcomes [10]. Dias et al. [21] point out that heterogeneous capabilities are even more important for multi-robot systems, since complex missions often have different requirements. Additionally, it is more practical to design and build an agent for a very specific task. In sensor networks, heterogeneity can take the form of different parameters, diversity in behaviour of each node, or even a variation of hardware and therefore capabilities between nodes. To find an advantageous, heterogeneous selection of behaviour, hardware, and parameters for each node to retrieve a high performance at the global level, well designed self-organising algorithms can be employed. Prasath et al. [72] highlight two key issues when trying to find beneficial heterogeneous assignment:

1. Whether heterogeneity allows optimisation beyond the level that is possible in the homogeneous case for a given scenario and corresponding objectives, and
2. what algorithms to use in order to achieve near-optimal heterogeneous networks.

Stone and Veloso [83] point out that heterogeneous multi-agent systems are able to solve various complex group level tasks, where teams of homogeneous agents may fail. These benefits are investigated by Campbell et al. [10] using a heterogeneous multi-agent system for a task allocation problem. They show that a variation of agents creates more possible *organisations* (configurations) of the system. Some of these configurations might enable a collective system to achieve its goal faster and/or with a better outcome. Römer et al. [80] propose an adaptation of the node's roles in a sensor network, based on its location and purpose. This adaptation is done using a predefined set of rules, which are the same for all nodes in the network. Furthermore, Frank and Römer [37] describe their idea of a system with generic role assignment as well as an improved version of their rule based role-assignment algorithm. Their approach relies on repetitively broadcasted messages of all nodes, containing their property information for rule evaluation. Mottola and Picco [69] introduce the idea of logical neighbourhoods for wireless sensor networks. They describe how sensor nodes may not be spatial neighbours, but codependent regarding the information they need and provide. Nakamura et al. [70] reactively assign roles for data routing to different sensor nodes, based on events to save energy during idle periods. Abbas and Egerstedt [2] also indicate that certain nodes in heterogeneous multi-agent system are more crucial and significant than others. Therefore, they investigate the dependency between heterogeneous nodes trying to accomplish specific tasks. To improve the performance of the network and optimise placement of the nodes, they use *graph colouring* to position them within a sensor network topology. Rojković et al. [79] present an agent-based approach to assign roles in different nodes of sensor-networks. Additionally, they use a genetic algorithm to find a near-optimal solution. In smart camera networks, Dieber et al. [23] adapt the number of cameras in the network, changing their settings and the tasks assigned to the cameras. They use a combination of an expectation-maximisation algorithm and evolutionary algorithm to satisfy predefined constraints.

The importance of dynamic configuration of heterogeneous sensor networks in environments where there is no *a priori* information has been pointed out by Salazar et al. [81]. They show how to re-configure a sensor network to environmental changes based on various local events by using a *collective diffusion search* algorithm. Similarly, Anders et al. [3] study the effects of heterogeneous agents in self-organising systems in uncertain environments. They apply two algorithms one based on schooling fish, and the other one based on honey bees to accomplish dynamic task assignments. While the honey bee algorithm can lead to oscillations or insufficient change of the agents contribution, the schooling fish algorithm is prone to these deviations but needs a high inter-agent variation to achieve better results. Their simulation results even indicate that there might be a critical amount of inter-agent heterogeneity which helps the algorithm to find near-optimal solutions.

Third Chapter

Autonomous Distributed Tracking in Networks of Self-organised Smart Cameras

Object tracking in multi-camera networks can be accomplished in two different ways: (i) tracking the object in all possible cameras concurrently, also referred to as *multi-camera tracking*, or (ii) the object is only tracked by a single camera with the best view at the time, which we refer to as *distributed tracking*. Both methods have their benefits and drawbacks. In multi-camera tracking, each camera tracks the object as soon as it is identified within its FOV. From a network point of view, this results in a high, accumulated resource consumption across all cameras due to constant searching for objects and concurrent tracking in multiple cameras. On the upside this approach can result in high tracking accuracy. In distributed tracking, the tracking responsibility is limited to a single camera. On one hand, resources are only consumed by a single camera in the network. On the other hand, a coordination of tracking responsibilities across all cameras in the network is required—a trade off between communication and tracking performance arises. We induce our cameras with self-interested autonomous software agents, trading tracking responsibilities in order to maximise their own utility. By trading tracking responsibilities, cameras learn their local neighbourhood during runtime. This allows them to target marketing and hence further reduce network-wide resource consumption. In pursuing local objectives, nodes are typically endowed with a common algorithm or behavioural strategy. However, nodes are often located in different areas, having different perceptions of the world, and are subject to different experiences. In these cases, nodes may be better off adopting different strategies from each other, in order to better achieve their own local objectives. It has also been shown that such heterogeneity among nodes can lead to better achievement of system wide objectives [3, 10], especially when nodes can adapt independently in response to uncertainties and changes in the environment during the network's lifetime [81].

The chapter proceeds as follows. In Section 3.1, the problem of distributed handover and autonomous camera control is formally defined. Section 3.2 describes our novel, distributed market-based mechanism to assign tracking responsibilities within a network of autonomous smart cameras in the absence of a central coordination. Moreover, we introduce the usage of artificial pheromones to learn the neighbourhood relationships between cameras. Additionally, we explain how these relations can be further exploited to reduce network-wide resource consumption by targeting only a specific subset of cameras when advertising tracking responsibilities. Subsequently, in Section 3.3, we outline how the presented market-based approach can also be used to initially assign tracking responsibilities to cameras at the start of the system.

Uncertainties in the smart camera network can have significant impact on the system wide tracking performance, especially when the neighbourhood relations are statically defined. This problem and the different associated uncertainties we consider are discussed in Section 3.4. In this section we also extend the initial handover algorithm to deal with the arising problems of uncertainties explicitly. Finally, we study the effects of heterogeneity among nodes in our distributed smart camera networks. After motivating the problem of homogeneous assignments of algorithms in Section 3.5, We illustrate how heterogeneous assignment of algorithms to different nodes to accomplish a common goal can outperform a single algorithm executed by all nodes in the network concurrently. Furthermore we discuss how nodes can learn the best strategy on their own and achieve a network-wide configuration with various algorithms capable of outperforming statically assigned heterogeneous configurations.

The presented socio-economic approach in this chapter has been introduced by Esterle et al. [31] and further elaborated on in the articles by Esterle et al. [32, 33]. Assigning algorithms heterogeneously and learning optimal strategies for each camera has initially been presented by Lewis et al. [53] and elaborated in Lewis et al. [52].

3.1 Problem Definition

This thesis deals with the problem of tracking up to m distinct objects within the aggregated FOV of a set of fixed cameras C in a network. Moreover, the objects should not simply be tracked by the network in a distributed fashion, but also the global performance, balancing tracking performance against communication overhead, should be maximised. Additionally, the cameras should behave according to their local and current situation in order to perform best towards a defined goal. Parts of the following problem definition have also been given in [31, 33].

As the resources of the employed cameras are limited but the number of tracked objects should be maximised, only a single camera is responsible for tracking an object at a time. This also applies when multiple cameras "see" the object at the same time. Thus, the network must distribute the tracking responsibility for a maximum of m objects among n cameras at any time. This tracking responsibility of camera i for object j can be expressed by j being a member of the set of objects "owned" by i, which we denote as O_i. When we say that camera i owns object j, we mean that it is responsible for tracking it, has the right to track it and may sell it to other cameras. However, since our cameras are controlled by autonomous software agents, they make independent decisions about which object(s) in O_i they attempt to track. The decision of camera i to attempt to track object j is expressed as the binary function $\phi_i(j)$.

We assume that a camera can track up to k objects simultaneously. Tracking k objects therefore does not exceed a camera's resource limitations, and furthermore not reduce its tracking performance. In the presented analysis it is assumed that the number of objects tracked by a camera is less than k. Thus, a conservative limit on the number of objects would be $m \leq k$. When camera i attempts to track object j ($\phi_i(j) = 1$), a tracking module is initialized with a description of that object to initially detect and afterwards track the object within the FOV of the camera. The tracking performance depends on various factors, such as object descriptor, distance, orientation, partial occlusion and so on. In this thesis single camera tracking is simplified and all these factors are subsumed in a visibility parameter v_j, which is determined by the distance and angle of the observed object to the observing camera. The tracking performance is estimated by a confidence value c_j. Both values, c_j and v_j, are between 0 and 1 as soon as the observed object is within the FOV of a camera, otherwise they are 0.

The handover of the tracking responsibility is a local mechanism in a camera network, i.e. only the (small) set of neighbouring cameras can contribute to the handover decision. The vision graph ($G_v = (C, E)$) expresses such neighbourhood relations in the camera network. Two cameras i and q (both members of the set of cameras C) are connected in G_v by a directed edge $e_{i,q} \in E$, if they have an overlapping FOV. We extend this basic definition of G_v to non-overlapping cameras as well, by introducing an edge, if a moving object in camera i can appear in camera q within some time. Weights of the edges can be used to express the likelihood and rates of the object's re-appearance (cp. Section 3.2.2). Thus, to reduce communication for deriving the handover decision at camera i, it is sufficient to exchange information only among i and its adjacent cameras $N_G(i)$. Receiving a request to take over the responsibility of tracking an object is accompanied with increased resource consumption due to searching the object within the FOV. As a consequence, the proposed approach not only reduces the communication, but also the overall resource consumption within the entire network because only a small set of neighbouring cameras contributes to the handover decision process.

To enable cameras to decide when and to whom to advertise objects, we implement a set of strategies S. Each strategy $s \in S$ gives rise to one out of two conflicting objectives: to maximise the tracking performance or minimising the communication with other cameras in the network. Initially, we study the performance of assigning the same strategy to all cameras homogeneously. We compare these results with a heterogeneous assignments, where at least two cameras in the network behave differently. This results in γ^n possible assignments including all homogeneous and heterogeneous combinations, where γ denotes to the number of strategies in the set S and n represents the number of cameras in C respectively. We also refer to a specific assignment of strategies on the network level as *configuration*. Having this large set of configurations allows a human operator to select a single configuration for the network. Due to uncertainties in the camera network as well as object movement, making this selection is almost impossible without knowing the camera topology and the movement patterns of the objects to be tracked. Therefore, each camera is required to select its own strategy based on available, local observations. We show that the proposed distributed learning technique, implemented at each camera agent, allows the network as a whole to perform better when compared to all homogeneous configurations and most heterogeneous configurations.

A complete list of all symbols used in this thesis can be found in the appendix.

3.2 Market-based Approach

Recently developed distributed tracking applications apply various control mechanisms to assign tracking responsibilities. They rely on *a priori* induced knowledge about the network topology and/or frequent exchange of information among the participating cameras. Our novel approach overcomes these limitations and is able to achieve robust, flexible and scalable multi-camera control with low computation and communication overhead. Each camera in the network tracks objects, transfers these tracking responsibilities to other cameras, and estimates its own vision graph using a novel socio-economic approach. The term socio-economic is used since it is inspired by social interaction and economic principles. To accomplish this, we induce our cameras with autonomous, self-interested software agents, capable of exchanging responsibilities for tracking objects in a market mechanism in order to maximise their own utility. Utility is generated by tracking objects but also used as artificial currency in the market. Whenever such a handover is required, the camera initiates an auction, containing a description of the corresponding object. Other cameras receiving this auction initiation try to detect the object within their own FOV. This also means, cameras will not track or even detect objects for which no auction has been initiated. In case the object has been detected, this camera can attempt to receive tracking responsibility by submitting a bid for this object to the auction. Applying this market-based approach ensures each camera only tracks those objects generating a high utility. Additionally, resource expensive tasks such as (re-)detection is kept at a minimum. Monitoring completed handovers allows cameras to learn local neighbourhood relations and build the vision graph of the network online. We use artificial pheromones, inspired by the ant foraging process, to grow this vision graph over time. This allows cameras to forget about previously neighbouring cameras in a changing network topology. This entire process is illustrated in Figure 3.1.

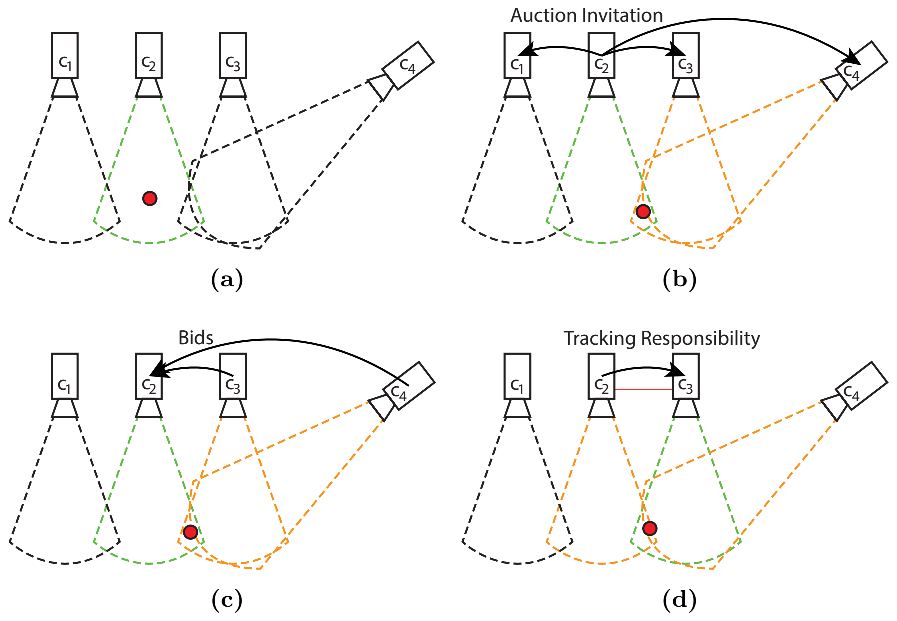


Figure 3.1: Illustration of the market-based handover approach: an object of interest is tracked by camera c_2 in Illustration (a) indicated by the dashed green line representing the FOV. c_2 initiates an auction for an object as it is about to leave the FOV in Figure (b). Bids for taking over tracking responsibility are sent by camera c_3 and c_4 in Illustration (c) which have the object within their FOV as indicated by the orange dashed lines representing their FOV. c_3 wins the auction and the tracking responsibility is transferred from c_2 to c_3 in Illustration (d). A link in the vision graph is created (indicated as red line between c_2 and c_3).

Our approach offers several significant benefits: it is fully decentralised, requires only the exchange of local information, is computationally inexpensive, supports online processing and does not require any a priori knowledge about the camera network or objects of interest. As a result, the system is highly robust and works in dynamic environments where a camera can be added or removed from the network at any time without affecting any other parts of the network. The approach presented takes inspiration from both social and economic systems, and is based on two distinct concepts. First, the allocation of objects to cameras makes use of a market-based approach, similarly to those described in Section 2.2. Second, a pheromone-based mechanism inspired by social interactions in ant colonies is used to build the vision graph online, based on the locally observed trading activity. This is then used to determine communication between cameras. The ant inspired approach is similar to ant colony optimisation [24], where artificial pheromones are used to find good (i.e., short) paths in a network. However the novel use of artificial pheromones to enable targeted marketing is a previously unexplored idea, which enables the efficient management of the trade-off between communication and utility. Additionally, the approach is robust to dynamics and inherently scalable. Therefore, we believe it has significant potential for a range of decentralised applications, of which distributed smart cameras are one example. The presented socio-economic algorithm is implemented locally in each camera and executed by each camera autonomously.

3.2.1 Utility and Market Mechanism

To trade objects or persons of interest as goods in a virtual market, a value has to be assigned to them. This value of an object to be tracked can be extracted from any quantifiable criteria. In tracking this could be the number of re-identified features, the size of the bounding box or the similarity of the colour histograms between the detected object and the initially selected model.

In this thesis, we assume instantaneous utility of camera i and its set of owned objects O_i is given by

$$\begin{aligned} U_i(O_i, p, r) &= \sum_{j \in O_i} u_i(j) - p + r && (3.1) \\ &= \sum_{j \in O_i} [c_j \cdot v_j \cdot \phi_i(j)] - p + r && (3.2) \end{aligned}$$

where $\phi_i : O_i \rightarrow \{0, 1\}$ is 1 if camera i attempts to track object j and 0 otherwise. v_j represents the visibility and c_j represents the confidence of the camera on the given object j respectively, where v_j and c_j are between 0 and 1. In addition to utility earned by tracking objects, a camera b may make a payment to another camera s in order to “buy” the right to track an object from that camera. This requires that the “selling” camera s already itself owns the object. If an exchange is agreed on, then the object is removed from O_s and added to O_b. p denotes the sum of all payments made in trades in a certain time step, and r conversely denotes the sum of all payments received.

To facilitate the exchange of objects, we propose the use of Vickrey auctions [88] hosted by the selling camera. The Vickrey auction, also known as the *second price sealed bid auction* is a single sided auction where bidders make one sealed bid for a single item. The auctioneer awards the item to the highest bidder, but at the price bid by the second highest bidder. The advantage of the Vickrey auction from an implementation perspective is that it has a dominant strategy for bidders: to bid one’s truthful valuation, regardless of the strategies of the other bidders. In contrast with other mechanisms, this removes the need for cameras to possess adaptive bidding strategies, or be required to learn a high performing context-dependent strategy. In common with other market-based control systems (e.g. [54]), currency is an artificial construct used as a tool for system management; no real money is used.

In our model each camera, in the absence of any neighbourhood information, broadcasts information about the objects it is currently tracking in order to solicit bids. Each camera i, upon observing such a broadcast, determines the likely value in case of receiving the right to track the object and if this value is positive, subsequently responds privately to the broadcasting camera with its bid. In order to determine the likely value, each camera receiving an auction invitation checks whether an object within its current FOV matches with the received object description.

3.2.2 Pheromone-based Vision Graph Generation

One of the key advantages of our approach is that it does not require the vision graph to be known *a priori*, since relative utility of the cameras is used to determine which camera the object should be handed over to. However, the broadcast method used to support this decision is inefficient in terms of communication overhead. For this reason, we use a pheromone-based method for building the vision graph online, from the trading activity occurring in the market. Initially, any camera in the network could be the neighbour of all other cameras. While this is highly unlikely, the possibility has to be considered. As the cameras learn the vision graph, they may scale down the amount of communication while still achieving high utility, by announcing their objects only to cameras which are their neighbours in the vision graph.

This use of artificial pheromones, built from previous trading activity to guide future marketing activity, is a novel and highly useful method to achieve efficient outcomes in the trade-off between communication and performance. Since the pheromones are both reinforced and evaporate over time, changes in the topology of the underlying vision graph during runtime can be adapted in a robust manner, and the loss or addition of individual cameras does not affect the wider system. This will be discussed in more detail in Section 3.4. Since marketing communication can be concentrated on only a small number of relevant camera nodes, our socio-economic approach allows significantly improved scalability.

In the presented approach, vision graph information is generated in a distributed fashion and only local information is stored in cameras. We therefore define for each camera i an adjacency list, E_i, the set of all links (or edges) local to that camera. Each element of E_i is the tuple (x, τ_{ix}), where x is another camera in the network and τ_{ix} is the strength of the link from camera i to camera x. Each camera is initialised with an adjacency list containing tuples from itself to all other cameras in the network, each tuple with a strength value $\tau_{ix} = 0$ for all x. Subsequently, each time camera i successfully sells an object to camera x, the corresponding strength value is increased by a value Δ. In ant colony optimisation, the value of Δ is often determined by the properties of the problem. Although we have not yet investigated the effect of different Δ values in our model, we expect that the properties of the camera network and objects to be tracked will similarly affect optimal values for Δ.

However, following the analogy with pheromone evaporation in ant colonies, over time the strength of the links also decreases, allowing the system to overcome changes in topology or fields of view of the cameras over time. The pheromone update rule is shown in Equation 3.3.

$$\tau_{ix} = \begin{cases} (1-\rho)\cdot\tau_{ix} & \text{if no trade occurs on the edge} \\ (1-\rho)\cdot\tau_{ix} + \Delta & \text{if trade occurs on the edge} \end{cases} \tag{3.3}$$

As in ant colony optimisation, ρ is the evaporation rate parameter, which can be understood as a forgetting factor; higher values lead the pheromone to evaporate faster, enabling the system to adapt to changes quicker, but at a penalty of losing more historical vision graph information. However, our approach here is not ant colony optimisation, since pheromone information is not used to find optimal routes through the network, but instead to represent a social environment of cameras with adjacent fields of view.

3.2.3 Exploitation of the Vision Graph

As the vision graph is built up, the initial communication behaviour can be adapted. Specifically, when advertising an object that other cameras may wish to buy, a camera i sends a message to camera x with probability $P(i, x)$, otherwise it does not communicate with camera i at that time. We consider three different policies of determining these communication probabilities:

1. BROADCAST, which communicates with all available cameras in the network. This approach does not miss any camera but also generates a high overhead since it includes cameras which are not likely to respond.

2. STEP, in which an advertisement is sent to a camera if the strength of the link to that camera is above a certain threshold, otherwise the camera communicates with the other camera with a very low probability.

3. SMOOTH, in which the probability of communicating with another camera is based on the ratio between its link strength and that of the strongest link in its graph.

More formally, when employing a policy, the probability of camera i communicating with another camera x is given by

$$P_{\textsc{broadcast}}(i, x) = 1 \tag{3.4}$$

when using the BROADCAST policy. For STEP, the probability is given by

$$P_{\textsc{step}}(i, x) = \begin{cases} 1 & \text{if } (\tau_{ix} > \epsilon) \vee (\tau_{im} = 0) \\ \eta & \text{otherwise} \end{cases} \tag{3.5}$$

where $\epsilon = 0.1$ and m is the camera with the highest strength value, i.e.,

$$m = \operatorname*{argmax}_{y} \tau_{iy}, \forall y.$$

For the SMOOTH policy, the communication probability is given by

$$P_{\textsc{smooth}}(i, x) = \frac{1 + \tau_{ix}}{1 + \tau_{im}} \tag{3.6}$$

where m is again the camera with the highest strength value.
Both of the presented communication policies illustrate a novel use of ant inspired systems in the computing domain, as a method of managing communication in a distributed network.

3.2.4 Autonomous Camera Control

Putting together the aspects of the utility function of the camera, decision process,trading behaviour and vision graph generation, we specify that each camera in the system behaves according to Algorithm 1.

As indicated in Step 4, the handover algorithm should be repeated at regular intervals to ensure that objects are handed over as close as possible to the optimal time, but without spending unreasonable resources identifying objects in the scene purely for the purposes of determining optimal bids. We define two different schedules to initiate auctions, ACTIVE and PASSIVE.

The ACTIVE auction schedule requires the camera to initiate auctions in very short intervals. This results in the handover of the object very close to the possible optimum with respect to the acquired utility. At the same time, the high frequency of initiating auctions generates a high communication overhead and network-wide resource consumption.

Algorithm 1 The camera handover algorithm

```
1. Object trading of camera $i$
    a) Advertise owned objects to each other camera $x$ with probability $P(i, x)$.
    b) For each received advertised object $j$, respond with a bid at value $u_i(j)$ if this
       is greater than zero.
    c) Accept received bids for each object $k$ for which $u_i(k)$ is less than the highest
       received bid. For each accepted bid:
        i. Remove $k$ from $O_i$.
        ii. Respond to the camera making the highest bid, informing it of the required
            payment, the value of the second highest received bid.
        iii. Increment the utility of the camera by the value of the second highest bid.
    d) For each object $l$ for which the bid sent was accepted, add $l$ to $O_i$ and deduct
       the payment amount from the utility of the camera.
2. Vision graph update of camera $i$: Update $\tau_{ix}$ for all $x$ according to Equation 3.3.
3. Tracking decisions of camera $i$: Select which objects in $O_i$ to track in order to
   maximise $U_i(O_i)$.
4. Repeat at regular intervals.
```

The second auction initiation schedule is referred to as PASSIVE. In this schedule, the camera only initiates auctions whenever the object is about to leave the FOV of the camera. While the PASSIVE approach has lower communication overhead for an object, it might not allow to acquire the maximum utility that would have been possible with a handover of the same object initiated earlier.

Combining these two auction initiation schedules with the three previously discussed communication policies results in six different strategies. An operator, trying to balance the trade-off between communication overhead and tracking performance in the system, can select one of these six strategies. To start our distributed tracking application, objects of interest have to be defined. In our system, this basic mechanism is accomplished by a human operator who has to connect to a remote camera and select the object or person to be tracked in a user interface. This user interface is only required to initiate the tracking process and does not act as a central component or is not needed in any way besides initialisation, to support our approach. An alternative approach on initialisation using the market-based mechanism to initiate the tracking process is outlined in the following section.

3.3 Initialisation of Tracking Responsibility

The socio-economic approach described in this chapter allows distributed tracking of objects in a network of smart cameras while keeping the communication overhead low. Nevertheless, the described approach assumes initial assignment of the tracking responsibility for each object to a specific camera. This could be done by a human operator or based on a predefined set of objects of interest. In case the network of autonomous smart cameras is deployed in an area, where objects are unknown at deployment and the network should track all objects visible in the aggregated FOV, manual selection may not be possible. This section will outline simple adjustments to the original market-based approach and a possible implementation of the same in a network of autonomous cameras to initially assign tracking responsibilities without additional bargaining effort.

3.3.1 Outline of the Approach

When a network of smart cameras is deployed in an area where the objects are unknown at start up, the market-based handover approach can be used to initially assign tracking responsibilities. To allow this initial assignment in a dedicated initialisation phase, three simple adjustments have to be made to the original market-based approach:

1. Cameras have to be able to autonomously detect moving objects and extract a description. Furthermore, they have to identify and label the objects coherently.

2. Multiple concurrent auctions by different cameras are allowed for the same object. In contrast to the original approach, initiating an auction does not grant "ownership" of the object to be tracked to the camera.

3. A camera may decide on its own to participate in an auction or not. It is of utmost importance, that for each object a camera initiated an auction for, this camera also has to participate in all received auction invitations for the same object.

Figure 3.2 illustrates the protocol to assign initial ownership among two cameras. In this illustration, all actions of these two cameras related to the auction initiated by Camera 1 are depicted in red and in blue for the auction initiated by Camera 2 respectively. When all cameras are switched on, they immediately try to find, identify and label objects to be tracked in their FOV. For every object j each camera initiates an auction. This might result in multiple auctions running concurrently on various cameras. In our example, Camera 1 and 2 have both spotted object j in their FOV and initiate an auction at the same time. Both cameras receive the invitation of the other camera and therefore assess the value of object j. This value is submitted as a bid to the other camera. Both cameras now concurrently define the winner of the auction. Since bids are based on utility, which combines visibility and tracking confidence, the object will be assigned to the camera having the best view of the object among both cameras.

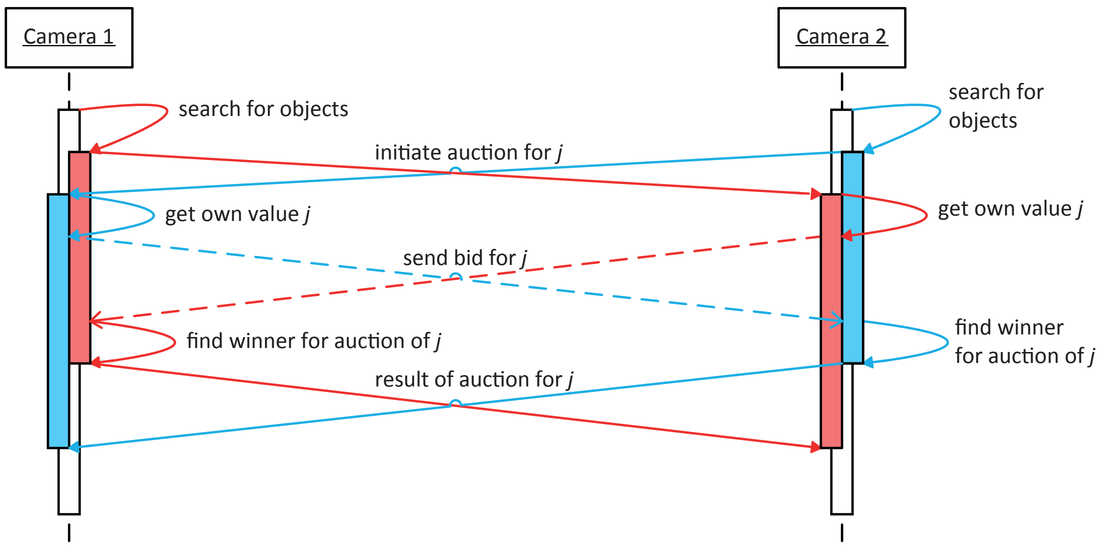


Figure 3.2: The illustration depicts the initialisation process using our socio-economic approach resulting in the assignment of the object to the camera with the best view. This is accomplished with only 3 messages per camera per object. Red and blue lines and bars indicate actions and messages due to the initialisation process of camera 1 and camera 2 respectively.

The proposed approach only needs a low number of exchanged messages among the participating cameras. For a network of n cameras having a total of m objects within their FOVs, the total number of messages to assign all objects in the network is given by

$$\#messages = 3 \cdot (n-1) \cdot m. \tag{3.7}$$

This maximum is only reached if all cameras see all objects at the time of initialisation.

3.3.2 Discussion

There are various benefits and drawbacks of the proposed approach for initialisation. The main benefit of the approach is that all objects are allocated to the camera having the best view. Only in the highly unlikely event of two cameras having the exact same valuation, the assignment of tracking responsibility cannot be decided. In this case, the cameras would need to re-initiate an auction for the object.

It has to be a fundamental rule that all cameras are committed to their submitted bids. This is important because a camera i winning an auction from another camera q would mean other cameras did not win the auction initiated by camera i for the same object. Indeed this would mean that, if this rule is broken, a camera could receive the tracking responsibility from its own initiated auction and hence without making any payment for a submitted bid in an auction by another camera for the same object. Therefore, every camera has to assure payment to other cameras in case auctions are won. Additionally, it is important that an object is only assigned to a single camera after the initiation phase.

A problem arises when more than two cameras have a view of the same object at start up. In this case the winning camera would have to pay multiple cameras for the same tracking responsibility. Indeed a camera would be required to spend multiple times its actual valuation to receive a tracking responsibility. Since cameras broadcast their auctions for initial assignment, each camera will receive as many auction invitations for a specific object as other cameras have this object within their FOV. To overcome the problem of overspending in case of x cameras, where $x > 2$, having the same object within their FOV, all auctioneering cameras may only be allowed to request a certain fraction of the initial bid b from the winning camera. The actual payment by a camera to all auctioneers is defined as

$$pay = \frac{b}{x-1}. \tag{3.8}$$

While initialisation of tracking systems is a research area of its own, we are able to accomplish this complex task using our market-based approach with only a minimum of exchanged messages per object. The required considerations and benefits of this approach have been outlined but are beyond the focus of this thesis.

3.4 Uncertainties in the Camera Network Topology

When defining the vision graph, and hence the neighbourhood relations, offline, the camera network is not able to adapt to changes within the network. Not only does each camera rely on all of their neighbouring cameras to be fully functional at all times, but also every camera assumes that other cameras in the network do not change their position or angle.

Unfortunately, smart camera networks are prone to uncertainties, changing the topology of the network. We consider three main types of uncertainties which have an effect on the topology of camera networks during runtime: (i) cameras being added to the network, (ii) cameras failing for a certain, limited time or dropping out entirely, and (iii) cameras changing their pose. While adding a camera to the network might not be considered a major problem, since it can only increase the overall utility of the network, removing as well as changing positions and viewpoints do affect the network performance directly and might occur due to external actions, failures or vandalism.

We extend our previously introduced socio-economic handover algorithm and demonstrate how this improves adaptivity and robustness substantially , when two important assumptions are relaxed. Specifically: (i) we no longer assume instantaneous handover and instantaneous communication between cameras, and (ii) we introduce uncertainties to the network, to better reflect realistic real world situations.

In this context, we show how the network can automatically deal with cameras failing as well as being added to the network during runtime and still maintain high performance. Furthermore, we show that the smart camera network can cope well with single cameras changing their poses during runtime.

3.4.1 Adding Cameras

In case of predefined neighbourhood relationships in a network of smart cameras, adding a new camera at a random position with a random orientation at an arbitrary time step required explicit incorporation into the network. Otherwise this new camera might not be able to gather valuable utility fo the entire network. We show that our market-based approach is able to incorporate newly added cameras during runtime and therefore increase the overall network utility (i.e., tracking performance of the entire network) when compared to a static approach, where the neighbourhood relations are defined at the deployment.

3.4.2 Failing Cameras

Cameras failing during runtime may have a severe impact on the tracking performance of the network. In such a case, objects might get lost for entire sub-networks. We consider two different types of failing cameras: *permanent* and *transient* failures. We illustrate, that the proposed socio-economic algorithm is capable of dealing with both types of failing cameras and keeping track of objects in separate sub-networks with missing links. This allows the network a continuous generation of utility in contrast to a static approach, where the neighbourhood relations are not adapted after deployment.

3.4.3 Changing Cameras

Changing the pose of a camera comprises the orientation as well as the position of the camera. Changing a camera's pose might render it irrelevant for its previously known neighbourhood and more valuable to a completely different area of the network at the same time. This is especially problematic when the old, as well as the new neighbours are not notified about the occurring changes. We consider four different types of possible changes.

- **Position:** The cameras' location has changed therefore the camera might observe a new area of the environment or the previously monitored area from a different viewpoint.
- **Orientation:** The cameras' orientation has changed and therefore the camera observes a new area of the environment.
- **Range:** Range defines how far objects can be from the camera observer while still being observed. This factor is closely related to *zoom*.
- **Zoom:** The camera is now able to observe a wider or more narrow area of its original FOV.

Essentially, changing the camera's pose means the observed area changes as well. The new observed area can be a partially, or even completely, new region or it only becomes a smaller part of the original observed space.

3.4.4 Non-instantaneous Handover and Bidding Periods

In this section we relax two simplifying assumptions about the behaviour of the camera network to create a more realistic setup as follows: (i) we no longer assume instantaneous handover and instantaneous communication, (ii) we consider uncertainties regarding the cameras in our networks. Due to these relaxations, the camera handover algorithm from Section 3.2.4 needs to be adapted. Instantaneous communication and handover favours cameras seeing the auctioneered object first. In contrast, we prefer the camera with the best view to win the auction. In order to allow multiple cameras, seeing the object within a very short time span but not at the exact same time, to participate in the auction, a non-instantaneous auction is required. To implement non-instantaneous handover, we have introduced Step 1c to our algorithm described in Algorithm 2. The introduced durations of auctions t_a, allows cameras not having the advertised object within their FOV at the exact same time to still participate in the same auction. Additionally, this facilitates dealing with non-overlapping FOVs. Algorithm 1 allows the network to deal with the uncertainties regarding the network topology over time. However, this is based on the evaporation rate of the artificial pheromones. Consequently, until the link to a changed camera has not evaporated, the auctioneer will still favour this camera when soliciting bids. In order to deal with uncertainties in the network topology faster, we introduced Step 1f to the algorithm. To ensure that bids can be received by the auctioning camera, t_b has to be longer than the duration of an auction t_a. In case no bids arrived at the auctioneering camera and as soon as the timespan t_b has elapsed, the object is advertised to all cameras using broadcast communication.

Algorithm 2 Extended camera handover algorithm

```
1. Object trading by camera i:
    a) Advertise owned objects to each other camera $x$ with probability $P(i, x)$.
    b) For each received advertised object $j$, respond with a bid at value $u_i(j)$ if this
       is greater than zero.
    c) After receiving the first bid, wait for $t_a$ time steps to receive more
       bids from other cameras.
    d) Accept received bids for each object $k$ for which $u_i(k)$ is less than the highest
       received bid. For each accepted bid:
        i. Remove $k$ from $O_i$.
        ii. Respond to the camera making the highest bid, informing it of the required
            payment, the value of the second highest received bid.
        iii. Increment the camera's utility by the value of the second highest bid.
    e) For each object $l$ for which the bid sent was accepted, add $l$ to $O_i$ and deduct
       the payment amount from the camera's utility.
    f) If no bids have been received after $t_b$ time steps, advertise owned
       objects to every other camera $x$ with probability $P(i, x) = 1$.
2. Vision graph update: Update $\tau_{ix}$ for all $x$ according to Equation 3.3.
3. Tracking decisions of camera i: Select which objects in $O_i$ to track in order to
   maximise $U_i(O_i)$.
4. Repeat at regular intervals.
```

Besides our socio-economic inspired communication policies discussed in Section 3.2.3, for comparison we also implemented a STATIC communication policy, which uses a predefined vision graph. A camera x using the STATIC communication policy only advertises objects to its known neighbours i with a probability given in Equation 3.9.

$$P_{\text{STATIC}}(i, x) = \begin{cases} 1 & \text{if } \tau_{ix} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

The predefined vision graph does not employ our ant-inspired approach and therefore does not have the concept of changing link strengths.

3.5 Heterogeneous and Learned Strategies for Distributed Tracking Systems

As discussed in Section 3.2.4, there are six different behavioural strategies available for the camera nodes disposal. The strategy determines the level of marketing activity the node undertakes, given the learnt vision graph and influences the communication overhead as well as the tracking performance. While communication intensive strategies typically obtain higher tracking performance; strategies with lower communication obtain the opposite result. The trade-off between the communication overhead and the achieved tracking performance is highly dependent on the scenario. This is due to varying camera positions and different object movements. Additionally, cameras often operate inefficiently since the homogeneous deployment of strategies forces a *one size fits all* approach, despite local differences in the vicinities of the cameras. In this section we discuss how permitting heterogeneity between cameras in terms of their strategies, enables the network to obtain more Pareto efficient global outcomes. At this stage, we consider the cases where (i) all cameras need not employ the same strategy at all times (i.e., a camera does not change its strategy), and (ii) a camera may learn which strategy to use during runtime. We therefore refer to the strategies employed by the cameras across the network as the configuration of the network. Based on the variation in the employed strategies across the network, we may describe two types of configurations:

1. Homogeneous: Refers to a network configuration where all cameras use the same marketing strategy.

2. Heterogeneous: Refers to a network configuration where at least two cameras use different marketing strategies.

Although heterogeneity can improve global efficiency, given unlimited possibilities for camera network deployments and accompanying environmental dynamics, identifying by hand the most appropriate behaviour for each node in a given scenario and at a particular point in time is not feasible. We have refined Prasath et al's [72] key issues for engineering heterogeneity in self-organising systems, to fit the context of market-based distributed smart camera networks.

1. Are there heterogeneous configurations which are capable of achieving results more Pareto efficient when compared to results achieved by a homogeneous configuration?
2. How can self-interested smart cameras find such a system-wide configuration by themselves to achieve a more Pareto efficient outcome, given a particular scenario?

To overcome these problems we propose using online learning algorithms, specifically multi-armed bandit problem solvers (e.g., [5]). We endow each of our cameras witch a so-called bandit solver to learn the appropriate strategy for each node during runtime. These bandit solvers balance exploitation behaviour, where a camera achieves high performance by using its currently known best strategy, with exploration, where the camera explores the effect of using other strategies to build up its knowledge. By employing bandit solvers in each camera, we are able to obtain near Pareto efficient global outcomes in many cases. In other cases, the dynamic nature of the online learning algorithms actually extends the Pareto efficient frontier, improving upon the best static configurations. While in principle there are many possible marketing strategies which could be conceived of and used by a bandit solver, for comparison purposes, we focus on the six strategies previously proposed in Section 3.2.4.

3.5.1 Pareto Efficiency of Heterogeneous Networks

Despite the six available marketing strategies presented in Section 3.2.4, we assumed, all cameras in the network use the same strategy, i.e., the networks have a homogeneous configuration. In this section we consider the case when individual nodes (cameras) in a network can use different strategies to govern how they advertise their auctions. Permitting this heterogeneity in the network enables nodes to specialise to their local situation and hence allows for a wider range of global outcomes when compared to the homogeneous case. We speculate that optimal heterogeneous assignment of strategies can lead to the global performance of the network being strictly better in terms of both of the considered objectives. This would extend the previous Pareto efficient frontier with respect to the network-wide observed trade-off between communication overhead and achieved tracking performance.

However, heterogeneity itself does not necessarily lead to better outcomes. It is also possible that nodes specialise wrongly, leading to a strictly worse global outcome when compared to any homogeneous case. Indeed, when considering all possible heterogeneous configurations for a given network, the number of possible configurations increases greatly compared to the homogeneous-only case. Having γ different strategies and n cameras in the network, an operator has to pick one configuration out of γ^n possible configurations.

3.5.2 Decentralised Online Learning of Pareto Efficient Configurations

By permitting heterogeneous configurations of camera networks we expect global outcomes to be obtained which are more Pareto efficient than in the homogeneous case. However, it is not clear which particular heterogeneous configuration should be chosen, in order to achieve an efficient global outcome, in a particular scenario. We are faced with the problem of choosing from numerous combinations, out of which only a few are Pareto efficient, which configuration the network should adopt.

This could be tackled as a classical offline search problem at deployment of the smart camera network. However, doing so would assume that we know the characteristics of the scenario in advance, including camera placement and orientation, expected object movement patterns and runtime failures or additions (e.g., as studied in [32] and discussed in Section 3.4). Indeed, this lack of *a priori* scenario knowledge is a key problem characteristic motivating this approach. Therefore, we extend our previous approach, where cameras used a single strategy for the entire deployment, by enabling individual cameras to learn behaviours online during run time.

This means, a node can select its marketing strategy autonomously using an online learning technique. This allows the camera to adapt its strategy during runtime based on local feedback. This local feedback comprises the auction invitations sent by the node and the tracking performance measured by the node (as opposed to the equivalent metrics for the network as a whole).

Learning Efficient Configurations using Bandit Solvers

From the perspective of an individual camera, its task is to select a marketing strategy from those available, which maximises its expected tracking performance while minimising its auction overhead over time. We therefore consider that a camera is faced with an online algorithm selection problem [39]. Our approach is to consider this as a variant of the multi-armed bandit problem [5]. This problem is analogous to being faced with n slot machine arms, where each pull of an arm returns a random reward drawn from an unknown distribution associated with that arm. Given m total arm pulls, the task is to select which arms to pull such that the total reward obtained is maximised. If the player were to know the distributions behind each arm, then the player could simply select the best arm for every pull. However, since the distributions are unknown, the player must sample from each arm in order to gain some knowledge of each arm's reward distribution. The multi-armed bandit problem therefore encapsulates the classic *exploration vs. exploitation* dilemma. However, some of the assumptions present in the classic multi-armed bandit problem formulation may not be appropriate in this setting. First, the reward distributions are usually assumed to be static over time, and second it is assumed that the bounds on the obtainable rewards are also known. It does not appear that either assumption can be made in our problem.

Nevertheless, the bandit framework is useful, and each marketing strategy can be considered an arm of a bandit. Each camera node can choose to use one strategy (i.e., pull an arm) at each time step, and can receive a resulting reward, derived from its local metrics. In this way, a camera learns which strategy performs well in its current situation within the scenario, and exploits that knowledge to maximise its performance. There are a number of so-called bandit solving algorithms to be found in literature. In this thesis we consider three well-known bandit solvers: the simple Epsilon-Greedy [87], UCB1, which is known to perform well in static problems [5], and Softmax [84]. Epsilon-Greedy requires an ϵ value to determine the amount of exploration. A low ϵ value ($\epsilon \to 0$) results in random selection of algorithms while a high value ($\epsilon \to 1$) selects greedily the best algorithm, based on the previous rewards. UCB1 requires no parameters. Softmax uses a temperature value τ to steer the probability of selecting an arm based on the expected reward. This means, that high temperatures ($\tau \to 1$) result in a random selection where each arm has nearly the same probability while lower temperatures ($\tau \to 0$) tend to select the arm based purely on the expected reward.

In applying bandit solvers to algorithm selection at the local level in a self-organising system, we must define local reward functions, such that the global system's objectives are achieved. Although, we try to achieve these global objectives, we can only rely on multiple corresponding metrics locally at each node for our reward function. In this chapter, we use a linear combination of the local metrics:

$$reward = \alpha \times utility - (1 - \alpha) \times communication \tag{3.10}$$

where *utility* is the utility function given in Section 3.2, which sums obtained tracking performance over all objects tracked by this camera in the current time step, plus its balance of payments from all trading activity during this time step. The number of messages sent by this camera at this time step is denoted by *communication*. α allows us to change the node's preference in favour of either maximising tracking performance or minimising the communication overhead. Therefore, α may be used to direct local learning such that outcomes at the global level favour appropriate regions of the Pareto efficient frontier.

3.5.3 Camera Level Normalisation by Distribution

The selected strategy for each camera using bandit solvers is dependent on the camera's location, the scenario and the camera's preferences between the two objectives. The local preferences can be customized using the α value. By varying the α value, a preference for the global outcomes on the Pareto efficient frontier can be set. However, the results based on the preferences are biased due to the nature of the observed metrics at the camera level. Ideally, α would be used to weight the two objectives evenly, such that the outcome position on the Pareto frontier can be determined directly by setting α. For example, an α value of 0.25 would lead to an outcome value 25% of the way along the length of the achieved front. In order to achieve this, we would need to normalise both parameters of the reward function. Normalising the first parameter is impossible due to the camera's lack of knowledge about its maximum possibly achievable utility. Since a camera has this information only, when it advertises all objects to all other cameras in the network, it does not obtain this when using a PASSIVE schedule nor the STEP or SMOOTH communication policy. Using a strategy containing one of these policies or the PASSIVE schedule, the camera will not advertise to all other cameras in the network. Hence, the advertising camera does not receive bids from some cameras. These other cameras might however have paid more for the object than actual participating cameras did. The upper bound on the camera's utility is therefore not known, and varies significantly with every time step. Nevertheless, we are able to normalise the second parameter of the reward function, the number of messages sent by the camera, and therefore alleviate the bias effect a little. The upper bound on the value for communication also varies, but in this case only with the number of objects and other cameras currently known to the camera.

We are therefore able to perform some estimated normalisation of the number of sent messages at the local level. Figure 3.3 shows the frequency distribution with which a camera sent messages to other cameras over time, in a typical run of a simple scenario with two cameras and four objects of interest. The results for three more, qualitatively different, scenarios using different bandit solvers are depicted in Figure 3.4, and show similar distributions. Clearly, cameras are less communicative more often than they are more communicative. As it turns out, this skew in the distribution appears to have a large effect on the bias observed in the outcome Pareto front. We are able to account for this skew effect by introducing a normalisation by distribution process into the auction invitation component of the local reward.

More specifically, each camera keeps track of the frequency of sent messages for predefined time intervals throughout its lifetime. When a new auction is initiated, the number of sent out messages is added to those in this time interval. The sum of all messages in this

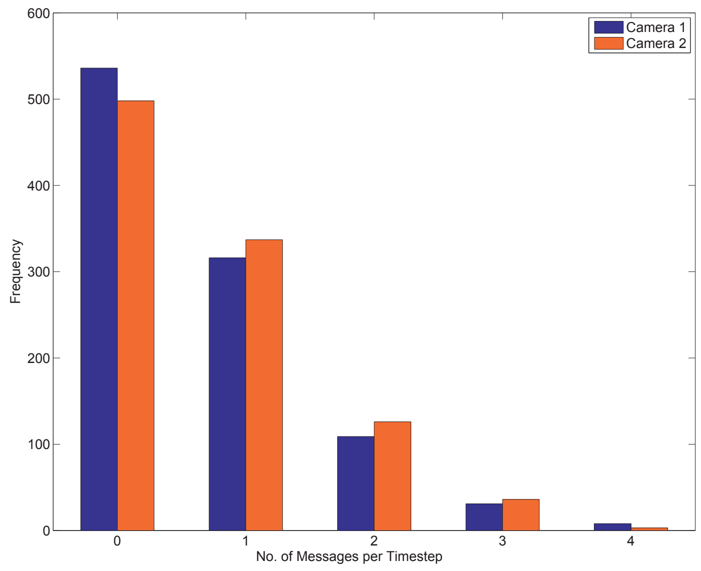


Figure 3.3: Histogram showing the frequency distribution of messages sent per time step in Scenario 1 when cameras use SOFTMAX with a temperature $\tau = 0.1$. Each camera is represented by one bar per bin. $\alpha = 0.5$.

interval is ranked within the historical values. This rank is normalised by the maximum number of messages sent in any time interval. For example, if the number of sent messages in an interval is greater than the largest observed number of sent messages so far, its normalised value is 1. Similarly, if the number of sent messages could be categorised within the middle of previously observed values, it is normalised to the value 0.5.

By normalising in this manner, we expect to obtain a more even spread of outcomes along the achieved frontier.

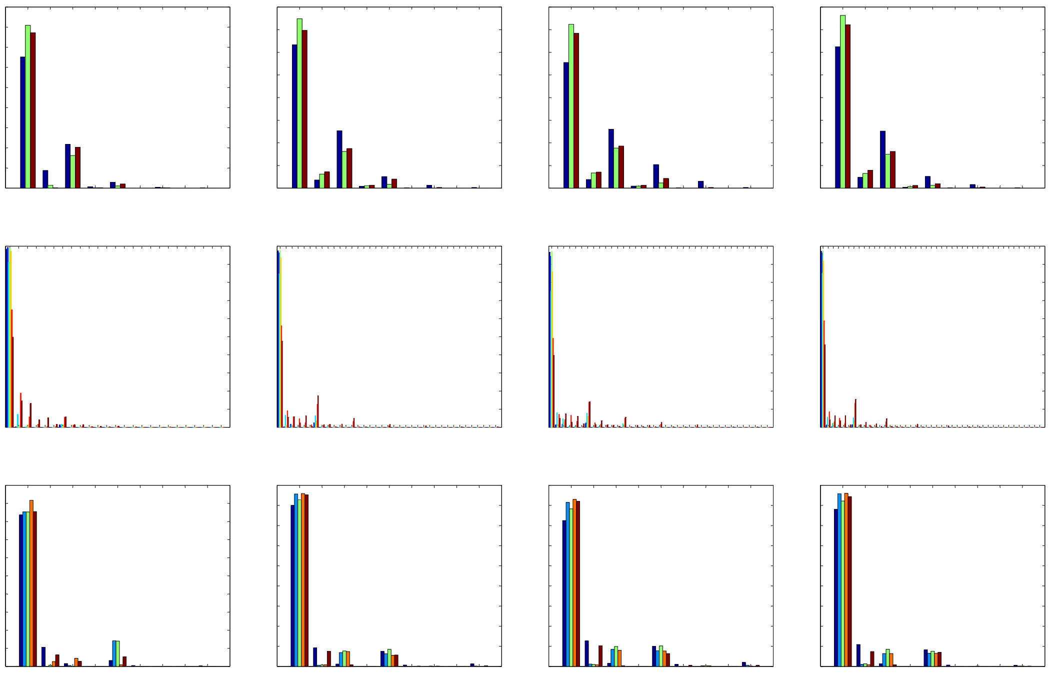


Figure 3.4: A selection of histograms showing the frequency of sent messages. Each camera is represented using one bar per bin. From top to bottom, the rows show scenario 3, 6 and 9 respectively. From left to right, the columns show results when cameras use EPSILON-GREEDY, SOFTMAX with a temperature $\tau = 0.1$, SOFTMAX with a temperature $\tau = 0.2$ and UCB1 respectively. In all cases $\alpha = 0.5$. The axes are as in Figure 3.3.

Fourth Chapter

CamSim - Smart Camera Simulation Tool

Setting up a network of smart cameras in the real world holds various difficulties. First of all, deploying the network itself can be challenging. This is especially true if a larger network of cameras is investigated, as cost can be prohibitive. Furthermore, legal issues vary from country to country and often do not allow to set up cameras in a public space. Setting up cameras only in a laboratory environment would drastically limit the number of used cameras. Finally, cameras and computer vision algorithms are both error-prone. Moreover, the study of self-adaptation and self-organisation techniques through repeatable experiments in a real environment can prove quite difficult to control due to limited robustness. This hinders reproducibility of experiments. To overcome these problems, a simple 2D-simulation tool has been developed for analysis, testing and evaluation of distributed algorithms for self-adaptation and self-organisation of smart camera networks. This chapter describes the simulation tool and its basic features in more detail. The next section gives an introduction and a rough overview of the simulation environment. Afterwards, Section 4.2 explains the user interface, the graphical components as well as the behaviour of objects in the environment. Section 4.3 elaborates on various uncertainties that can be induced in simulation runs. Section 4.4 focusses on the induction of errors which can affect cameras in general (i.e. camera failure) or the computer vision (i.e. tracking errors). Section 4.5 explains how scenarios can be defined in files for repeatability. Our simulation environment is also capable of processing results from real video feeds. This capability is discussed in Section 4.6. An overview on how to use the simulator including possible startup parameters is given in Section 4.7. Finally, Section 4.5 summarises and discusses all scenarios used for evaluation in this thesis. The simulation tool was also presented in [30].

4.1 Overview

The simulation tool, *CamSim*, which arose from this thesis and is available at `https://github.com/EPiCS/CamSim`, was extensively used in our research [31–33, 52, 53]. This simulation environment allows the user to set up a network of virtual smart cameras and infuse them with autonomous software agents. As such, it can be used to test and compare self-organising camera control techniques. The key benefits of *CamSim* are:

- Ease of generating test scenarios with an arbitrary number of cameras and objects.
- Implemented camera control algorithm, allowing different strategies to be assigned to individual cameras [31–33].

- Several bandit solvers are implemented to provide meta-management at the camera level, selecting dynamically between communication strategies at runtime [53].
- Simplified communication using message passing.
- Abstraction of computer vision processing.
- Various object movement patterns.
- Easy adaptation of camera behaviour, including bandit solvers, communication strategies, pheromone learning and movement behaviours, using reflection mechanisms.

To support easy implementation, testing and comparison of distributed algorithms for self-adaptation and self-organisation of the network, various assumptions to relax the real operating environment of multi-camera applications were made. Smart camera networks are usually connected using either Ethernet or Wi-Fi. These communication channels can be highly reliable, and therefore the assumption of a perfect communication channel without any communication loss was made in the simulation. The communication is implemented via synchronous message passing. The controlled environment allows the analysis of deployed algorithms very easily. Due to the simulator's use of discrete time steps, problems and anomalies can easily be identified and debugging is simpler than in real camera networks. Additionally, the exact position of all objects and the states of every camera can be extracted at any time step for further analysis. In case intrinsic and extrinsic camera factors are required, the simulator does not require complicated calibration techniques for cameras. The uncertainties of computer vision algorithms are removed from the simulation environment by abstraction. This allows the user to focus on the development of algorithms, to self-adapt cameras to a certain scenario and to self-organise the entire network of smart cameras.

4.2 Simulation Environment and User Interface

This section describes the basic features of *CamSim*. More details are available in the documentation[1] of the simulation environment.

Since *CamSim* is implemented in Java, it is highly flexible and portable. It is able to simulate a large number of cameras and objects, limited only by available computer memory. *CamSim* can be run with or without a graphical user interface. This allows user

[1] `https://github.com/EPiCS/CamSim/wiki`

interaction and visual inspection as well as batch running of experiments, for example on a cluster. Figure 4.1 shows a screen shot running a simple scenario with five cameras and a single object. The simulation environment has a predefined area for each scenario, depicted as a thin blue rectangle. All objects, illustrated as black dots, and all cameras, shown as green dots, have to be placed within this simulation environment. The labels for cameras and objects can be turned on or off as desired.

The top menu shows various options:

- Select simulation file: Allows to select from three pre-defined scenarios or to select a new scenario file from the disk.
- Start: Starts the simulation with continuous time steps. Each time step lasts for 100 ms when running the graphical user interface.
- Stop: Stops the previously started simulation.
- Step: Moves the simulation one time step forward in time.
- Randomise: Takes the current number of cameras and objects and places them randomly on the simulation environment. All parameters are randomly set as well.
- Cams++: Adds a new, random camera to the environment.
- Cams–: Removes a randomly selected camera from the environment.
- Objs++: Adds a random object to the environment.
- Objs–: Removes a randomly selected object from the environment.
- Save Scenario: Allows the user to save the initial state of a randomised scenario. This can be used to re-run randomly generated scenarios.

4.2.1 Cameras

CamSim depicts every camera as a green dot. Each camera is controlled by a completely autonomous and self-interested agent. As in the real world, each camera has its own field of view (FOV) and a unique name for identification. This FOV is a circular segment, but illustrated as a triangle. This triangle turns from grey to yellow as soon as an object is within the respective camera's FOV. The FOV of a camera can have an arbitrary size and viewing angle. Cameras can communicate with other cameras in the network by using message passing. Since *CamSim* simulates smart cameras, each camera in the

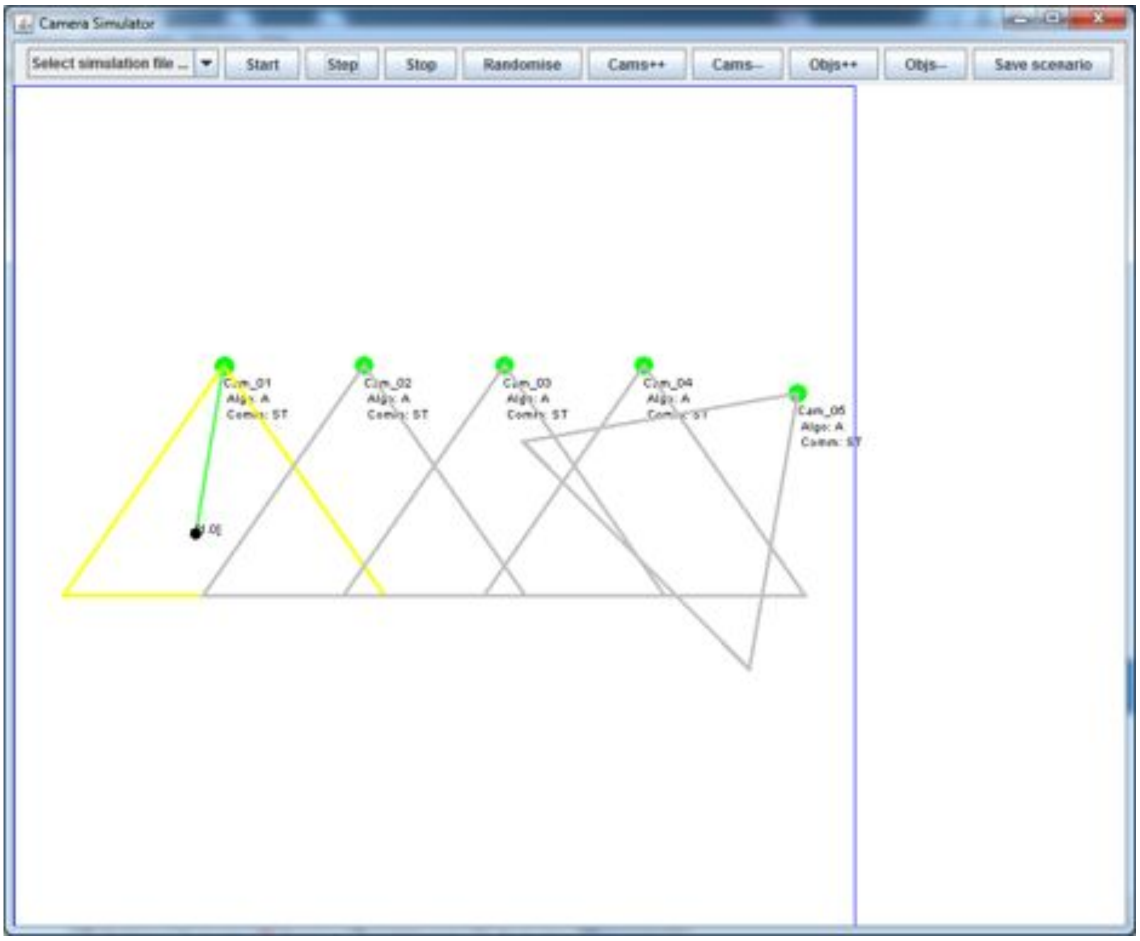


Figure 4.1: Screenshot of the simulation environment *CamSim*. Green dots represent cameras while gray triangles represent the corresponding FOV. Yellow triangles illustrate objects within the FOV of the corresponding camera. Black dots represent objects to be tracked and a green line connects the camera and the currently tracked object. A thin blue line indicates the border of the simulation environment.

network implements its individual behaviour. For the distributed tracking application three different communication policies and two auction schedules were implemented as discussed in Chapter 3. Various events and errors can affect cameras in the simulation. A complete list of all possible events and errors is given in Section 4.3 and Section 4.4 respectively.

The computer vision algorithms are implemented as part of each camera. A camera itself can identify objects within its FOV as well as calculate the visibility of this object. This visibility v_i of an object i is given in Equation 4.1. It is calculated using the inverted Euclidean distance between the object and the camera δ_i, normalized by the range of this specific camera, and the angular position ψ_i of the object to the camera, normalised by the angle of the FOV of the camera.

$$v_i = \delta_i * \psi_i. \tag{4.1}$$

4.2.2 Objects

Objects to be tracked are depicted as black dots. Every object is identified and distinguished among others by its globally unique ID. Furthermore, an object has predefined attributes, such as a starting point, an initial direction and speed. There are currently four different movement patterns implemented in the simulation environment:

- **Predefined:** The object follows predefined waypoints. Each waypoint is approached in a straight line with predefined speed. The waypoints will be looped for the entire duration of the simulation.

- **Random:** The object moves in a straight line towards the predefined initial direction. When hitting the boundary of the simulation environment, the object bounces back in a random direction and again follows this direction in a straight line.

- **Brownian Motion:** The object moves completely randomly. The next position is calculated using the Wiener Process and a standard normal distribution starting at a predefined position [68].

- **Directed Brownian Motion:** While the Brownian Motion calculates the next position of the object, the Directed Brownian Motion applies the Wiener Process to the speed and the direction of the object. For the speed, the normal distribution with the expected value of $\mu = 0$ and the variance of $\sigma^2 = 0.1$ is used, while the parameters $\mu = 0$ and $\sigma^2 = 0.2$ are applied for the direction. This results in smooth changes of speed as well as direction and allows a more natural movement of the objects compared to standard Brownian Motion.

To keep the number of objects constant, objects do not leave the simulation environment but bounce back in a random direction. This is valid for any movement behaviour. New movement behaviours can be defined and added to the simulator by using reflection mechanisms. A green line between the object and the camera indicates the object currently processed by the respective camera.

4.3 Uncertainties

After the deployment of a camera network, various events can alter the network topology. Since the events themselves as well as their time of occurrence is unknown, we refer to them as *uncertainties* as discussed in Section 3.4. In our simulator, various uncertainties can be predefined to occur at specified time steps in the scenario. This allows a user to simulate more real-world behaviour. Table 4.1 gives an overview of different uncertainties, the affected element in the simulation as well as its effect.

Uncertainty	Affected element	Effect
Add	Camera, Object	Adds an element with random or predefined parameters to the simulation.
Remove	Camera, Object	Removes a random element from the simulation.
Change	Camera	Changes a specified parameter of the camera. Such as • Location • Orientation • Range • Viewing angle.

Table 4.1: An overview of the existing events, the elements these events can be applied to and the respective effect.

4.4 Errors

Since smart cameras as well as the employed computer vision algorithms are prone to failure during runtime in real life, *CamSim* allows to introduce errors to the simulations. Errors can either occur in cameras or in the underlying, abstracted computer vision. A camera can fail due to crashed software or failing power supply. In computer vision we focus on errors related to misidentification and misdetection of objects. Misdetection means the object is detected even though it is not in the FOV of the camera. Misidentification refers to an object being wrongly identified as a different one. The current version of the simulation environment does not induce errors on the communication layer.

Cameras: An error occurring in the camera means that the camera will not respond to other cameras or contribute to the network's tracking performance either temporarily or permanently. In case the camera only fails for a limited time, which is randomly chosen between 1 and 100 time steps, the camera can either 'loose' or recover previously learnt knowledge. The probability to fail for the entire duration as well as for recovering previously learnt knowledge can be set as a parameter on start-up.

Computer Vision: Errors related to computer vision affect the detection and tracking of objects within the simulation. There are four possibilities which can occur in every time step to any camera:

- True Positive (TP): An object is detected correctly.
- True Negative (TN): An object is rightly not detected.
- False Positive (FP): An object is detected even though the object is not there.
- False Negative (FN): An object is not detected even though it is within the FOV.

The probabilities for a true positive $P(TP)$ as well as false positives $P(FP)$ can be set as a parameter between 0 and 1 when starting the simulation environment. The probabilities for false negative $P(FN)$ and for true negatives $P(TN)$ are calculated as $P(FN) = 1 - P(TP)$ and $P(TN) = 1 - P(FP)$ respectively.

4.5 Scenarios

Multiple, qualitatively different, scenarios are provided with *CamSim*. All scenarios are basic XML-files and therefore allow simple creation of new scenarios. The structure of these files is defined as follows:

The main elements of an XML-based scenario file are `<simulation>`, `<cameras>`, `<objects>`, `<events>`, `<visiongraph>`, and `<graphnode>`. `<simulation>` defines the size of the simulation environment. `<cameras>` describes a set of cameras where each `<camera>` has specified attributes. `<objects>` is similar to cameras, describing a set of objects. `<events>` describes all events in this simulation. For each event, the attributes are pre-defined. `<visiongraph>` and the corresponding `<graphnode>` element describe the vision graph. An overview of all elements of the XML-based scenario is given in Table 4.2. A full example of such a scenario file can be found in the appendix.

4.6 Real Video Data

Besides completely simulated scenarios, the *CamSim* simulation tool can also process information extracted from real video feeds. To use the data from real videos, each video has to be preprocessed using any arbitrary tracking algorithm. The results have to be stored separately for each video stream. Hence, each file represents a camera and its corresponding tracking results. Each line holds the information of a single object in a specific frame. Each object information is separated by a single space. The first value represents the frame number and the second value indicates the unique ID of the tracked object. The third value and fourth value represent the x- and y-coordinate of the upper left corner of the bounding box. Value five and six are the width and height of the bounding box. The last double value represents the tracking confidence and has to be between 0 and 1, where 1 represents perfect tracking confidence and 0 means the object has not been detected by the tracking algorithm. The list of entries has to be ordered by the frame numbers. If two objects are visible in a certain frame, two consecutive lines have the same frame number with differing object IDs.

The main benefit of the possibility to process real video data is the quick and simple repeatability of experiments. Additionally, various constraints can be relaxed, such as transmission times for network messages or limitations of processing power on smart cameras.

4.7 Startup Parameters

The simulator can be used with or without parameters. Parameters can set various values for the simulations. The most important parameter is setting a scenario file. As discussed, scenarios are specified using XML. This is done without any specific qualifier. An overview of all possible parameters is given in Table 4.3.

Element	Attribute	Description
`simulation`	`max_x`	Maximum x-value for the simulation environment.
	`max_y`	Maximum y-value for the simulation environment.
	`min_x`	Minimum x-value for the simulation environment.
	`min_y`	Minimum y-value for the simulation environment.
`camera`	`name`	The unique name of the camera.
	`ai_algorithm`	Optional. Defines the algorithm used by the camera. Default can be defined as parameter.
	`x, y`	The location of the camera within the simulation environment.
	`heading`	Optional. Direction the camera is oriented in. Default is random.
	`range`	Optional. Defines how ‘far’ a camera can *see*. Default is random.
	`viewing_angle`	Optional. The width of the FOV of the camera. Default is random.
	`comm`	Optional. Which communication policy should be used. 0 = broadcasting, 1 = smooth, 2 = step, 3 = static, 4 = custom policy. Default can be defined as parameter.
	`customcom`	Optional. Needs the fully qualifying name of the policy.
	`FP`	Optional. The probability of a FP. Default is zero.
	`TP`	Optional. The probability of a TP. Default is 1.
	`failing`	Optional. The probability of failing for a random time. Default is 0.
	`reset`	Optional. The probability of a reset and loosing all previously learnt knowledge.
`object`	`features`	The unique features of each object.
	`x, y`	Starting position of the object.
	`heading`	Optional. The initial direction of the object at start. Default is random
	`speed`	Optional. The initial speed of the object. Speed only changes with *Directed Brownian* movement. Default is random.
	`move`	Optional. Fully qualified name of custom movement pattern. Default is *random*.

`event`	`time step`	At what time step the event occurs.
	`participant`	What element is affected. `camera` or `object`.
	`name`	The name or the features of the affected element.
	`event`	The event type that shall occur. Either `add` to add an element, `error` to remove an element or `change` to change the attributes of a `camera`
`visiongraph`	`static`	Defines if vision graph is static or can change over time.
`graphnode`	`name`	For which camera the list of neighbours describes the vision graph.
`neighbour`	`name`	The name of the camera which is a neighbour of the given graphnode.

Table 4.2: An overview of the elements and keywords of the XML-based scenario file.

Even though *CamSim* was developed with smart camera networks in mind, the simulation environment can easily be extended to various distributed networks, since many of the techniques implemented are not specific to smart camera networks. Further details on the usage of *CamSim* can be found within the help files and tutorials at `https://github.com/EPiCS/CamSim`.

4.8 Experimental Study in Simulation

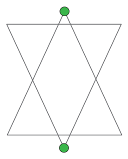
Since we are interested in performing repeatable experiments to investigate the performance of our approaches, we use our simulation environment *CamSim* with different scripted scenarios replicating smart camera networks. We define 17 qualitatively different camera network layouts with a total of 26 scenarios used for evaluation. A layout of a camera network refers to the complete definition of positions, orientations, and FOVs of all cameras in this network. An overview of all scenarios is given in Table 4.4 where the number of the scenario ID corresponds to the layout used for the scenario. We employed each of these layouts to compare the performance of the six different variants of the approach presented. All camera network layouts are depicted in Figure 4.2. Cameras are represented by green dots with a corresponding grey triangle indicating its FOV. The blue arrows in the illustrations indicate the movement of the objects in case of predefined movement paths. If no paths are defined, only random movement patterns have been used. We tested the presented socio-economic approach with a varying number of objects in dif-

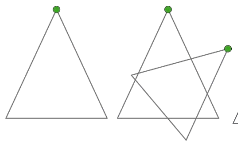
PARAMETER	ATTRIBUTE	DESCRIPTION
`-h`		Prints the help information.
`-t`	`time steps`	Sets the number of time steps.
`-o`	`filename`	Specifies the output file.
	`name`	The name or the features of the affected element.
`-a`	`auction schedule`	Specifies an auction schedule (e.g., `epics.ai.PassiveAINodeMulti`) for all cameras.
`-u`	`communication policy`	Specifies an communication policy (e.g., `epics.commpolicy.Smooth`) for all cameras.
`-f`	`filename`	Creates a summary file for quicker evaluation.
`--no-gui`		Will launch simulator in command line mode.
`-e`	`number`	Probability of camera to fail for a random time between 0 and 100 time steps.
`-r`	`number`	Probability of a camera to reset after failure.

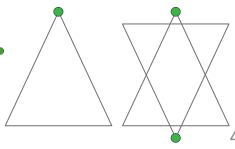
Table 4.3: An overview of the parameters to be provided to the simulator at start-up.

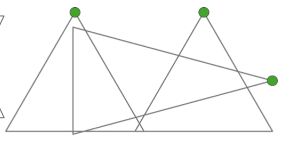
ferent scenarios covering all depicted layouts. Layouts 15, 16, and 17 (Figure 4.2o, 4.2p, and 4.2q) have been used to analyse the behaviour of our approach under uncertainties as discussed in Section 3.4. For these three distinctive layouts we conducted experiments where we added cameras during runtime, removed cameras from the test environment and changed the extrinsic parameters of cameras. The numbers in those layouts indicate camera IDs influenced by uncertainties which occur during simulations. In Table 4.5 an overview of the conducted experiments with uncertainties is given. Scenarios 1 through 8, 9b, 10, and 11a are used to investigate the effects of heterogeneous strategy assignment as well as online learning of best fitting behaviour for cameras as discussed in Section 3.5.1. The number of possible configurations for a given layout are presented in Table 4.6.

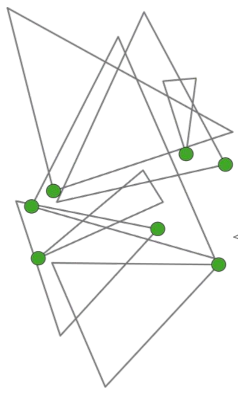
In the simulation, the fields of view of the cameras are modelled as segments (however, visualised as triangles in all depicted scenarios). As already mentioned, the number of objects is kept constant for the duration of each simulation run. For all tested scenarios, all objects have the same behaviour. For all experiments reported in this chapter, each scenario was run for 1000 discrete time steps (each corresponding to one measurement time window). Due to stochasticity, 30 independent runs were conducted for each evaluation.

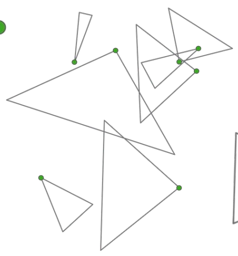

(a) Layout 1

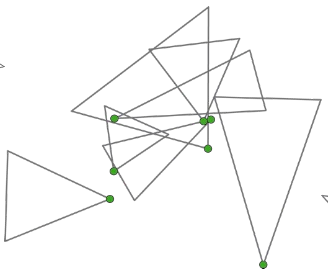

(b) Layout 2

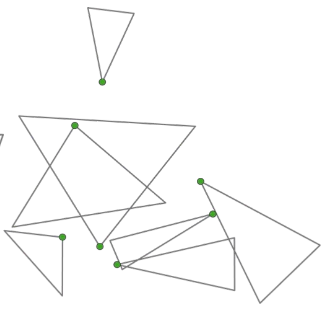

(c) Layout 3

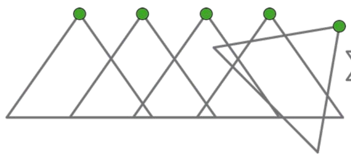

(d) Layout 4

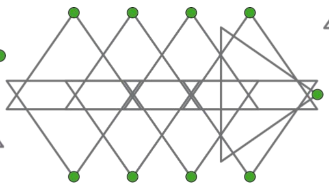

(e) Layout 5

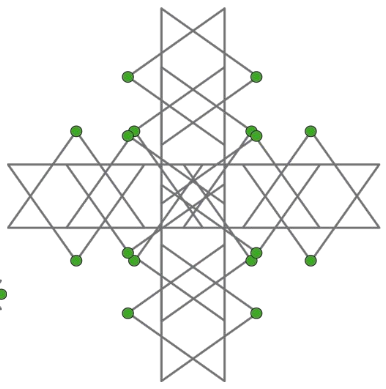

(f) Layout 6

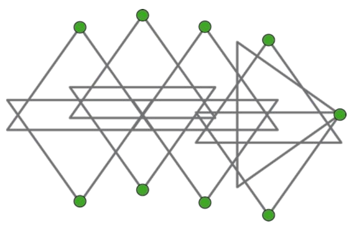

(g) Layout 7

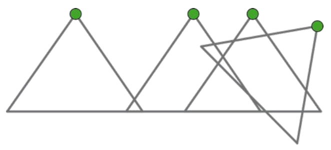

(h) Layout 8

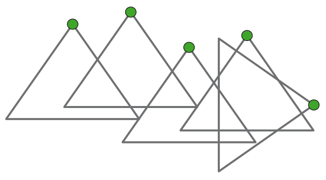

(i) Layout 9

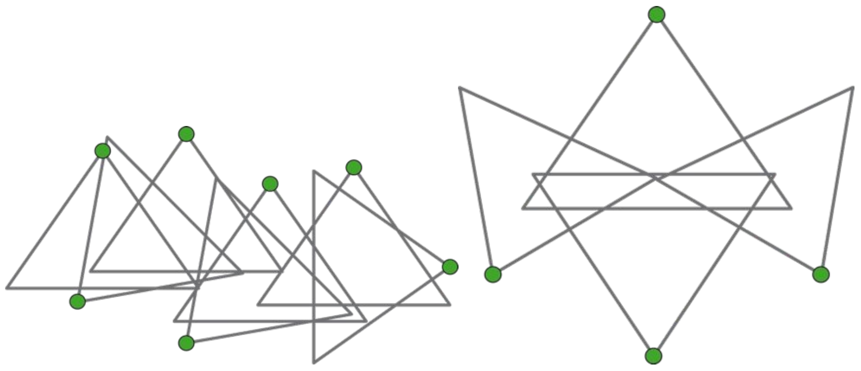

(j) Layout 10


(k) Layout 11

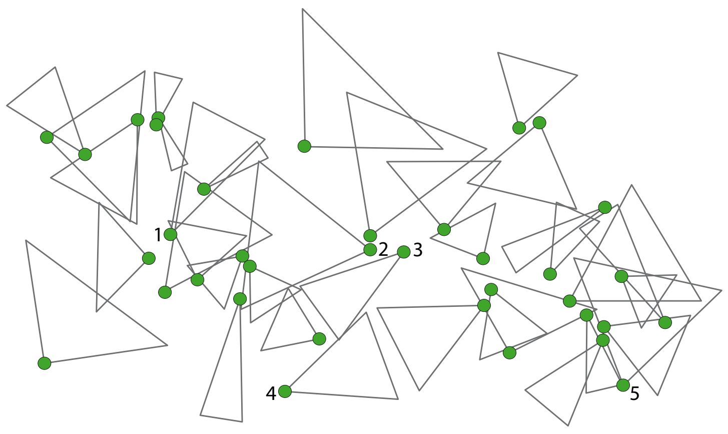

(l) Layout 12


(m) Layout 13


(n) Layout 14



(o) Layout 15 **(p)** Layout 16



(q) Layout 17

Figure 4.2: Illustrations of all the layouts used for scenarios to test our proposed camera coordination algorithm. Each camera is represented by a green circle, with its field of view indicated by the associated triangle. The arrows in Layout 9, 10, 11, 15 and 16 indicate a path alongside objects traversed during the experimental run. In Layout 15–17, numbers indicate cameras influenced by uncertainties. Numbers in brackets indicate cameras not being at the given location in the initially deployed network but were added or moved to the illustrated location dynamically.

Scenario ID	No. of Cameras	No. of Objects	Object Movement Path
1	2	4	Random
2	3	11	Random
3	3	4	Random
4	3	4	Random
5	7	9	Random
6	7	9	Random
7	7	9	Random
8	7	9	Random
9a	5	3	Random
9b	5	3	Predefined
10	9	1	Predefined
11a	16	5	Predefined
11b	16	11	Random
12	9	3	Random
13	4	3	Random
14	5	3	Random
15a	5	4	Predefined
15b	4	4	Predefined
15c	5	4	Predefined
15d	6	4	Predefined
16a	3	4	Predefined
16b	2	4	Predefined
16c	3	4	Predefined
16d	4	4	Predefined
17a	36	31	Random
17b	31	31	Random

Table 4.4: Summary of scenarios used in our study. A random object movement path means that each object moves in a straight line until it reaches the border of the simulation area and bounces back with in a random direction A predefined object movement path means that each object follows a predetermined path through the simulation area. The number of the scenario ID corresponds to the layout used. All simulations lasted for 1000 time steps.

Scenario ID	Layout	Uncertainty
15d	Layout 15	Add Camera (6)
15b	Layout 15	Remove Camera 3
15c	Layout 15	Change Position 3 to (7)
16d	Layout 16	Add Camera (4)
16b	Layout 16	Remove Camera 2
16c	Layout 16	Change Orientation of Camera 2 by -55 degree
17b	Layout 17	Remove Cameras 1, 2, 3, 4, 5

Table 4.5: Overview of the performed experiments with uncertainties to show the robustness of our market-based approach. All uncertainties occurred in time step 518 where no object was visible by the affected camera(s).

Scenario ID	No. of Possible Configurations
1	36
2	216
3	216
4	216
5	$\sim 2.7 \times 10^5$
6	$\sim 2.7 \times 10^5$
7	$\sim 2.7 \times 10^5$
8	$\sim 2.7 \times 10^5$
9b	$7,776$
10	$\sim 1.0 \times 10^7$
11a	$\sim 2.8 \times 10^{12}$

Table 4.6: Summary of scenarios used in our study to show the benefits of heterogeneous algorithm assignment illustrating the number of possible configurations.

Fifth Chapter

Evaluation

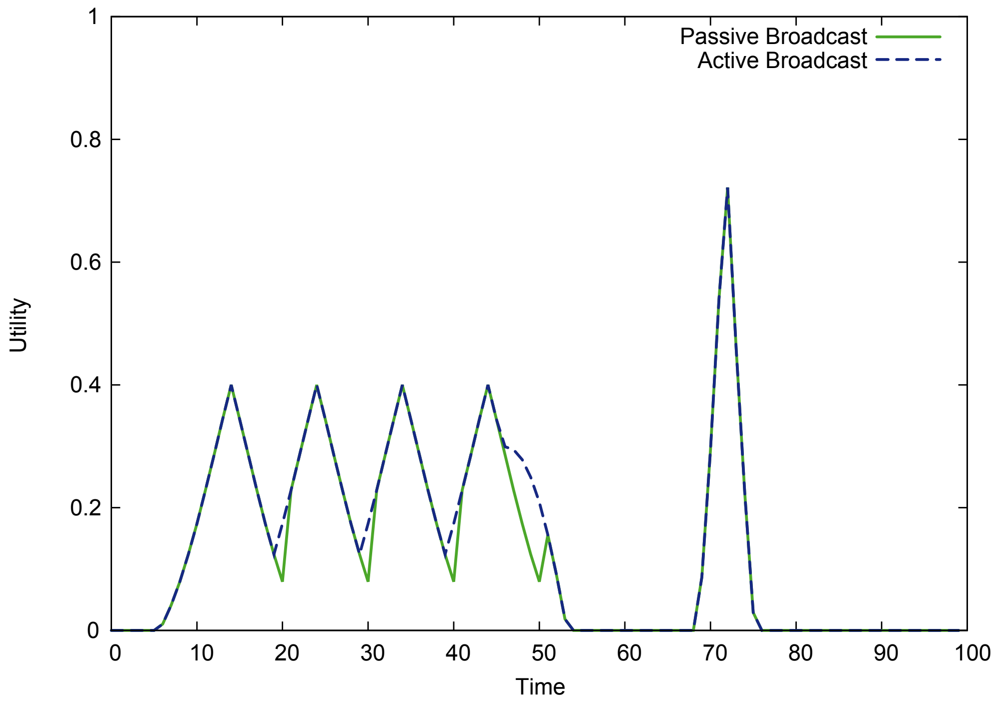
5.1 Evaluation of Socio-economic Approaches

Initially, our two auction schedules ACTIVE and PASSIVE in combination with the BROADCAST communication policy were tested in our simulation environment. In both approaches, each advertisement message is broadcasted to all other cameras in the network. In the ACTIVE approach, each camera advertises every object it owns to the entire network at each simulation time step. This means that other cameras attempt to gain ownership of objects as soon as they enter their FOV. On the one hand this results in the best tracking utility since the camera with the highest utility for an object always has ownership of it. On the other hand the communication between the cameras is significantly higher when compared to the PASSIVE approach. Contrary to this, the PASSIVE approach minimises the communication by sending advertisement messages only when an object is about to leave the FOV of its current owner. Furthermore, due to less initiated auctions, cameras attempt to gain ownership of objects less often and hence have lower resource consumption. Though this reduces communication, it requires that the utility of the camera from the object is almost zero before handing over, even though another camera might have had a better view earlier. This means, our ACTIVE approach refers to auctions being initiated at regular intervals while our PASSIVE approach initiates auctions only on-demand, dependent on the physical environment.

Figure 5.1 shows the overall system utility (i.e., the tracking performance of the network) and the communication overhead for the ACTIVE and PASSIVE algorithms in a camera network as depicted in Layout 9 (4.2i) with a single object moving from left to right. The spikes in utility occur when the object moves into the areas of high visibility in front of each camera. Due to the particular set-up of this scenario, there is little difference in utility between the two approaches. Only when the object moves into the FOV of the last camera on the right, the ACTIVE approach is able to hand over the object sooner, due to increased visibility. This also results in higher system utility. However, it is clear that the ACTIVE approach uses significantly more communication.

Since the ACTIVE approach yields the highest possible levels of communication and utility, the subsequently presented results in this thesis are normalised in each case by the results from the ACTIVE broadcast approach.

Based on this initial result, one can observe that the market-based approach presented does not require a vision graph in order to achieve effective object handover. However, by generating the vision graph during runtime, the camera network is able to achieve outcomes which balance more efficiently the trade-off between communication and tracking performance.



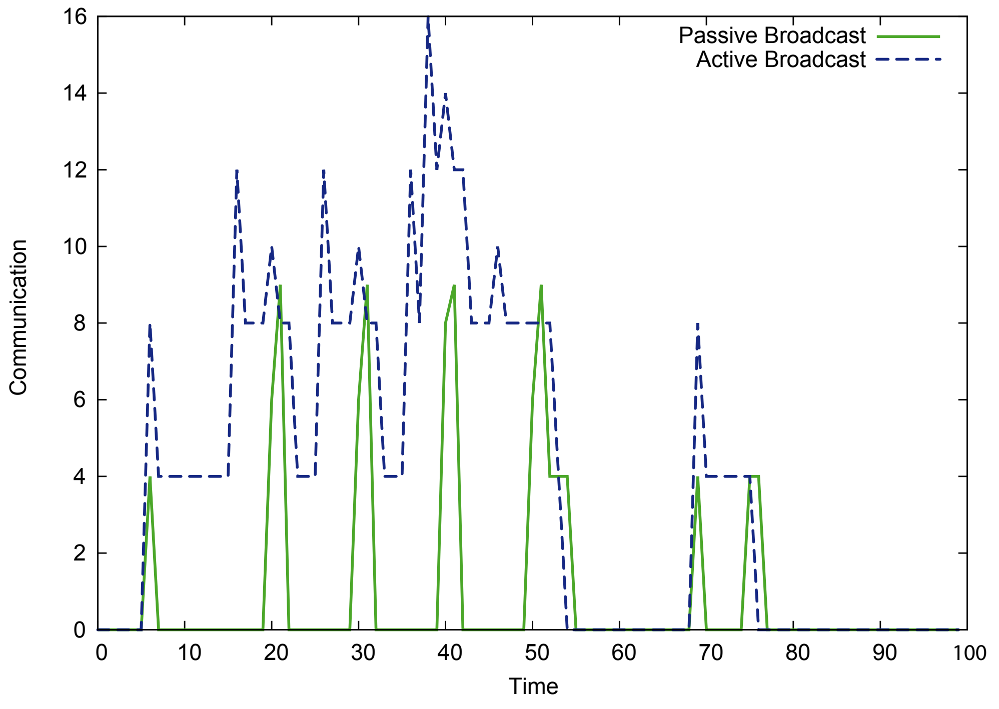


Figure 5.1: Network-wide utility (above) and communication usage (below) over time, during an experimental run with a single object in Layout 9. ACTIVE and PASSIVE BROADCAST algorithms are compared.

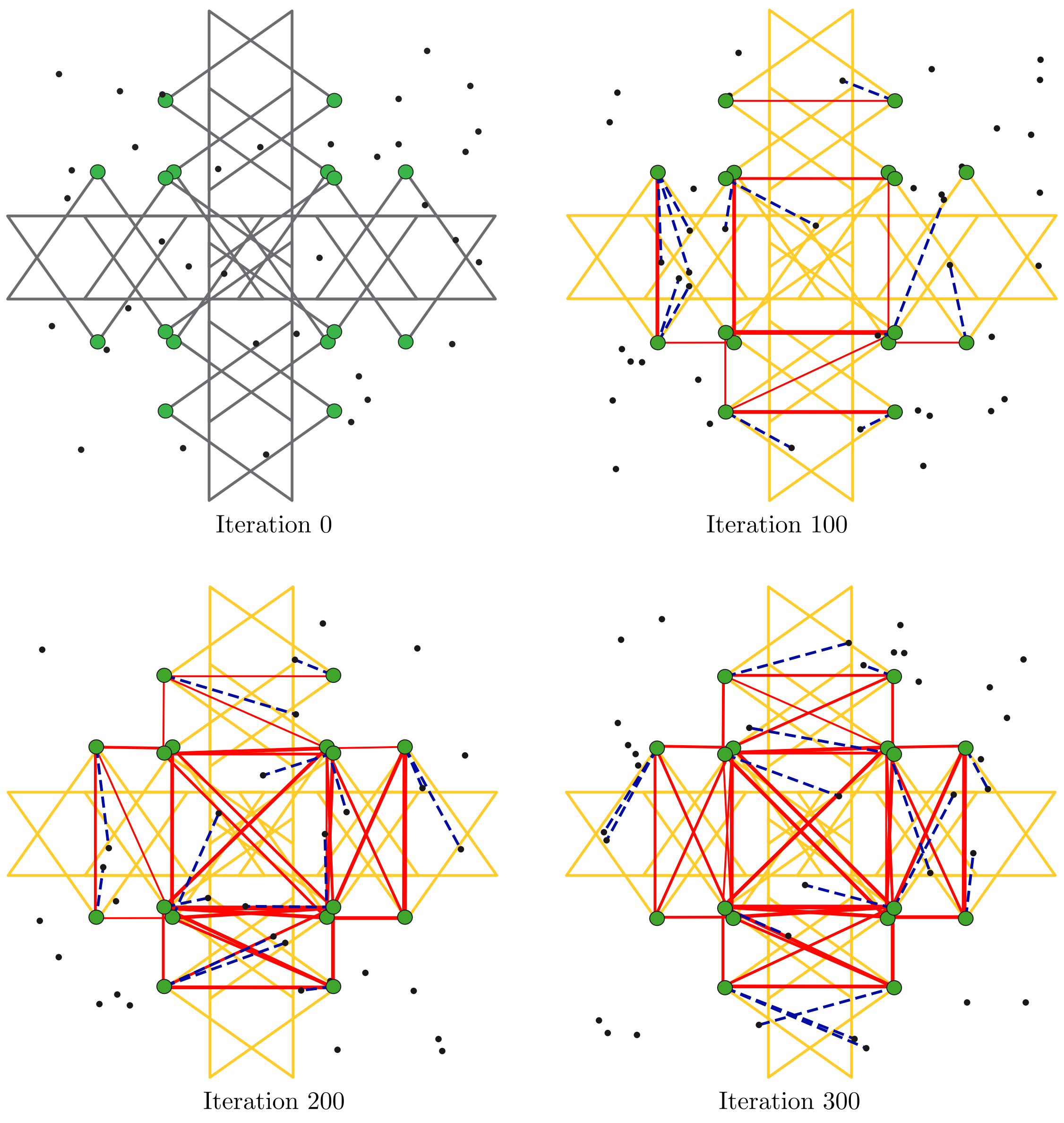


Figure 5.2: The vision graph is built up during runtime through trading interactions. Green dots represent a camera and grey triangles the corresponding FOV. A yellow triangle indicates a detected object within this FOV. Red lines indicate links in the vision graph; thickness indicates strength. Tracked objects are assigned to the respective camera via a dashed line.

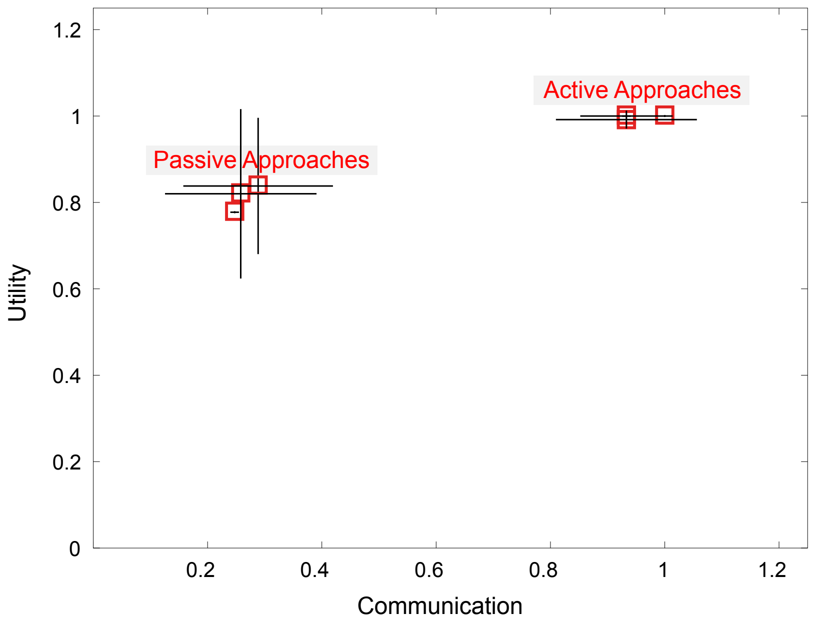
Figure 5.2 illustrates the pheromone-based approach to building the vision graph during runtime. The state of the vision graph is shown at four points through the simulation, from initialisation where no adjacency information is known. As the objects are traded between cameras, the links (indicated by red lines) are constructed. The thickness of these lines represent the amount of pheromones deployed between cameras and hence the strength of the link. Over time, pheromones of unused links evaporate, which corresponds to reduced link strength. By scheduling the communication intelligently and exploiting this vision graph, as described in Section 3.2.3 , the cameras might be able to reduce communication, while minimising the associated performance penalty.

The following experiments illustrate the effect of the multicast approaches SMOOTH and STEP, as described in Section 3.2.3 , when applied to both the ACTIVE and PASSIVE schedules. In all cases, $\rho = 0.005$, $\Delta = 1.0$ and $c_j = 1$ has been set based on heuristics for all cameras.

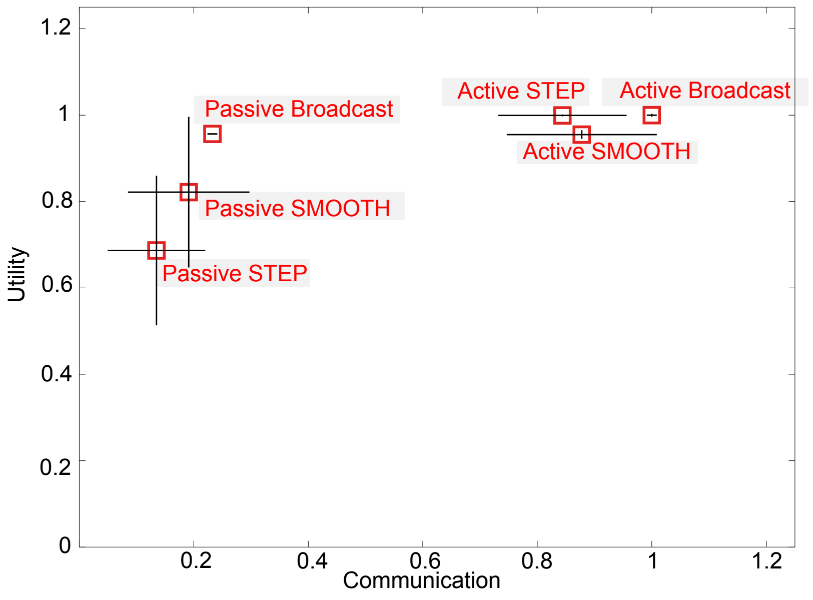
Figures 5.3, 5.4, and 5.5 show the overall performance of each of the six variants of the approach on all scenarios with one object in the environment, and also a random scenario - Scenario 17b - with 31 objects in the environment as depicted in Figure 4.2. Due to the stochastic nature of the trajectory of the object and the communication policies, mean and standard deviation are shown for each approach, calculated over 30 independent runs.

These results clearly show that the greatest difference between outcomes in the simpler scenarios is obtained when switching between ACTIVE and PASSIVE approaches. However, in the more complex scenarios (e.g., Scenario 11b or 12), the different schedules yield different outcomes in the trade-off between communication and tracking performance. A Pareto front emerges, allowing the operator to select between different handover algorithms based on how performance and communication are valued. As one can see, the results for scenarios with a lower number of cameras make it important to select the right auction strategy, while scenarios with a higher number of cameras have to consider the correct communication policy as well.

For example, for a camera network, with a complexity similar to Scenario 11b or 12, where high tracking performance is required and cameras do have a vast amount of resources, the ACTIVE BROADCAST or ACTIVE SMOOTH approaches might be most suitable. However, in a deployment where cameras have limited communication ability, some tracking performance can be traded off for communication efficiency by selecting perhaps PASSIVE BROADCAST or even PASSIVE STEP. In completely random scenarios, though, switching between ACTIVE and PASSIVE approaches has less impact on the performance than switching between communication schedules. This is mainly due to the fact that a high number of objects may create very strong links between certain cameras at an early

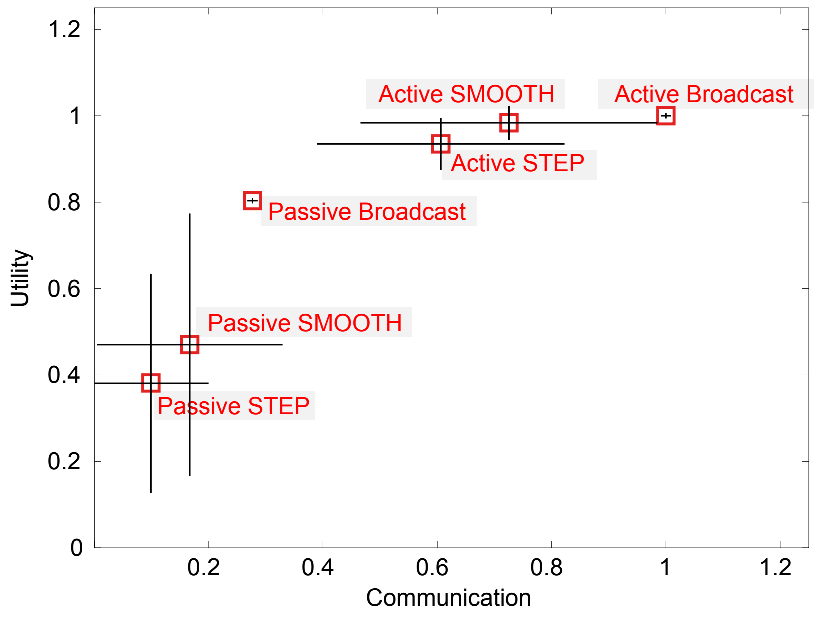


(a) Scenario 9a

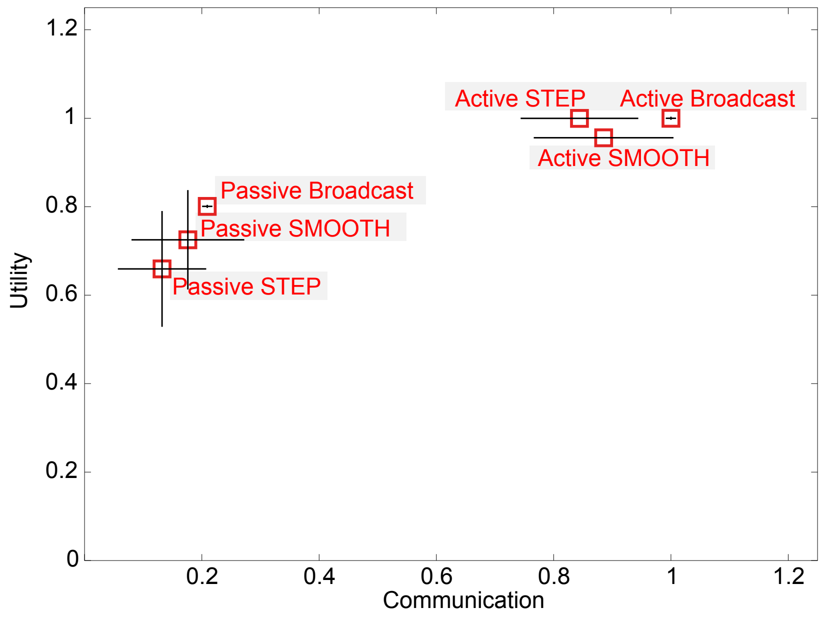


(b) Scenario 13

Figure 5.3: Performance (overall utility calculated across 1000 time steps) for Scenario 9a (above) and 13 (below) using our different algorithms. Both utility and communication values are normalised by those from the active broadcast algorithm. The trade-off between performance and communication is apparent. Due to the stochastic nature of the object paths and algorithms, the mean and standard deviation are shown, calculated over 30 independent runs of the simulation.

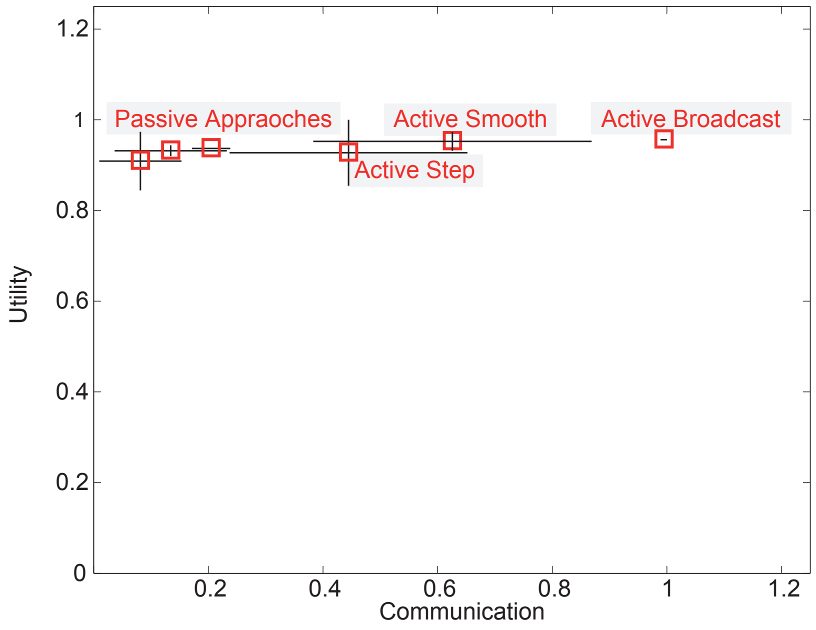


(a) Scenario 11b

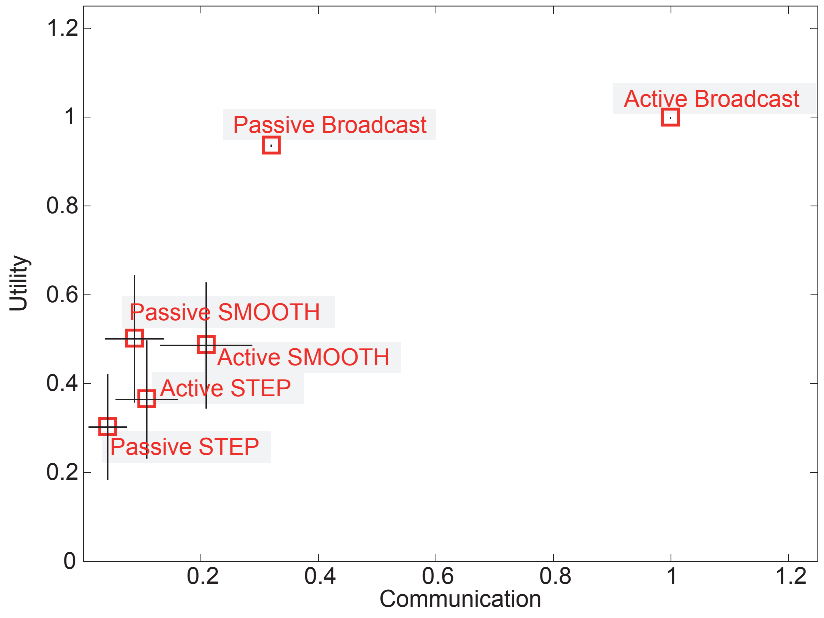


(b) Scenario 12

Figure 5.4: Performance (overall utility calculated across 1000 time steps) for Scenario 11b (above) and 12 (below) using our different algorithms. Both utility and communication values are normalised by those from the active broadcast algorithm. The trade-off between performance and communication is apparent. Due to the stochastic nature of the object paths and algorithms, the mean and standard deviation are shown, calculated over 30 independent runs of the simulation.



(a) Scenario 14



(b) Scenario 17

Figure 5.5: Performance (overall utility calculated across 1000 time steps) for Scenario 14 (above) and 17b (below) using our different algorithms. Both utility and communication values are normalised by those from the active broadcast algorithm. The trade-off between performance and communication is apparent. Due to the stochastic nature of the object paths and algorithms, the mean and standard deviation are shown, calculated over 30 independent runs of the simulation.

stage of the simulation. Hence other cameras are not considered anymore as communication partners at a later stage and therefore utility is lost. Again an operator could switch from broadcast communication to a less communication intensive policy as soon as the vision graph is developed.

In simple simulation scenarios, PASSIVE approaches achieved a communication reduction of around 75% for a 20% penalty in tracking performance when compared to the ACTIVE BROADCAST approach. Using our implementation in a real camera network, similar simple scenarios achieved a communication reduction around $40-45\%$ for only $10-15\%$ penalty in tracking performance. The more complex the scenarios got, the higher was the trade-off between ACTIVE and PASSIVE approaches and allowed reductions in communication by as much as 90%. Interestingly, in completely random scenarios the broadcast approaches showed superiority over our SMOOTH and STEP approaches where PASSIVE broadcast had about 65% more overall utility than PASSIVE STEP but only about 25% more communication effort when compared to the ACTIVE BROADCAST approach.

5.2 Evaluation of the Effects of Uncertainty

In this section, we show the robustness (i.e., the ability to deal with changes in the network autonomously) and scalability of a smart camera network when employing our improved socio-economic handover approach as discussed in Section 3.4. We conduct a series of experiments where we introduce uncertainties into our scenarios. A summary of the conducted scenarios and corresponding induced uncertainties is given in Table 4.5. For the new aspects of our extended camera handover algorithm, we selected $t_a = 3$ and $t_b = 6$ time steps based on heuristics as the duration for auctions and time-out in case of BROADCAST, SMOOTH and STEP communication policies, respectively.

Besides learning the vision graph online, we also performed each of those experiments where we define the vision graph *a priori*. In cases with predefined vision graphs, we use only the STATIC communication policy, where each camera only communicates with its own neighbours, to show the robustness of our socio-economic approach. When using this predefined vision graph and the STATIC communication policy, we do not use the ant-inspired approach to degrade the link strength or build up new links between cameras during runtime.

We conducted multiple experiments for the camera network layouts 15, 16, and 17 in Figure 4.2. In each simulation run, the total cumulative tracking performance, as given in Equation 3.2, was recorded across all cameras (the social welfare) as a measure of tracking performance. The total number of messages sent between cameras was also

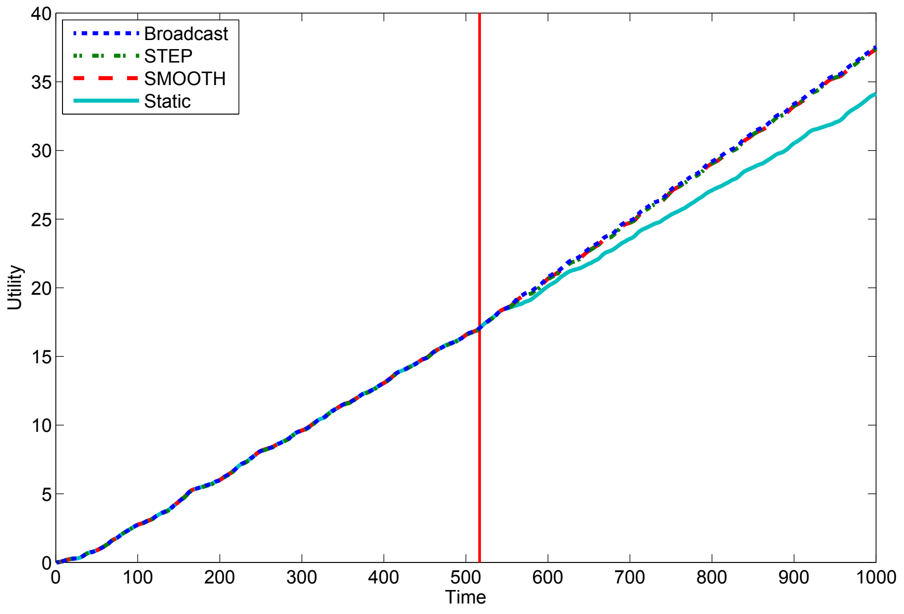


Figure 5.6: Cumulative sum of the entire network tracking performance over time for a typical simulation run of Scenario 16d comparing our ACTIVE socio-economic approaches with a static handover. The red vertical line indicates the time step when the event occurred. The simulation ran for 1000 time steps.

measured. We show the robustness of our approach by comparing our results with a STATIC communication policy based on an *a priori* known vision graph. For the network layouts 15 and 16 the *a priori* vision graphs have only been defined for cameras with an overlapping FOV but for the camera layout 17 non-overlapping FOVs were also considered as long as the FOVs of the cameras were adjacent to each other. We focus on highlighting the key results of our experiments.

The results of Scenario 15d are shown in Figure 5.6 where we added a new camera during runtime. The occurrence of the event is indicated with a red vertical line at time step 518. The increased accumulated utility using the ACTIVE approaches is apparent. Even though the camera was placed at a location which was already covered by another camera, we are able to observe a small improvement in tracking performance of the entire network compared to the STATIC approach. This is due to the better view of the object when the object passes through the FOV of the new camera for a short period of time and the dynamic incorporation of this new camera into the network.

Figure 5.7 shows results for Scenario 15c employing our ACTIVE approach. We changed the position of a single camera within the environment to show the ability of our approach

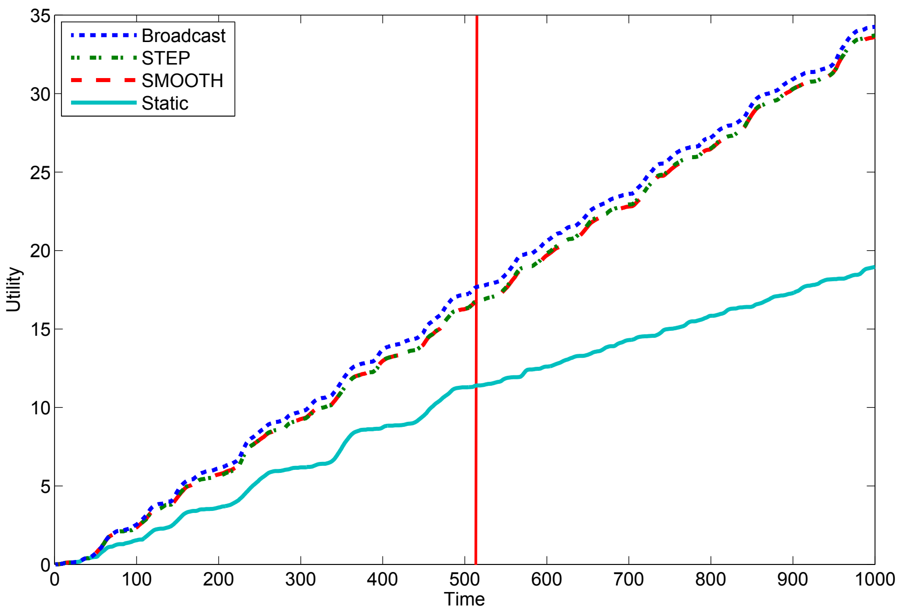


Figure 5.7: Cumulative sum of the entire network utility over time for a typical simulation run of Scenario 15c and using our PASSIVE approaches. The vertical line indicates the time step when the event occurred. The simulation ran for 1000 time steps. We changed the position of a single camera within the environment to show the ability of our approach to deal with changes of the extrinsic parameters of cameras.

to deal with changes of the camera's pose. The vertical line shows the time at which the event happened. It is apparent that the STATIC approach is not able to generate as much tracking performance when compared to our socio-economic approaches. This is due to the inability of the STATIC communication policy to adapt to changes in the network. While the STATIC communication policy loses overall utility, the SMOOTH and STEP policies are able to keep a high tracking performance after the event, indicating their robustness to change. Additionally, STATIC does not perform as well as any of the socio-economic approaches from the beginning due to the initially defined neighbourhood relations.

Figure 5.8 illustrates the results of Scenario 16b with a camera failure event, when PASSIVE approaches were used. The camera fails for the remaining duration of the simulation run. Here the drop of the accumulated tracking performance is noticeable for the static approach, while the socio-economic approaches are able to relearn the vision graph online. This allows the network to continue tracking the object and hence achieve higher network-wide tracking performance.

To compare the adaptivity and robustness of the different variants of the socio-economic

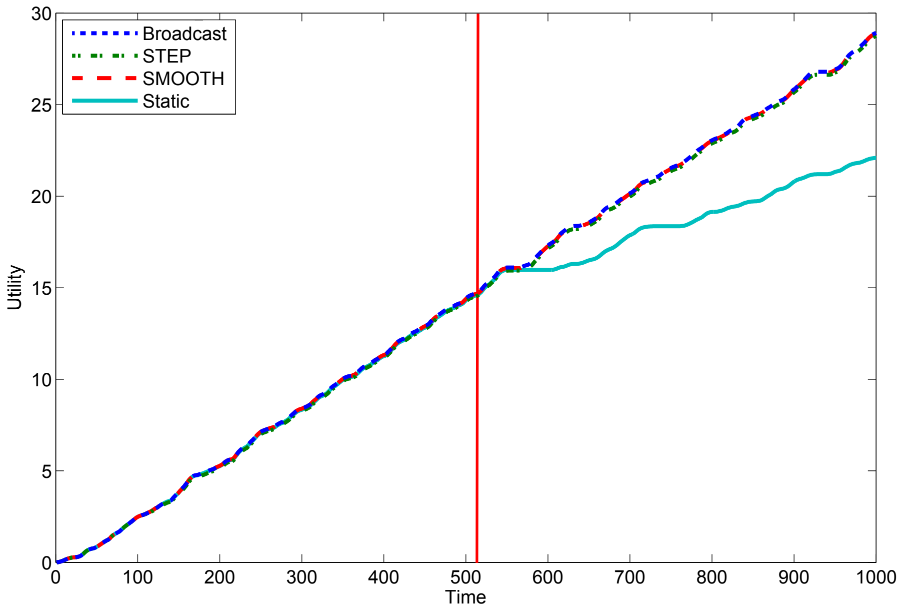
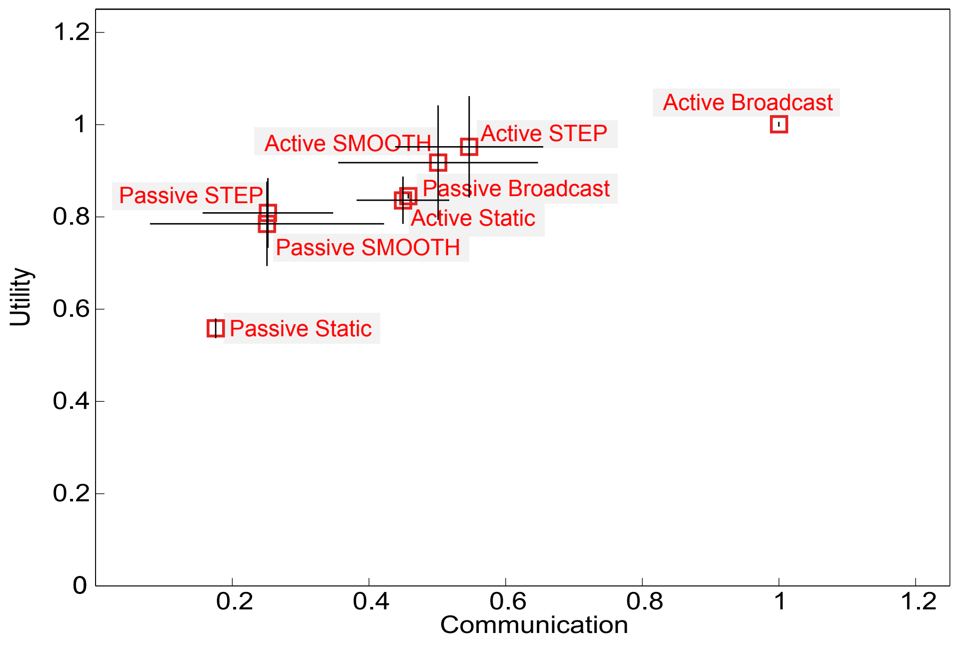


Figure 5.8: Cumulative sum of the entire network tracking performance over time for a typical simulation run of an experiment with Layout 16 with an error event (Scenario 16b) and using our ACTIVE approaches. The red vertical line indicates the time step when the event occurred. The simulation lasted for 1000 time steps.

approach with the static one, we accumulated the total social welfare not only for each single step but also for the entire simulation run. We plotted the results of the accumulated social welfare against the total accumulated number of exchanged messages for each experiment. The upper plot in Figure 5.9 shows the performance of the different communication strategies in a camera network with a topology as depicted in Layout 15 without any events. The lower illustration plots the results of the same experimental setup but this time having a change event after 518 time steps (Scenario 15c). In an experiment without any changes, the ACTIVE STATIC approach performs almost as well as the PASSIVE BROADCAST approach. In contrast, in experiments with uncertainties, the drop in performance when using the STATIC policy is apparent; the STATIC approaches are not as robust.

In Figure 5.10 we compare the performance of the different approaches when adding or removing cameras from the network during runtime. All three illustrations show results of a camera network with a topology as Layout 16. The top figure shows the results for an experiment where no events occur (Scenario 16a), while the middle figure shows the results where a camera has been added to the network during runtime (Scenario 16d). Even in



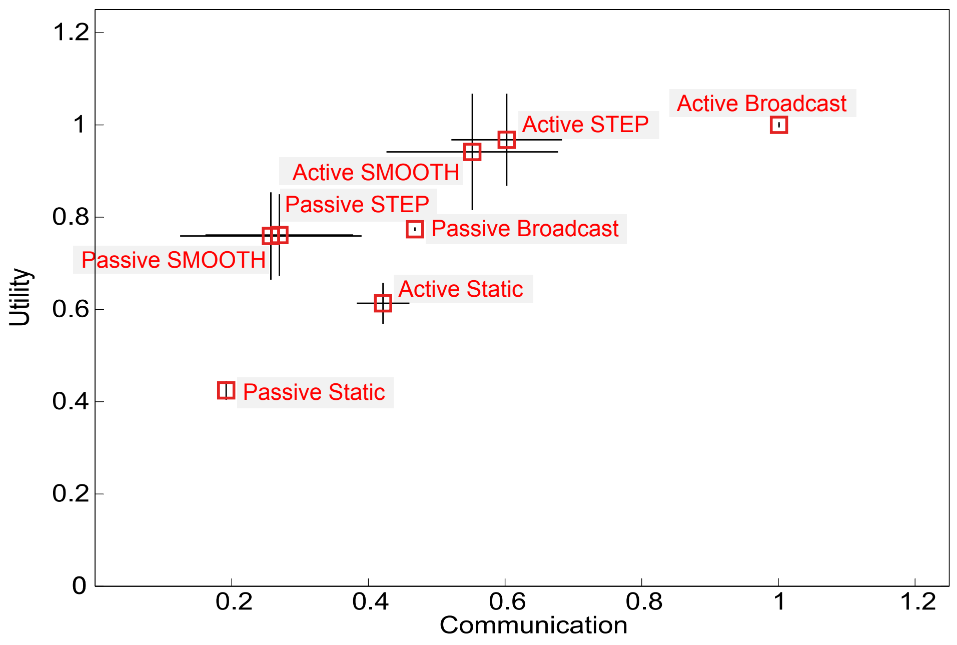


Figure 5.9: Overall utility of experiments with a network topology as depicted in Layout 15 calculated over 1000 time steps. Mean and standard deviation have been calculated over 30 runs. The upper graph shows the results for an experiment with no events while the lower graph shows the result for an experiment where the position of a camera was changed after 518 time steps.

the presence of a very simple uncertainty such as this, the decrease in performance of the STATIC policy are about 10% while all the dynamic socio-economic approaches maintain their high performance after the events. The bottom plot in Figure 5.10 illustrates the performance when a camera is removed from the network during runtime (Scenario 16b). While the utility drops for the STATIC communication policy by about 20%, the socio-economic approaches are able to maintain a high overall cumulative tracking performance, due to their ability to relearn the changed vision graph online.

Figure 5.11 shows the performance of a scenario where we used Layout 17. In the upper graph no events occur (Scenario 17a) but the lower figure illustrates an experiment with the same layout when multiple cameras fail after 518 time steps (Scenario 17b). In this scenario, the socio-economic approaches generally achieve a substantially higher tracking performance than the STATIC approaches, though they also require more communication. Due to the randomness of the scenarios, the drop in performance when using the STATIC policy is rather low, indicating that the robustness of the approaches varies with the scenario's properties.

5.3 Evaluation of Heterogeneous Assignment

In the previous section we showed the applicability of our novel socio-economic approach for distributed tracking where all cameras use the same strategy. In this section, we will consider to assign different strategies to the employed cameras in the network in order to improve the overall performance and reduce the communication overhead. After showing the benefits of heterogeneous configurations, we also discuss online learning techniques to allow each camera to identify the best fitting strategy autonomously. For our evaluation of heterogeneous assigned strategies as well as dynamically learnt strategies during runtime we considered the Scenarios 1–8, 9a, 10, and 11a.

We first consider Scenario 1, a baseline scenario with two cameras and four objects. Figure 5.12 shows the mean global performance, calculated over 30 independent runs. Each point represents the global outcome from one configuration over 1000 time steps, in terms of both metrics: its total tracking performance and the communication overhead within the entire network. Again, the achieved results are normalised by the outcomes obtained when using the ACTIVE BROADCAST approach.

By enforcing homogeneous configurations in the entire network, we have six possible deployment options. The outcomes from these homogeneous configurations are depicted as squares in Figure 5.12. In this scenario, despite the six possible homogeneous configurations, there are only two extreme observed outcomes in the objective space, one

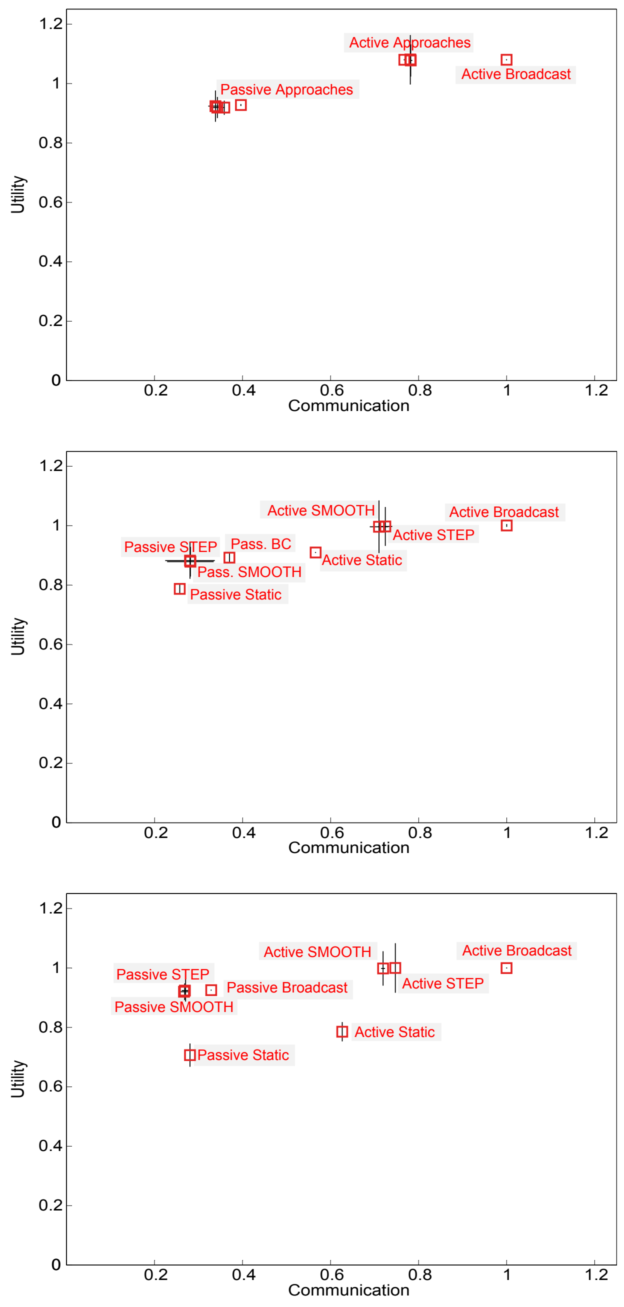
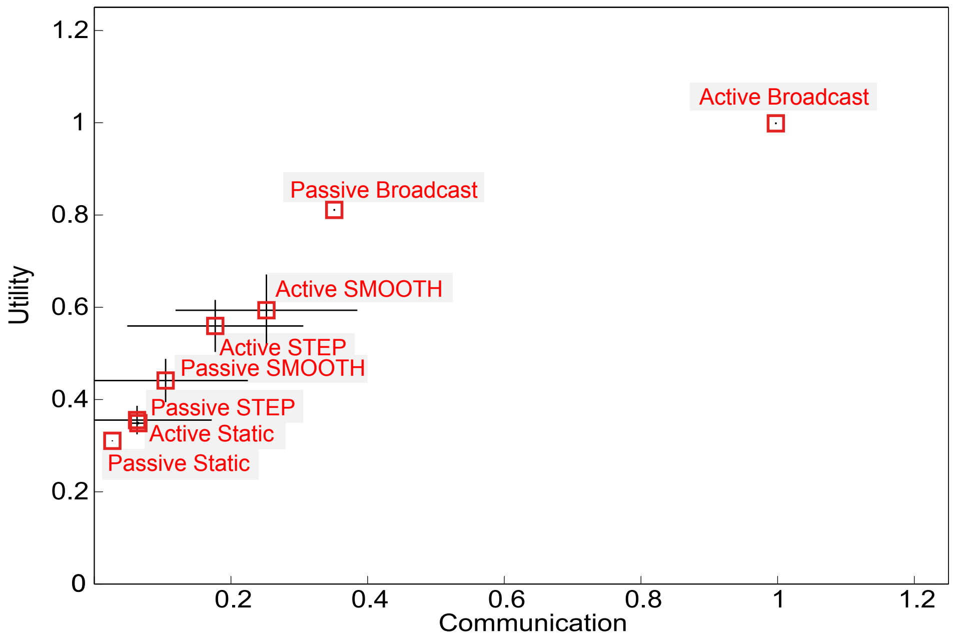


Figure 5.10: From top to bottom: overall utility of Scenario 16a, 16d, and 16b calculated over 1000 time steps. Communication and utility are shown on the x and y axes respectively, normalised by the maximum obtained values; ACTIVE BROADCAST always obtains a utility of 1 with a communication overhead of 1. Mean and standard deviation are shown, calculated over 30 runs. The upper graph shows the results for an experiment without any events. The middle graph shows the result for an experiment where a camera has been added to the scenario after 518 time steps. The lower plot shows the result for an experiment where a camera was removed after 518 time steps.



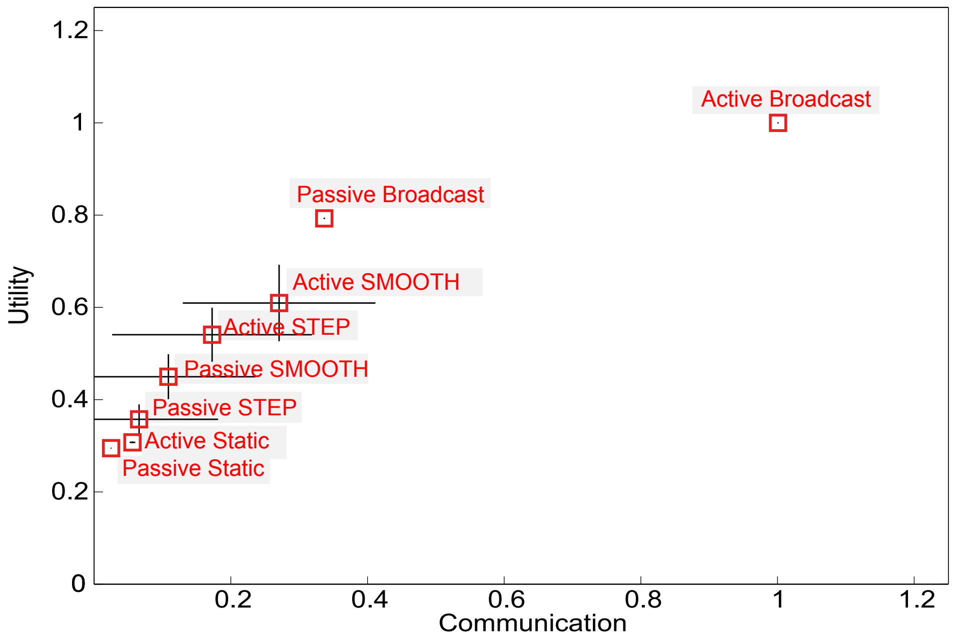


Figure 5.11: Overall tracking performance of Scenario 3 calculated over 1000 time steps. Mean and standard deviation have been calculated over 30 runs. The upper graph shows the results for an experiment with no events while the lower graph shows the result for a Experiment 7 where multiple cameras were removed after 518 time steps.

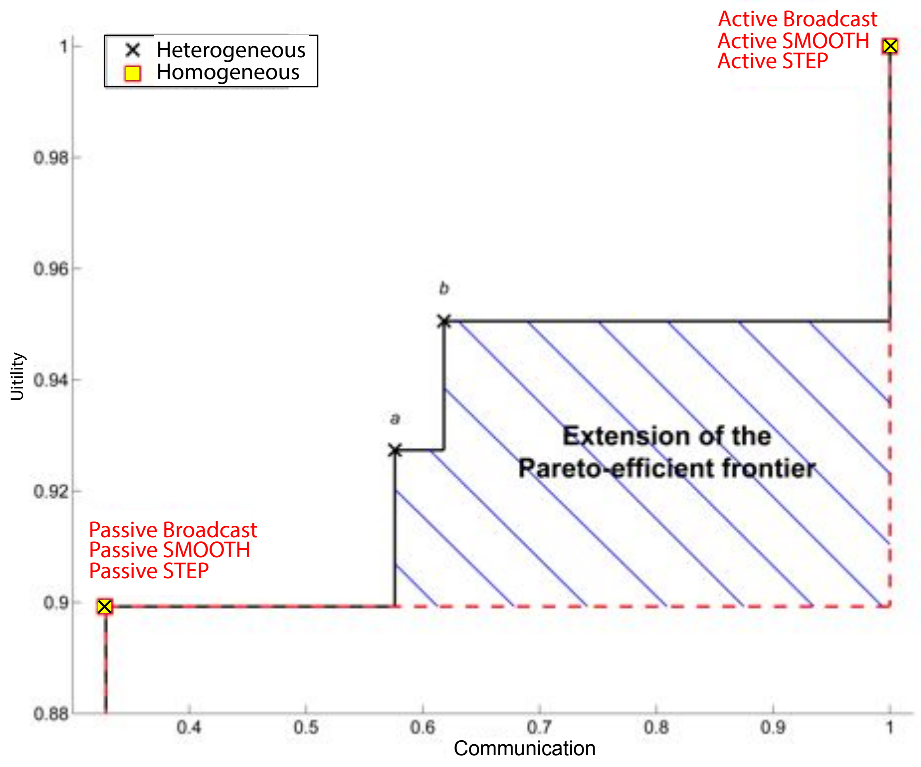


Figure 5.12: Results for a baseline scenario (Scenario 1) with two overlapping cameras. The original Pareto frontier when homogeneity is enforced is depicted by the dashed line. The solid line indicates the newly extended Pareto frontier when heterogeneous configurations are permitted.

favouring each objective. This is due to the very simplicity of the scenario, where some strategies give rise to the same communication behaviour as others; homogeneity does not permit any more balanced outcomes in this case. However, allowing the cameras to adopt different strategies from each other introduces new possibilities. When heterogeneous configurations are included, there are 36 possible deployment options. The heterogeneous configuration outcomes are depicted as crosses.

Outcomes *a* and *b* in Figure 5.12 clearly extend the Pareto efficient frontier, indicating new efficient configurations for tracking objects within the network. Additionally, both of these points lie on the newly extended Pareto frontier, since for each, no other outcome is better on both objectives. It is therefore clear from this example that heterogeneous configurations can lead to additional efficient outcomes. As we are interested in these extensions of the Pareto front, we focus on the achieved results and do not plot the entire objective space for these figures.

Furthermore, we consider more complex scenarios, consisting of larger numbers of

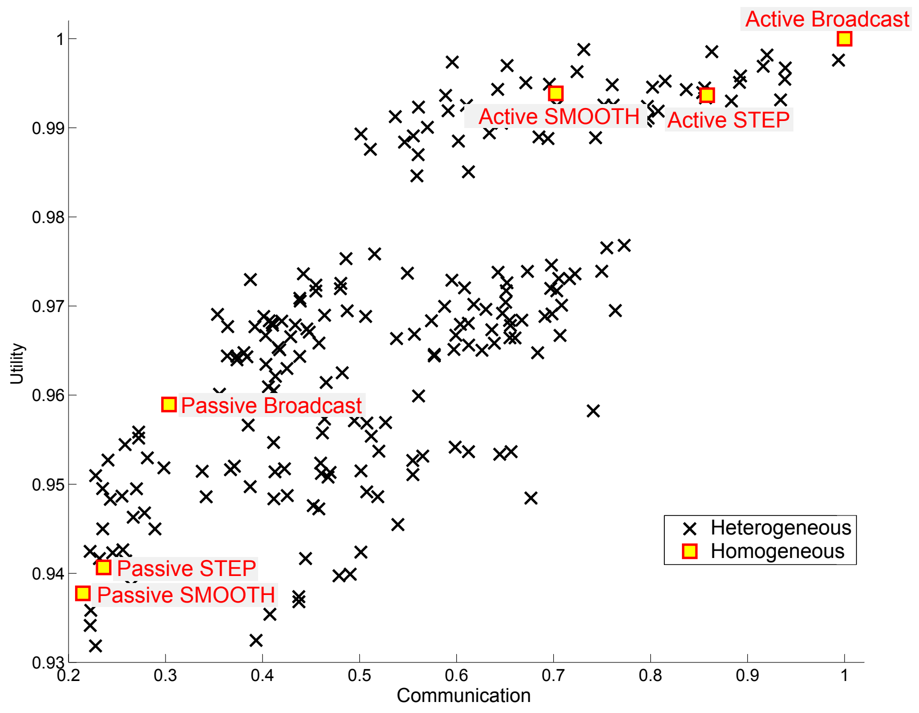


Figure 5.13: Performance for Scenario 4 showing homogeneous and heterogeneous assignment of strategies. The results have been normalised by the maximum value of the ACTIVE BROADCAST strategy and are averages over 30 runs with 1000 time steps each.

cameras in a range of layouts. The number of possible configurations depends on the number of cameras in a given scenario. If γ is the number of available strategies and n is the number of available cameras, there are γ^n possible network configurations. We currently consider $\gamma = 6$ different strategies. We evaluated all six homogeneous configurations in all scenarios, and all possible heterogeneous configurations in Scenario $1 - 8$ and 9b. An exhaustive evaluation of all heterogeneous configurations for scenarios with more than 7 cameras is computationally infeasible.

Figures 5.13 and 5.14 compare outcomes from heterogeneous and homogeneous configurations in Scenario 4 and 9b, respectively. In these more complex scenarios, heterogeneous configurations led to many more outcomes in the objective space. In each case, the extension of the Pareto efficient frontier caused by heterogeneous configurations is also apparent. However, it is also clear that the outcomes of many heterogeneous configurations are dominated, and many are strictly worse than the original outcomes from the homogeneous cases. Indeed, in all evaluated scenarios, when heterogeneous configurations of cameras are allowed, we observed system wide outcomes which both dominate and are dominated by outcomes from homogeneous configurations. In all cases, heterogeneity

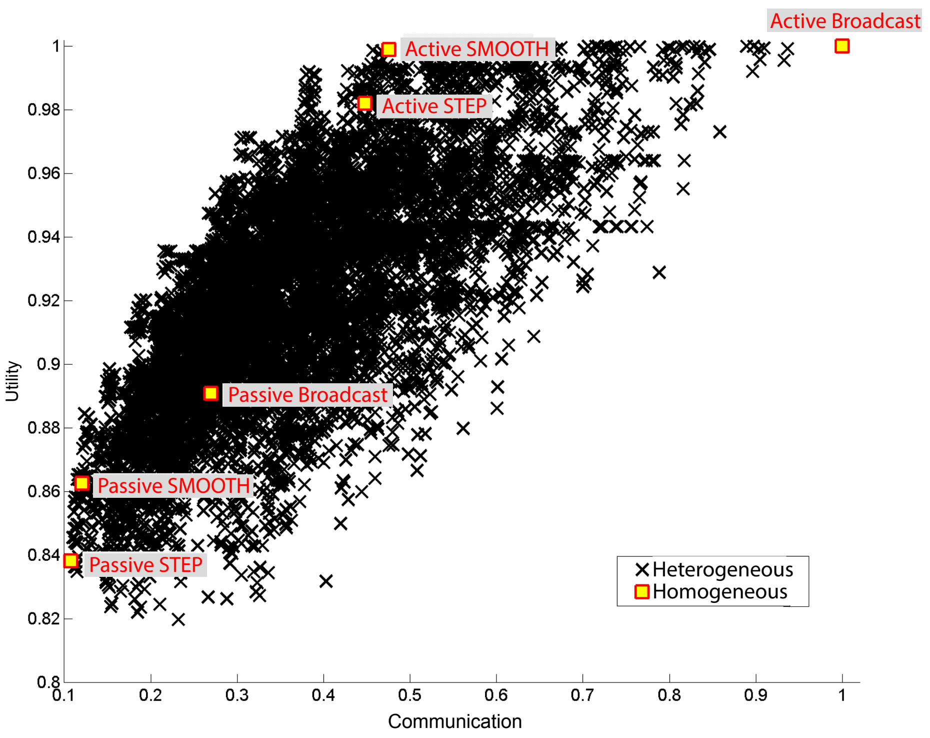


Figure 5.14: Performance for Scenario 9b showing homogeneous and heterogeneous assignment of strategies. The results have been normalised by the maximum value of the ACTIVE BROADCAST strategy and are averages over 30 runs with 1000 time steps each.

extended the Pareto efficient frontier.

5.4 Evaluation of Decentralised Online Learnt Configurations

While heterogeneous configurations can have a beneficial effect on the network wide results, an operator is not able to select the appropriate configurations without extensive study of the deployment area and the deployed camera setup. In Section 3.5.2, we proposed to employ multi-armed bandit problem solvers to learn the best strategy for each camera locally.

Figure 5.15 shows the results of Scenario 1, when configurations learnt using bandit solvers are compared with static homogeneous and heterogeneous configurations. For EPSILON-GREEDY, ϵ values of 0.1, 0.01 and 0.001 were investigated. In all scenarios, with 1000 time steps, $\epsilon = 0.1$ obtained the most Pareto efficient outcomes and is therefore used in all results in this chapter. Outcomes are shown for EPSILON-GREEDY, UCB1,

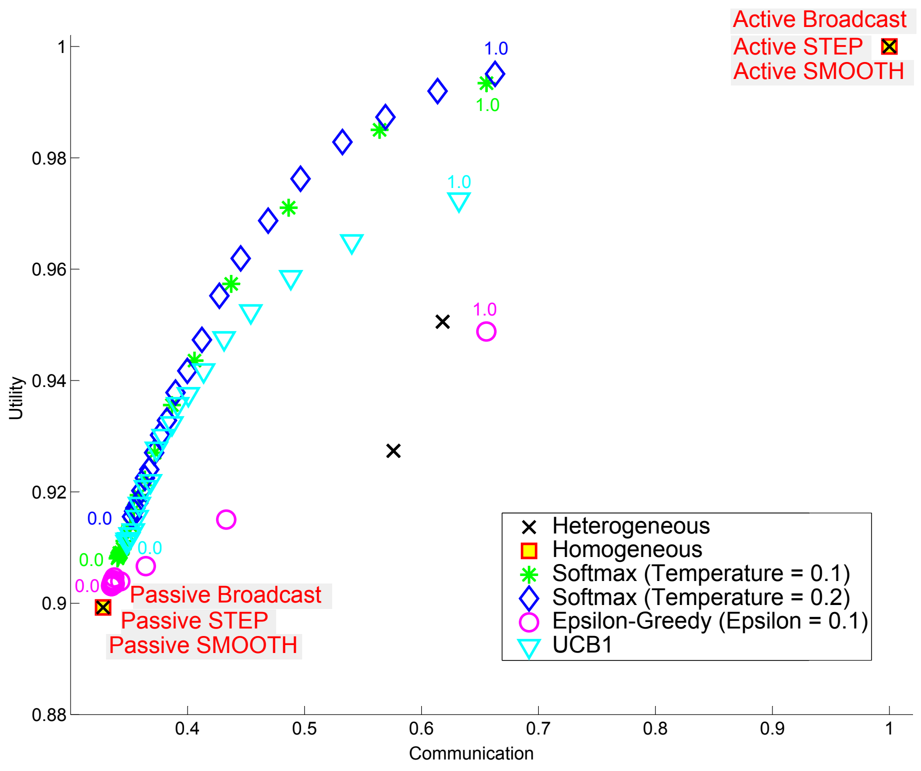


Figure 5.15: Performance for Scenario 1 showing homogeneous and heterogeneous assignment of strategies as well as assignments done by bandit solvers. The results have been normalised by the maximum value of the ACTIVE BROADCAST strategy and are averages over 30 runs with 1000 time steps each.

and SOFTMAX, the latter with temperature values 0.1 and 0.2. For each bandit solver, results are shown when α is varied between 0 and 1 in intervals of size 0.05 for the reward function given in Equation 3.10.

The results in Figure 5.15 clearly show that bandit solvers allow configurations to achieve a better performance when compared to the results of static homogeneous and heterogeneous configurations. Furthermore, the majority of these outcomes are highly Pareto efficient. Even though we presented the static heterogeneous configuration outcomes exhaustively, using bandit solvers enables the network to obtain system wide outcomes which extend the Pareto efficient frontier originally obtained in the static heterogeneous case.

Still, these results have not been normalised and are biased by their local observations. Therefore, we apply a normalisation by distribution on the camera level to the parameters of the reward function as presented in Section 3.5.3. Figure 5.16 shows the effects of this normalisation by distribution for Scenario 1, and can be compared with Figure 5.15. A less pronounced bias is still present with EPSILON-GREEDY and UCB1. This skewed

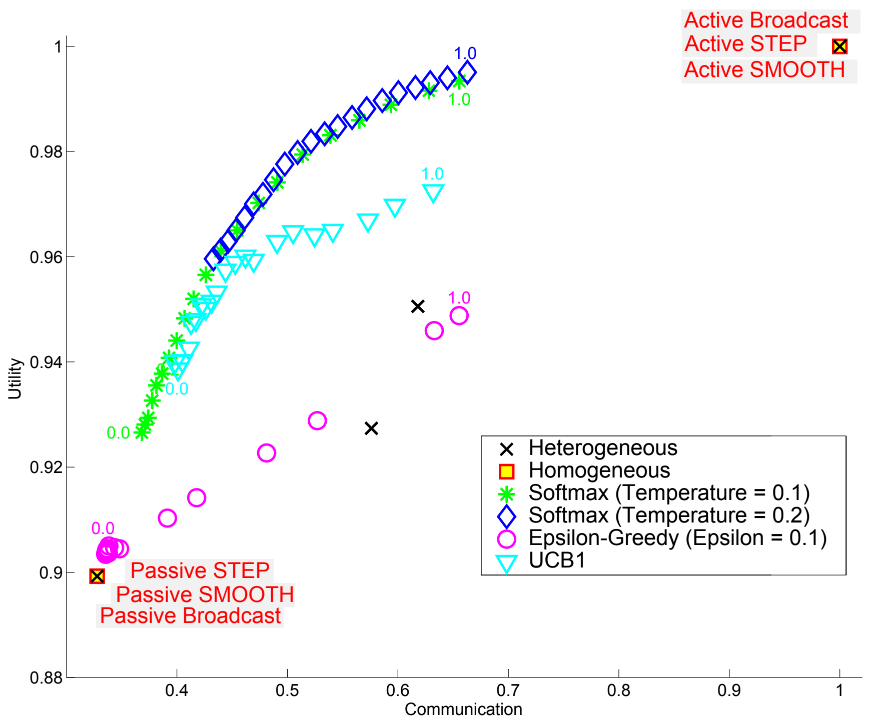


Figure 5.16: Performance for Scenario 1 showing static homogeneous and heterogeneous assignment of strategies as well as assignments done by bandit solvers. The results have been normalised by the maximum value of the ACTIVE BROADCAST strategy and are averages over 30 runs with 1000 time steps each. The bandit solvers' reward functions normalised the number of auction invitations by distribution at runtime.

distribution pervades all scenarios we evaluated, therefore we adopt this normalisation method in all subsequent experiments.

Figures 5.17, 5.18 and 5.19 show results for the more complex Scenarios 3, 5 and 11a respectively. In each of these more complex scenarios, bandit solvers were able to obtain outcomes which extend the Pareto efficient frontier of the evaluated static configurations. This is particularly true for SOFTMAX (with both temperature values) and UCB1, all of which obtain a range of highly Pareto efficient outcomes. The spread of outcomes can be observed to vary depending on the particular scenario and the choice of bandit solver employed. The bias associated with EPSILON-GREEDY, and to a lesser extent UCB1 remains; outcomes from the other bandit solvers are evenly spread as α varies. Of all the bandit solvers, SOFTMAX typically obtains the best spread across the frontier.

Additionally, it can be observed that bandit solvers never reach either extreme of the objective space, but rather tend towards the middle of the frontier. This behaviour is due

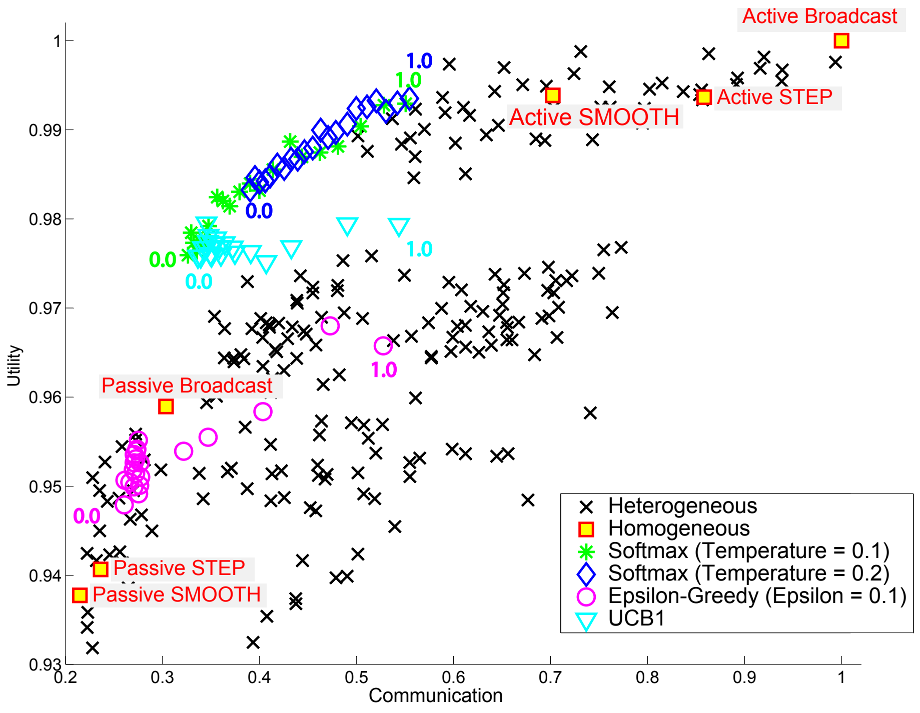


Figure 5.17: Performance for Scenario 3 showing homogeneous and heterogeneous assignment of strategies as well as assignments done by bandit solvers. The results have been normalised by the maximum value of the ACTIVE BROADCAST strategy and are averages over 30 runs with 1000 time steps each. The bandit solvers' reward functions normalised the number of auction invitations by distribution at runtime.

to the selection of the α value of the reward function. While the α value allows to define a preference between tracking performance and communication overhead, the selected values do not prefer either extreme. Van Moffaert et al. [86] present an approach on learning weights in multiple simulation runs to cover both extremes as well as achieving an even spread along the Pareto frontier.

To compare Pareto efficient frontiers, achieved by our different configurations, we compute the hypervolume [91] under each frontier, given a reference point. Calculating the hypervolume in a two-dimensional objective space comes down to computing the area under the frontier. The reference point can be specified as the vector of worst case values. Thus, a tracking confidence value of 0.0, and a number of auction invitations value of 1.0 specifies our reference point. The greater the hypervolume of a Pareto frontier, the more efficient it is.

If there is a new result a which is not dominated by a single outcome defining the current frontier, a extends the existing frontier and becomes part of a new frontier. This

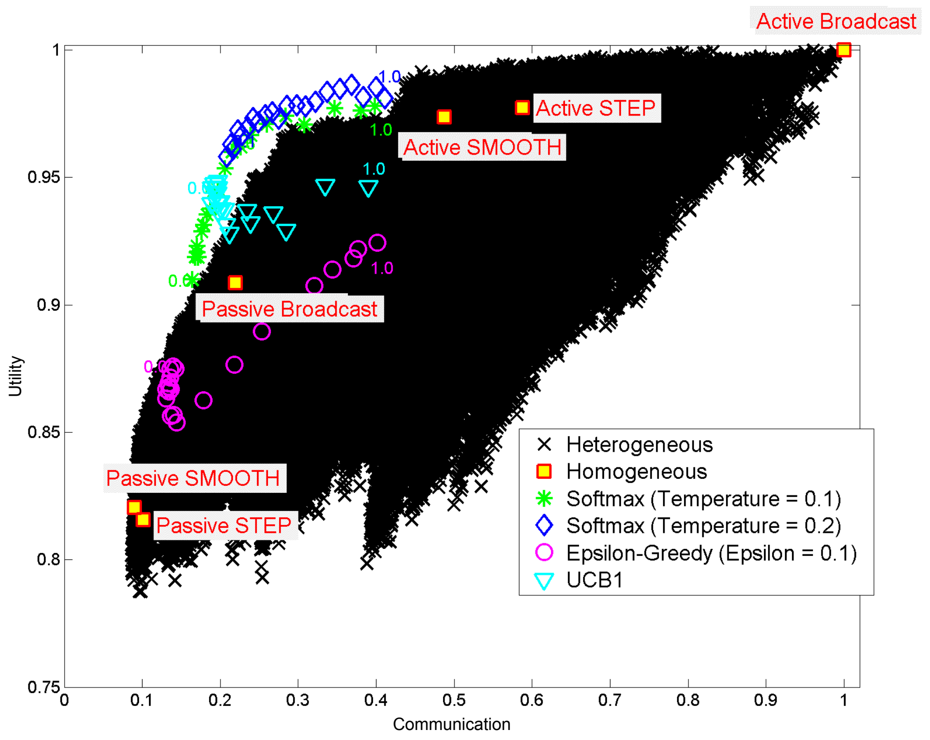


Figure 5.18: Performance for Scenario 5 showing homogeneous and heterogeneous assignment of strategies as well as assignments done by bandit solvers. The results have been normalised by the maximum value of the ACTIVE BROADCAST strategy and are averages over 30 runs with 1000 time steps each. The bandit solvers' reward functions normalised the number of auction invitations by distribution at runtime.

new front has then a larger hypervolume than the previous front. We exemplify such an extension of the frontier in Figure 5.12. If we strictly consider only outcomes from the static homogeneous configurations, we call the Pareto efficient frontier h. The Pareto efficient frontier considering all the outcomes from both static homogeneous and static heterogeneous configurations gives us a different frontier we call *h-he*. Outcomes based on the results of bandit solver and in combination with static homogeneous configurations are entitled *h-eg*, *h-sm*, or *h-ucb*, depending on the considered bandit solver being EPSILON-GREEDY, SOFTMAX, or UCB1 respectively. Similarly we call the frontiers *h-he-eg*, *h-he-sm*, or *h-he-ucb* when combining results from bandit solvers with static homogeneous configuration outcomes and static heterogeneous configuration outcomes. Table 5.1 shows the medians (across 30 independent runs) of the hypervolumes of the aforementioned frontiers for the scenarios considered in this section. We performed a Wilcoxon rank sum test [92] with a 95% confidence level to asses the statistical significance. This significance test allows a pairwise comparison of approaches and shows that one approach has a larger

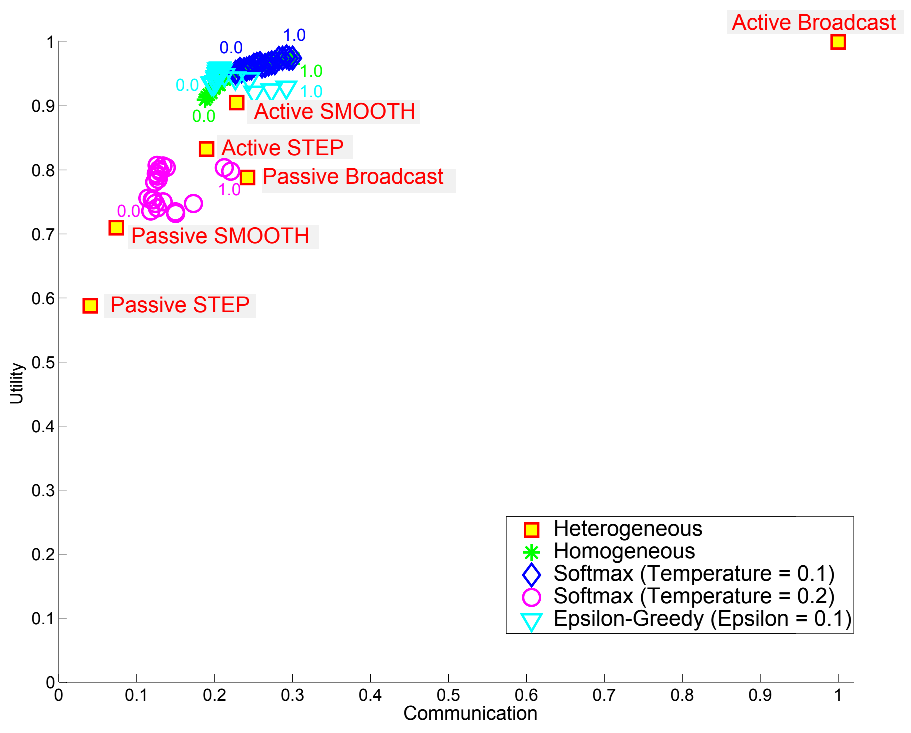


Figure 5.19: Performance for Scenario 11a showing homogeneous of strategies as well as assignments done by bandit solvers. The results have been normalised by the maximum value of the ACTIVE BROADCAST strategy and are averages over 30 runs with 1000 time steps each. The bandit solvers' reward functions normalised the number of auction invitations by distribution at runtime.

area underneath the resulting Pareto front than another approach in at least 95% of the results. In the table, this is denoted in the extensions with respect to *h* and *h-he*. "*" indicates a significant increase with respect to the static front *h*. "†" denotes a significant increase with respect to the static front *h-he*. "–" denotes tests that were not performed due to computational infeasibility of evaluating all heterogeneous configurations.

It is evident that heterogeneity of marketing strategies, which results in outcomes contained in the frontiers *h-he*, extend the homogeneous frontiers *h* in any given scenario. The frontiers *h-eg*, *h-sm*, and *h-ucb*, which contain outcomes from bandit solvers, often extend the frontier arising from static homogeneous configuration outcomes *h* significantly. Moreover, the frontiers *h-he-eg*, *h-he-sm*, and *h-he-ucb*, apart from extending the frontier *h*, sometimes further extend the frontier that includes both static homogeneous and static heterogeneous configuration outcomes *h-he*. Thus, decentralised online learning, based on bandit solvers, leads to self-organisation of the network, allowing achievements of global outcomes that are more Pareto efficient than those from static configurations. Extensions of the Pareto efficient frontier tell us that dynamic configurations induced by online learning allow the network to reach favourable parts of the objective space, which are inaccessible in the static case. Furthermore, we analysed the network-wide performance of autonomously learnt configurations and its efficiency compared to statically assigned approaches in a real world setting using Scenario R13. The results of this real world scenario are discussed in Section 6.3. The efficiency of this real world scenario is also given in the Table 5.1.

	Static fronts		Fronts with learnt outcomes					
Scenario ID \ Pareto front	*h*	*h-he*	*h-eg*	*h-sm*	*h-ucb*	*h-he-eg*	*h-he-sm*	*h-he-ucb*
1	0.6059	0.6271 ∗	0.6337 ∗	0.6612 ∗†	0.6516 ∗†	0.6443 ∗†	0.6612 ∗†	0.6532 ∗†
2	0.7926	0.8255 ∗	0.8107 ∗	0.8080 ∗	0.8061 ∗	0.8320 ∗†	0.8338 ∗†	0.8313 ∗†
3	0.7129	0.7608 ∗	0.7457 ∗	0.7607 ∗	0.7558 ∗	0.7808 ∗†	0.7841 ∗†	0.7830 ∗†
4	0.7658	0.7824 ∗	0.7834 ∗	0.7864 ∗	0.7839 ∗	0.8012 ∗†	0.7993 ∗†	0.7993 ∗†
5	0.8552	–	0.8702 ∗	0.8895 ∗	0.8778 ∗	–	–	–
6	0.8957	–	0.9095 ∗	0.9097 ∗	0.9094 ∗	–	–	–
7	0.8191	–	0.8420 ∗	0.8450 ∗	0.8446 ∗	–	–	–
8	0.8258	–	0.8455 ∗	0.8608 ∗	0.8503 ∗	–	–	–
9b	0.8487	0.8800 ∗	0.8581 ∗	0.8762 ∗	0.8706 ∗	0.8810 ∗	0.8840 ∗†	0.8823 ∗
10	0.9219	–	0.9229	0.9243	0.9244	–	–	–
11a	0.8384	–	0.8490	0.8954 ∗	0.8832 ∗	–	–	–
R13	0.8998	0.9153 ∗	0.9053 ∗	0.9002	0.9039 ∗	0.9153 ∗	0.9153 ∗	0.9153 ∗

Table 5.1: Medians over 30 independent runs of the hypervolume of the Pareto front resulting from various configurations: static homogeneous (*h*), both static homogeneous and static heterogeneous (*h-he*), both static homogeneous and learnt (*h-eg*, *h-sm*, *h-ucb*), and static homogeneous, static heterogeneous and learnt (*h-he-eg*, *h-he-sm*, *h-he-ucb*). "∗" denotes a significant increase w.r.t. static front *h*. "†" denotes a significant increase w.r.t. static front *h-he*. "–" denotes Wilcoxon rank sum tests that were not performed due to computational infeasibility of evaluating all heterogeneous configurations.

5.5 Further Discussion

In this chapter, we presented the results of our socio-economic approach to identify spatial relations among FOVs in smart camera networks. We were able to show that the different presented strategies prefer one out of two opposing metrics, maximising tracking performance or minimising communication overhead. This leads to different outcomes on the Pareto front. In multiple experiments we have explored this trade-off between communication effort and tracking performance. Hence, a network operator can choose among different performance/communication settings.

Additionally, we demonstrated the adaptivity and robustness in networks of distributed smart cameras, when cameras employ an improved version of our socio-economic algorithm. Our approach is able to keep a high tracking performance of objects in the camera network in the presence of uncertainties when compared to a static approach using a vision graph that does not change over time. Furthermore, it improves the adaptivity by enabling the network to take advantage of new cameras, which may be added during runtime. Similarly, it improves the robustness by enabling the network to relearn the vision graph in case of failing cameras. We showed that different variants of our approach are able to retain or even increase their utility after events occur. For example, when comparing cumulative network performance with overall communication overhead, the technique maintains a steady performance despite the presence of uncertainties. Conversely, the utility in all scenarios using a STATIC communication policy dropped by 10-20% depending on the event.

As the presented approach allows a network operator to chose one option for all cameras among a selection of settings, we have also studied heterogeneous assignment of approaches as well as the self-organising behaviour of our smart camera networks. We showed that applying heterogeneous configurations of marketing strategies in the network can outperform homogeneous configurations in terms of network-wide tracking performance as well as the communication overhead, which is also a proxy for processing overhead. We presented an increase of the Pareto efficiency in a range of heterogeneous configured scenarios compared to the homogeneous equivalents. Since scenarios are not known in advance, and the performance of configurations varies significantly between scenarios, the exact configurations which lead to Pareto efficiency in a given deployment will also not be known in advance. Therefore, we showed that online learning techniques, in the form of multi-armed bandit solving algorithms, on the individual camera level are able to find well performing heterogeneous configurations.

These outcomes extended the system-wide Pareto efficient frontier when compared to the homogeneous case. Furthermore, in many cases, the dynamic behaviour resulting from online learning led to outcomes which extended the Pareto frontier even when compared to the best possible outcomes from static heterogeneous configurations.

The results in this chapter show the complexity of trading behaviour, vision graph learning and bandit solvers and the impact of those complexities on local reward functions to achieve more Pareto efficient global outcomes. Nevertheless, the presented results are highly encouraging.

Although this work is based on camera networks, the principles behind both heterogeneous configuration and decentralised online learning are not limited to camera networks. Indeed, the benefits observed due to increased configuration possibilities in the heterogeneous case should be applicable to other networked systems. Similarly, the technique used for decentralised online learning of configurations should also be more widely applicable. Evaluation of these principles in other systems is an area for future work.

Sixth Chapter

Evaluation in Real-world Scenarios

To evaluate our approach under realistic conditions, we used two different approaches. First, we implemented our novel multi-camera tracking approach in a network of five distributed and autonomous smart cameras. Second, we pre-processed real world video data and extracted location and tracking confidence of all objects of interest. Using this information in our simulator allows us, on one hand to process information from real world video feeds as well as the errors induced by the tracking algorithms, on the other hand, to replicate experiments quickly and reliably. In this real world study, feature-based trackers are responsible for tracking the object of interest within the FOV of the camera. Each camera runs the autonomous camera control algorithm, as discussed in Chapter 3, to acquire and hand over tracking responsibilities as well as update the locally stored vision graph information.

The rest of this chapter is structured as follows: In the next section, implementation details of the real world deployment as well as the employed tracking algorithms are given. Afterwards, in Section 6.2, the utilised hardware is described. Section 6.3 explores the different scenarios used for evaluation. Finally, Section 6.4 discusses the results found and concludes this chapter. The results presented in this chapter have also been published in [1], [33], [52], and [53].

6.1 Implementation Details

There are a number of differences between the simulation environment and the real camera system. This requires some refinements to the used approach, when implementing the techniques designed on the simulation platform in the real network. These implementation details are concerned with (i) the tracking algorithm and the computation of the confidence, (ii) the required time for deriving the handover decision and (iii) the triggering of a handover for the passive algorithm. We describe these implementation aspects in the following paragraphs.

We use various frame-to-frame feature-based matching techniques as single camera tracking algorithms which exploit either SIFT [58] or SURF [7] features, or colour histograms combined with background subtraction [19]. While this thesis does focus on tracking algorithms, any computer vision algorithm can be used which is capable of re-identifying objects in different images. We selected the feature- and colour histogram-based implementations due to their availability. To perform tracking in a single camera, the object must first be detected in an single frame using either the SIFT, SURF or colour feature model respectively, and then if successful, be re-identified on a frame-by-frame basis. Thus, to advertise an object to a neighbouring camera (cp. step 1(a)) in

Algorithm 1, we need to transfer the object model. One should note, that we do not use the SIFT, SURF or colour features to build our vision graph. We rather use these features to track and re-identify objects or individuals in other cameras and therefore, do not require overlapping FOVs.

In our implementations using the SIFT or SURF features, we define the matching rate of the corresponding features as confidence for object detection and tracking. The model of object j is given by the set of features F_{m_j} of a specified region of interest[1]. To identify an object within the FOV, the features of the current frame img are compared to F_{m_j}, using the feature matching algorithm. Thus, we define the confidence c_j of detecting or tracking object j by

$$c_j = \frac{|F_{m_j}|}{|Match(F_{img}, F_{m_j})|} \tag{6.1}$$

where $Match(F_{img}, F_{m_j})$ represents the set of matched features. Note that in the current implementation, we set the visibility parameter of an object j as $v_j = 1$. Thus, currently we do not consider distance and orientation information for the utility. When using colour histograms as the preferred tracking method, the Bhattacharyya coefficient was used to calculate the similarity between the histogram of the model and the histogram found within the current FOV of the camera. The colour histogram tracker was implemented by our partner in the EPiCS project, the Austrian Institute of Technology (AIT).

Due to communication delays in the network, the object advertisement, and the subsequent object detection procedure at neighbouring cameras, the handover process is no longer instantaneous. For example, the SIFT- or SURF-based models of an object typically have a size of around 100 kB. Thus, the transfer of such a model, from the advertisement until re-detection, requires approximately 300 ms. To allow every camera enough time to participate in an auction, auctions have a certain duration t_a. This so called auction window starts with the arrival of the first bid. As soon as the time of the auction window has elapsed, the winner of the auction is determined and newly arriving bids are discarded. For the update of the vision graph, we defined the "sampling time" as 100 ms. Thus, the strength τ_i of each edge is decreased by the amount ρ every 100 ms. As defined in Equation 3.3 a link strength τ_{ix} is increased by Δ only if a trade between the two cameras has occurred. In our implementation, we set $\Delta = 1$ and $\rho = 0.005$.

For the passive approach, we defined a boundary margin in the FOV to trigger the advertisement of an object (cp. Figure 6.3). Thus, when an object is detected for three consecutive time steps within the margin we consider the object to be about to leave the

[1]Currently, this region of interest is defined by the user.

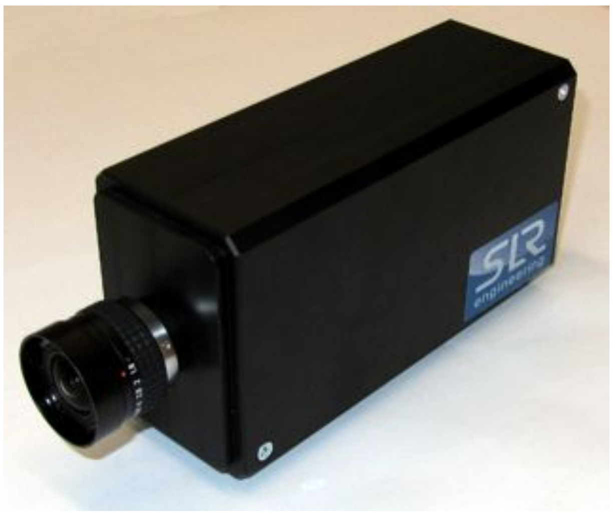


Figure 6.1: The SLR smart camera platform. A custom-built camera running Linux on a 1.6GHz Intel Atom processor with 2 GB memory and 100 MBit Ethernet interface. The image sensor is a CCD colour image sensor with a native resolution of 1280 × 1024 pixels.

FOV of the camera and start with the handover process.

6.2 Camera Platforms

Our camera networks are composed of two different types of camera platforms. The two camera platforms are (i) custom-built smart cameras from SLR Engineering and (ii) PandaBoard based platforms in combination with off-the-shelf web cams. The custom-built SLR cameras are equipped with an Intel Atom processor, running at 1.6 GHz and an 100 MBit Ethernet interface (cp. Figure 6.1). These smart cameras include a CCD colour image sensor with a native resolution of 1280×1024 pixels. The processor runs a standard Linux distribution that provides flexible integration of additional hardware modules and external software libraries. In contrast, the PandaBoard platforms are equipped with Logitech HD Pro C920 web cams. The PandaBoard platforms are based on Texas Instruments OMAP 4430 system-on-a-chip, which features a dual core ARM Cortex-A9 CPU running at 1.0 GHz and uses a 802.11 b/g/n wireless connection to the network. The connected Logitech C920 web cam operates with a native resolution of 3MP (cp. Figure 6.2). While the SLR Engineering cameras communicate via Ethernet, the PandaBoard based platforms facilitate a wireless network connection. We used this network to record multiple scenes for evaluation. Details on recorded video data and related scenarios can be found in the following section.

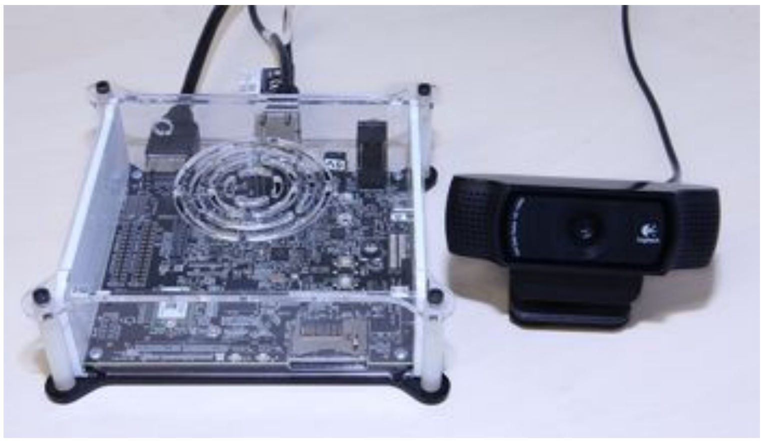


Figure 6.2: The PandaBoard based smart camera platform running Linux on a 1.0 GHz ARM processor with 802.11 b/g/n wireless connection interface. A Logitech C920 is used to acquire images with a native resolution of 3MP.

6.3 Evaluated Scenarios

For evaluation purposes we used two different settings. The first setting consists of two scenarios similar to Layout 9 and Layout 13 in Section 4.5, shown in Figure 4.2, using SLR cameras only. We refer to these scenarios as Scenario R1 and Scenario R2. In these experiments we performed tracking of a single person within a network of five cameras. To achieve reproducible results, we recorded videos of two different scenarios on each camera. To allow for reproducibility in our evaluation process, we used the recorded videos as input for the cameras and did not use the live video feed. Scenario R1 was recorded with overlapping cameras and lasts about 70 seconds. Hence, the person was visible by at least one camera during the entire test sequence. Scenario R2 represents a non-overlapping camera setup. Here the person followed a different path; as a result, the person was not visible by any camera for a certain period. Scenario R2 lasts about 130 seconds. The evaluation and generation of the vision graph of Scenario R1 and Scenario R2 was done during runtime. For tracking purposes of these two scenarios, a frame-to-frame SIFT feature matching algorithm was used. Figure 6.3 shows selected captured images of all five cameras at different points in time for scenario R1.

The camera network for the second setting is installed at the Universität Klagenfurt and employs six cameras. Figure 6.4 shows a rough floorplan with the location and fields of view of the different cameras (blue lines). In this setting, we employ both of the camera systems discussed in the previous section. This introduces heterogeneous hardware as well as two different communication channels. In Figure 6.4 the cameras 1-4 are SLR cameras,

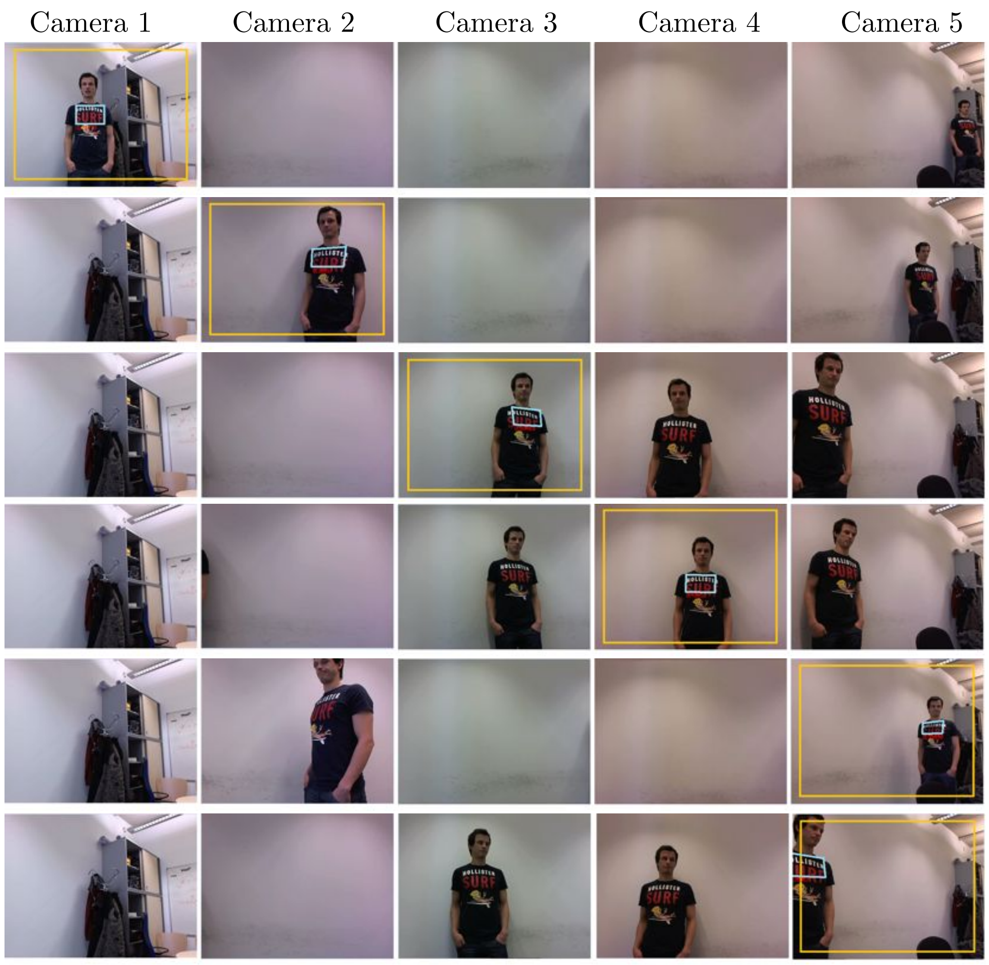


Figure 6.3: Captured images of test Scenario R1. Each column corresponds to one camera, and each row corresponds to a specific time point. The time difference between individual rows is about 15 seconds. The blue box represents the currently tracked object, and the orange box represents the boundary margin of the FOV of the camera.

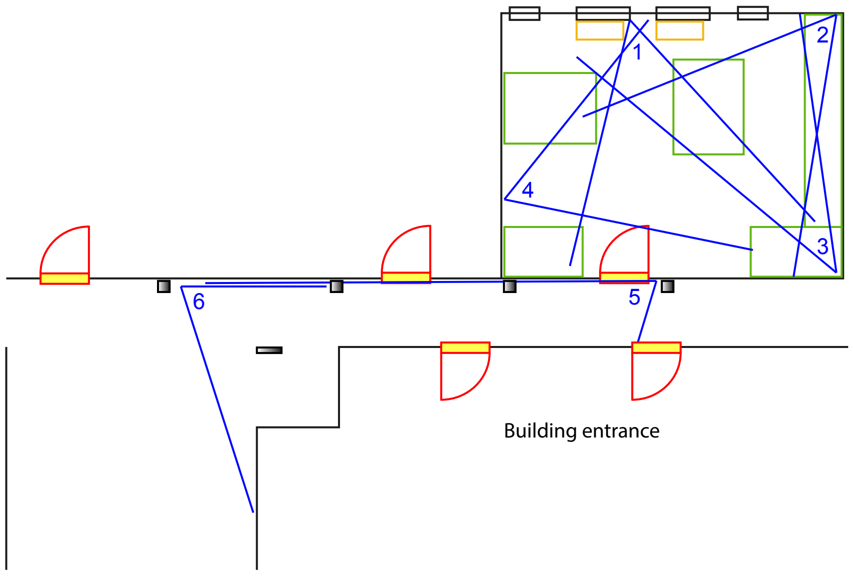
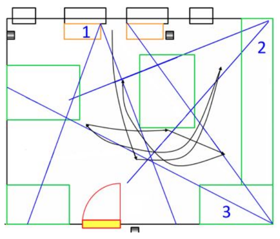


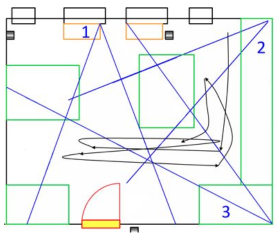
Figure 6.4: Floorplan including the location and orientation of the cameras within the network. Green squares depict standard height working tables. Doorways are marked yellow. Orange squares are cabinets.

while cameras 5 and 6 are PandaBoard based smart-cameras. For the evaluation of our work, we recorded various video sequences using these available cameras with different orientations. We recorded ten different scenarios to conduct our experiments. In seven scenarios (R3 – R9), only a subset of three cameras was used (camera 1 to 3). For the remaining three scenarios, all cameras have been used to record video sequence. This results in an overall of 39 video sequences. The cameras were not time synchronised. Figure 6.5 gives a rough overview over the scenarios we have recorded for evaluation. We refer to these scenarios as Scenarios R3 – R12 respectively. The blue lines show the FOV of the different cameras. The green rectangles show tables which might occlude the object partially. Orange blocks represent obstacles and dark squares depict pillars; both can occlude the person completely. The yellow box and the red circle segment represent the doorway and the opening area of the door respectively. The approximate movement paths and direction of the object being tracked are illustrated as black arrows. Figure 6.6 shows snapshots of the recorded data. For Scenarios R3 – R12, we employed a colour histogram based tracker, combined with background subtraction, to process the video feeds offline and used them as input for our simulation environment.

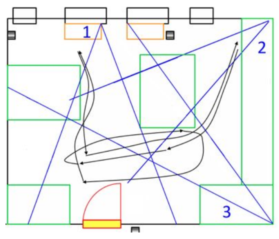
Similarly, an extended version of the video data of Scenario R2 was also used as input for our simulation tool to evaluate heterogeneous assignment in real camera networks



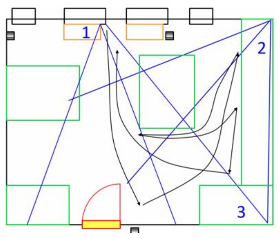
(a) Scenario R3
1834 frames



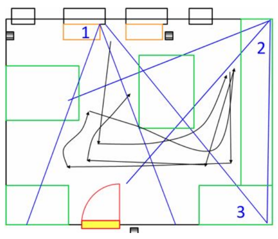
(b) Scenario R4
1859 frames



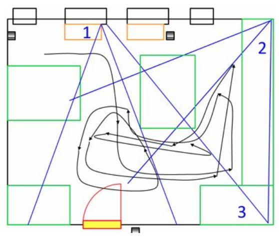
(c) Scenario R5
1816 frames



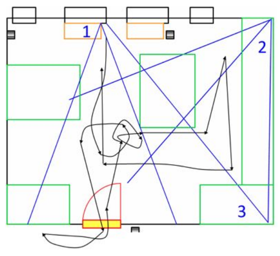
(d) Scenario R6
1953 frames



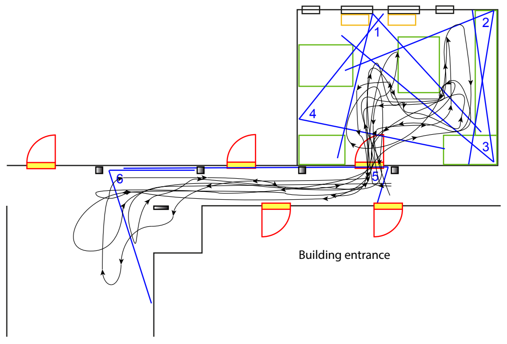
(e) Scenario R7
1987 frames



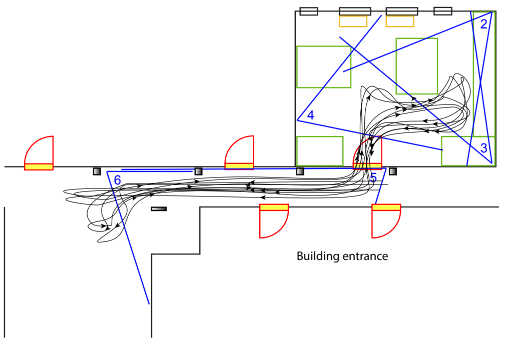
(f) Scenario R8
1986 frames



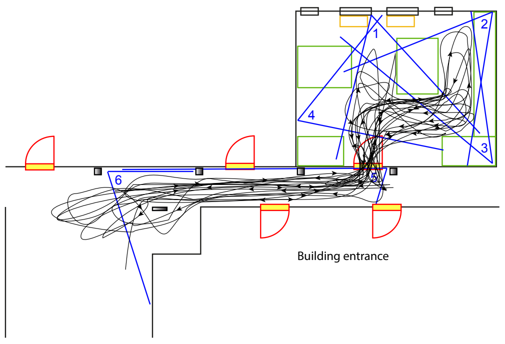
(g) Scenario R9
2307 frames



(h) Scenario R10
114 seconds

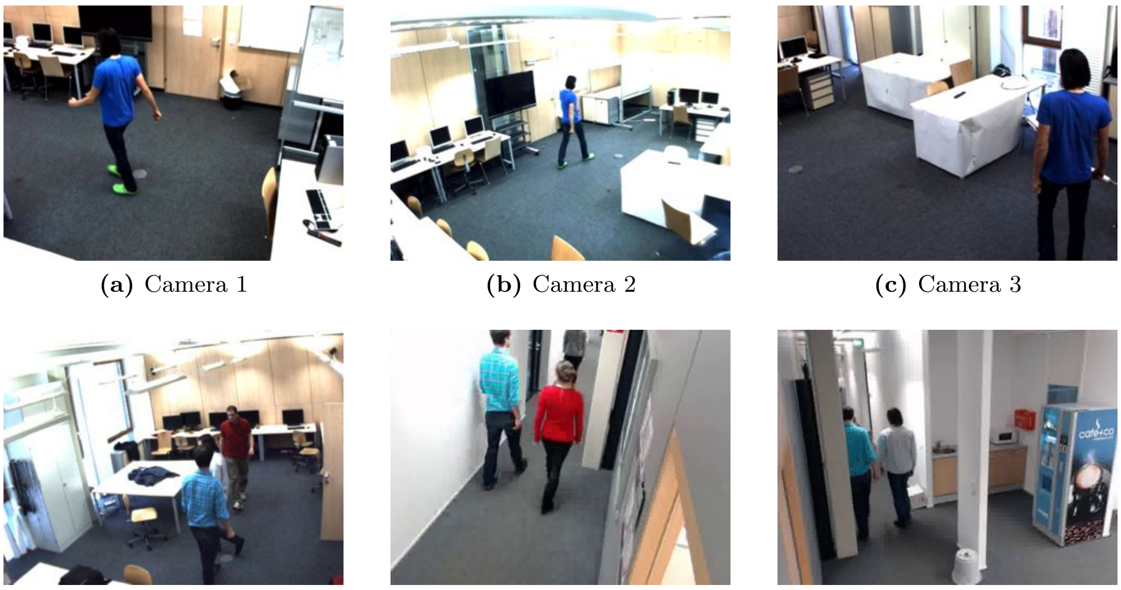


(i) Scenario R11
161 seconds



(j) Scenario R12
195 seconds

Figure 6.5: The scenarios tested with our simulation tool *CamSim* and their corresponding duration. Floorplan including the location and orientation of the cameras within the network. Green squares depict standard height working tables. Doorways are marked yellow. Orange squares are cabinets. The black lines represent the movement pattern of the person.



(a) Camera 1 **(b)** Camera 2 **(c)** Camera 3

(d) Camera 4 **(e)** Camera 5 **(f)** Camera 6

Figure 6.6: Snapshots of recorded video sequences for evaluation of all 6 cameras.

as well as autonomous learning of the best strategy for each camera. We refer to this special case as Scenario R13. The extension was achieved by looping the video feed. For Scenario R13, the video has been pre-processed using a frame-to-frame SIFT-based feature matching algorithm.

An overview of all real world based scenarios, the number of employed cameras, the duration as well as the implemented tracking algorithms is given in Table 6.1

6.4 Results

6.4.1 Auction Windows and Handover Times

When implementing the proposed algorithm on a real camera system, the significant effect of two key differences become apparent. Firstly, the simulator operates instantaneously using discrete time, with one auction corresponding to one time-step, while the real system is not synchronised and introduces time delays. Since the tracker is not able to process every single frame in real time due to the computationally limited smart cameras, the utility can neither be acquired from every captured frame. The deviation for the acquired utility is propagated and emerges with every handover between the cameras. Again, we ran our scenarios 30 times and, similar to our simulation, we calculated the average of the accumulated utility over all cameras in every second. We studied the cases while using no

Scenario	# Cameras	# Objects	Tracker	Duration
R1	5	1	SURF	1,750 frames
R2	5	1	SURF	3,250 frames
R3	3	1	CH	1,834 frames
R4	3	1	CH	1,859 frames
R5	3	1	CH	1,815 frames
R6	3	1	CH	1,950 frames
R7	3	1	CH	1,986 frames
R8	3	1	CH	1,984 frames
R9	3	1	CH	2,301 frames
R10	6	3	CH	2,850 frames
R11	5	3	CH	4,025 frames
R12	6	6	CH	4,875 frames
R13	5	1	SIFT	7,120 frames

Table 6.1: Overview of the real world scenarios used for evaluation. *SURF* refers to a SURF-based feature tracker and *SIFT* to a SIFT-based feature tracker respectively. *CH* refers to the Colour Histogram based tracking approach, employed to extract the model of the object of interest.

	Auction Duration		
Scenario	0 ms	500 ms	1000 ms
Scenario R1 - passive	4	4	4
Scenario R1 - active	9	6	6
Scenario R2 - passive	8	9	7
Scenario R2 - active	18	14	14

Table 6.2: Comparison of handovers using different auction windows within the active and passive approach in Scenario R1 and Scenario R2.

auction window as well as when the auction window was 500 milliseconds and one second for both our active and our passive approaches; Figure 6.7 shows the results of this for Scenario R1 and Scenario R2. Table 6.2 compares the number of handovers for different scenarios using our active and passive approaches with different auction windows.

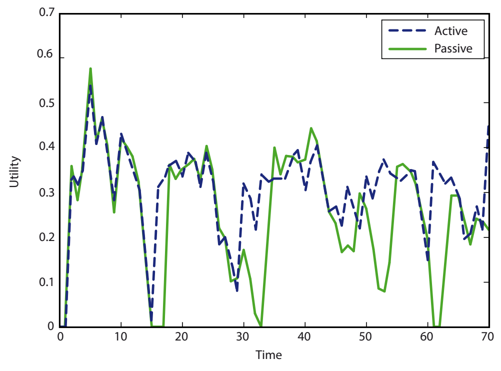
As described in Step 4 of Algorithm 1, our algorithm has to be repeated at a regular interval. We found that the choice of the interval to repeat our algorithm as well as the choice of the auction window affects the utility obtained by each camera, and hence the underlying tracking performance of the network. Altering the auction window has the same effect as altering the interval at which the algorithm is repeated, since the cameras do not re-advertise objects while waiting for existing bids to arrive. This is also due to the fact that handing over tracking responsibility from one camera to another does not work seamlessly. As soon as the auction ends and the winning camera has been determined, the tracker at the *selling* camera is stopped and the tracking responsibility is handed over to the winning camera. During this short time no utility is gathered, and hence utility is *lost*. In case the tracker gains only little utility by switching to another camera, this lost utility is not compensated. The illustrations in Figure 6.7 show this effect in certain situations (e.g., Scenario R2 with 500 ms auction window), where the active approach does not constantly perform as well as the passive approach. Figure 6.8 clearly shows the deviation in overall performance for small auction windows, where our active approach without an auction window does not lie on the Pareto front. To overcome this problem, one could think of employing suitable learning techniques in each camera locally, to reason about appropriate timings for auction invitations and auction durations, but this goes beyond the focus of this thesis.

The second key difference when executing our proposed algorithm in smart cameras is the processing power which is available to the employed trackers. Such resources can quickly become consumed when too many trackers are operating concurrently, for example due to a large number of objects being tracked, or the need to respond to a large number of auction calls. In case multiple trackers run concurrently, every single tracker will process fewer frames per second. This enhances the effect of creating less utility for a certain camera and the respective trackers within a discrete time window.

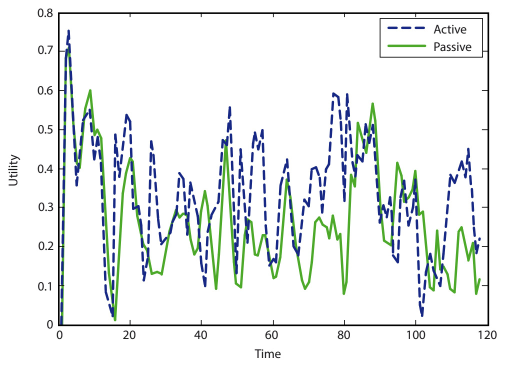
As in our simulation environment, we generated the vision graph during runtime. Figure 6.9 illustrates the generation of the vision graph during runtime for Scenario R2 using the BROADCAST communication policy. Figure 6.10 illustrates the same effect in Scenario R12 for a larger network. For both scenarios one can see how the vision graph is built up over time, and how it evaporates due to a lack of handovers between the cameras on the left.

6.4.2 Performance Analysis

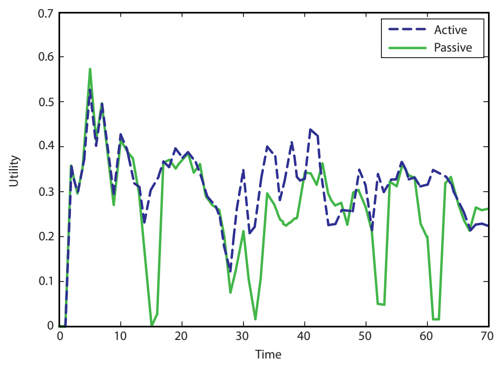
Figure 6.11 shows the overall performance of our ACTIVE and PASSIVE approaches using BROADCAST, SMOOTH and STEP communication policies, for Scenario R2, with one object in the environment. Again, there is a clear trade-off between the achieved utility and communication. It is also apparent that the results of the deployed system are very similar to the equivalent scenario studied in the simulation environment (cp. Figure 5.3).



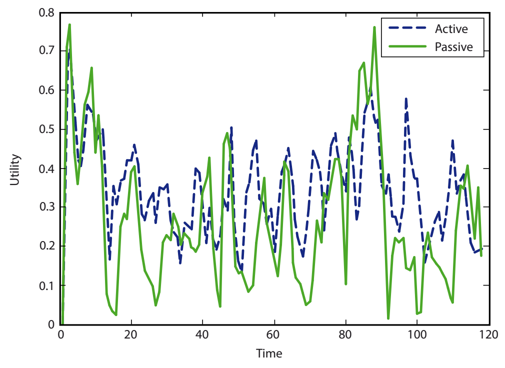
(a) Scenario R1 - 1 Second



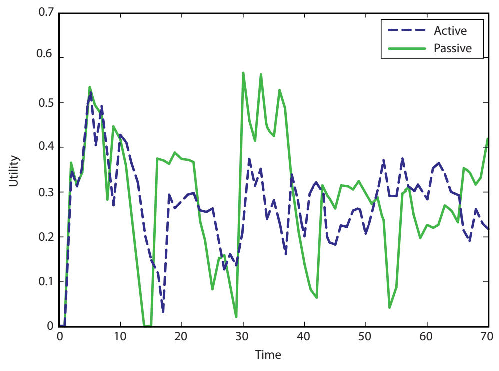
(b) Scenario R2 - 1 Second



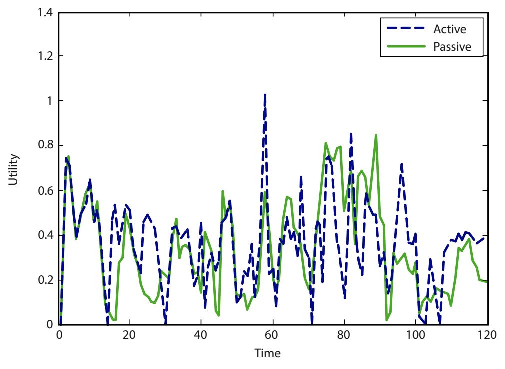
(c) Scenario R1 - 500 ms



(d) Scenario R2 - 500 ms



(e) Scenario R1 - 0 ms



(f) Scenario R2 - 0 ms

Figure 6.7: Illustrations of the accumulated utility over time, utilising the 1 second auction window for auctions used in the ACTIVE and PASSIVE approach in Scenario R1 as well as for Scenario R2 (top row), the 500 millisecond auction window for auctions used in ACTIVE and PASSIVE approach in the first and second scenario (middle row) and both approaches without an auction window (bottom row).

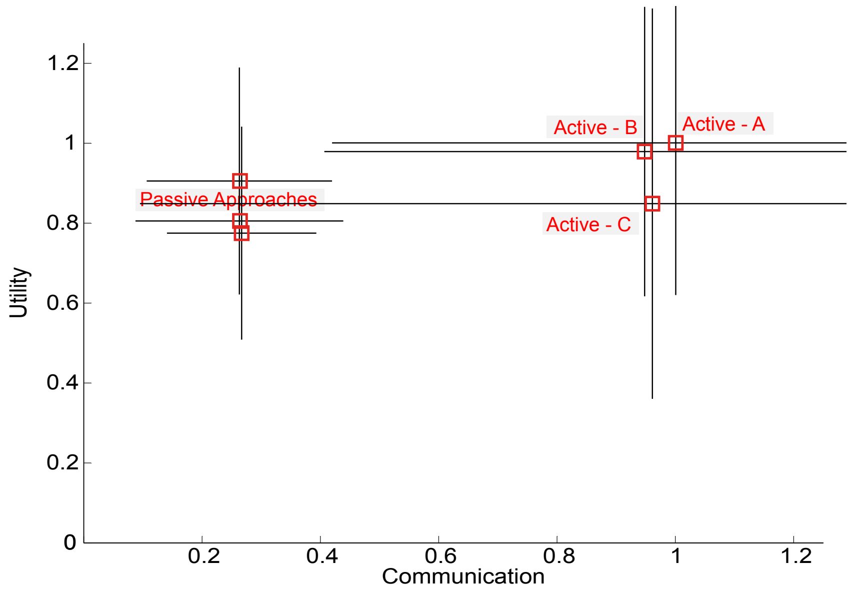
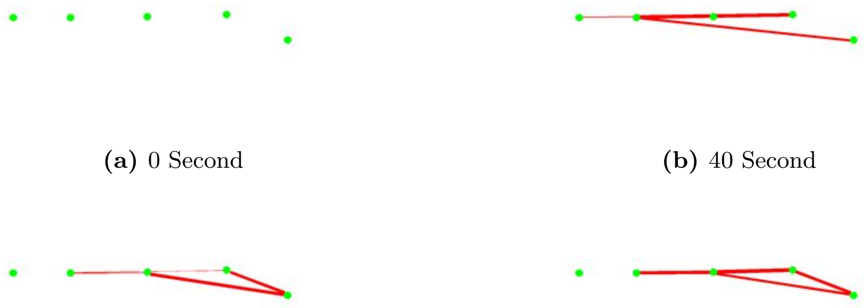


Figure 6.8: Direct comparison of the different durations of auctions in our ACTIVE approach versus the different auction windows in our PASSIVE approach. Again, *Active - A*, *Active - B*, and *Active - C* represent the active approach initiating an auction every third processed frame, with timing windows of 1 second, 500 ms, and no timing window respectively. It is apparent that the selection of an appropriate auction window has a high impact on the overall performance.



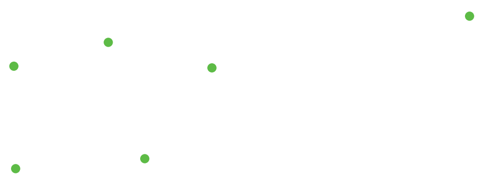
(a) 0 Second

(b) 40 Second

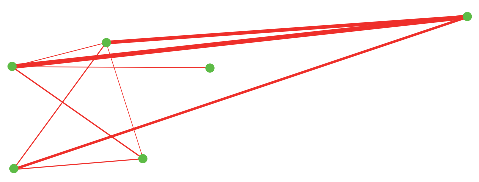
(c) 80 Second

(d) 120 Second

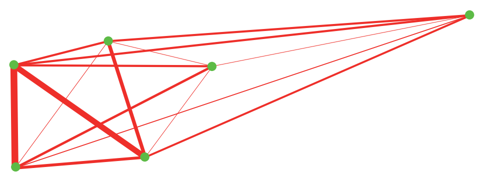
Figure 6.9: The vision graph is built up during runtime through trading interactions for scenario R2 in our real camera network using the ACTIVE BROADCAST strategy. Dots indicate cameras, lines indicate links in the vision graph. The thickness of the line indicates strength of the corresponding link.



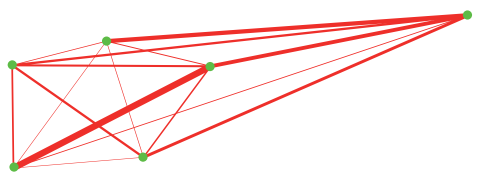
(a) 500 time steps



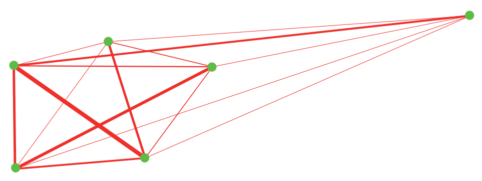
(b) 1000 time steps



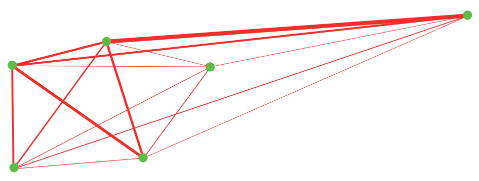
(c) 1500 time steps



(d) 2000 time steps



(e) 2500 time steps



(f) 3000 time steps

Figure 6.10: Snapshots of the generated vision graph using our market-based approach for Scenario R12 at various time steps. Thickness of the red lines indicate the strength of the link in the vision graph.

Furthermore, we utilised recorded video data from Scenario R3 – R12 using our second camera setting. This data was pre-processed using a background weighing algorithm, and afterwards colour histograms to identify and assign labels to the tracked objects. Figures 6.12, 6.13, and 6.14 show a selection of results for our tested scenarios. In general, as with our simulated scenarios, the drop in communication when using the different auction schedules on the cameras is apparent. Based on our previous experience, the drop in confidence is less significant then expected for this real world scenarios when compared to simulated scenarios. All scenarios were normalized by the ACTIVE BROADCAST approach, which gives us the best tracking confidence possible, but also has the highest communication overhead. For each frame the confidence is given by the tracker. The communication

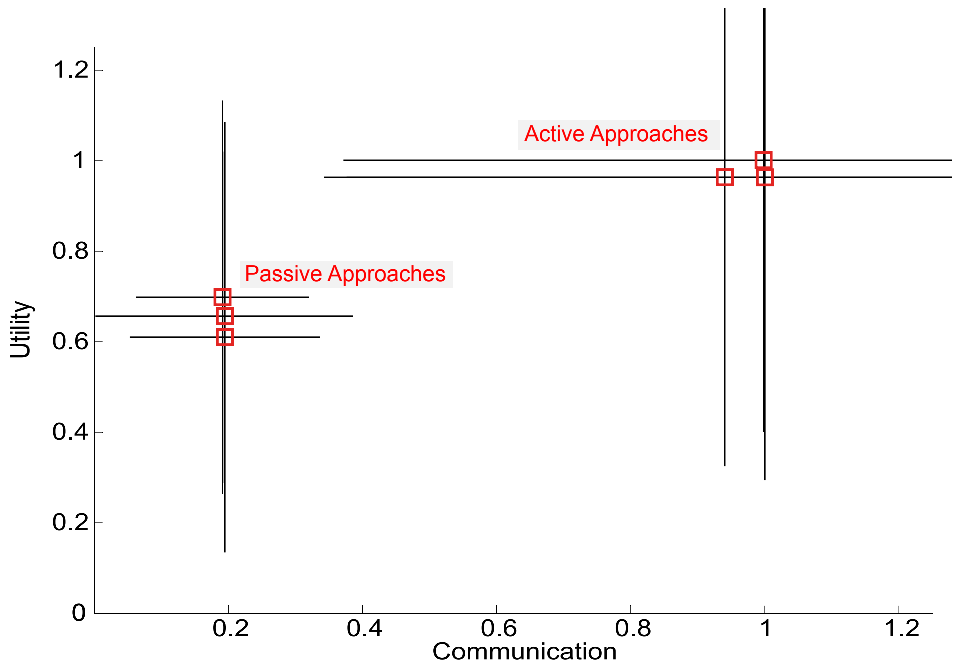


Figure 6.11: Performance (overall utility calculated over duration of scenarios) of each of the six marked-based algorithms in the real world deployment of Scenario R2. Both, utility and communication values, are normalised by those of the ACTIVE BROADCAST. The trade-off between the two approaches in a real system is apparent.

overhead represents the sent out messages at the local and processing overhead at remote camera at the same time.

Figure 6.12 shows the result for Scenario R5, depicted in Figure 6.5c. In this particular scenario, we had the highest drop in utility over the whole duration of the run. However, we were still able to achieve between 88% and 82% of the utility, while requiring only 33% to 16% of the communication when using our PASSIVE approaches in comparison to the ACTIVE BROADCAST approach. For the ACTIVE approaches, we had a drop of only 1% in utility but compared to ACTIVE BROADCAST reduced the number of initiated auctions by 19% and 24% for SMOOTH and STEP respectively.

Figure 6.13 shows a similar result for Scenario R6, where a clear Pareto front is created by our different approaches, in which no approach dominates any other approach. We were able to reduce the number of sent out messages by up to 68%, while keeping up to 86% of the possible utility when compared to the ACTIVE BROADCAST approach.

In our last exemplary results, shown in Figure 6.14, for Scenario R7, it becomes apparent that the SMOOTH communication strategy dominates the corresponding BROAD-

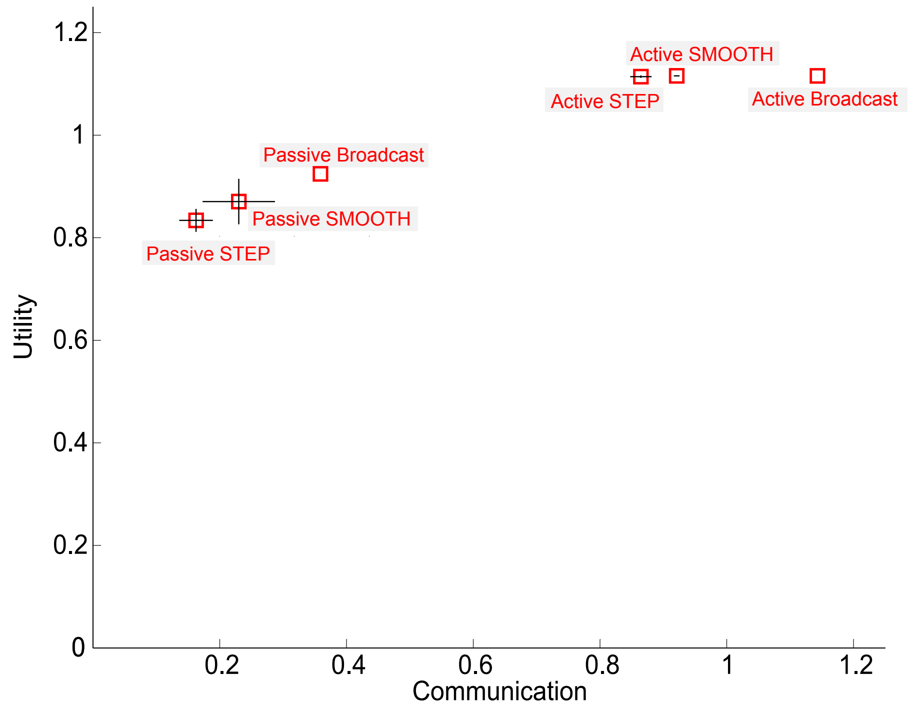


Figure 6.12: Performance (overall utility calculated over duration of scenarios) of our six marked-based algorithms in the real world deployment of Scenario R5. Both, utility and communication values, are normalised by those of the ACTIVE BROADCAST. Experiments have been repeated 30 times; standard deviation is given.

CAST communication policy for the PASSIVE and ACTIVE approach respectively. ACTIVE SMOOTH can furthermore reduce its communication effort by 14% compared to ACTIVE BROADCAST, and PASSIVE SMOOTH was able to reduce its communication effort by 8% compared to PASSIVE BROADCAST. PASSIVE STEP was even able to reduce communication by more than 55%, while only reducing its own confidence by 7% of the overall possible gathered utility given by the ACTIVE BROADCAST approach.

Even though the results are similar in two different experiments using the same camera setting, it becomes apparent that the movement patterns of objects and persons of interest do have an impact on the network-wide performance.

6.4.3 Performance of Autonomous and Static Heterogeneous Configurations in Real Networks

To analyse static heterogeneous assignments and online learnt configurations for camera networks in more realistic settings, we used scenario R13 and employed a SIFT-based

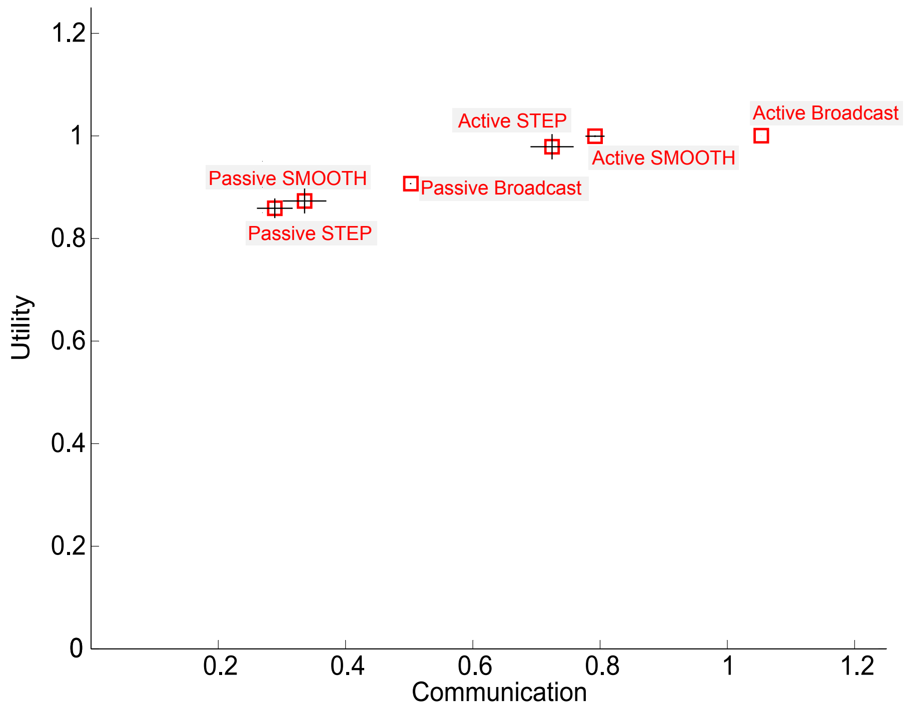


Figure 6.13: Performance (overall utility calculated over duration of scenarios) of our six marked-based algorithms in the real world deployment of Scenario R6. Both, utility and communication values, are normalised by those of the ACTIVE BROADCAST. Experiments have been repeated 30 times; standard deviation is given.

tracking approach [6] to detect and track a person within the network of cameras. Each camera captured 1780 frames. We repeated the video feed four times to create a total of 7120 frames for a longer evaluation video.

Figure 6.15 focusses again on the results obtained from all homogeneous and heterogeneous configurations as well as those obtained employing decentralised online learning. As with the results in Section 3.5.1, heterogeneous configurations lead to system wide outcomes, which are more Pareto efficient then those possible in the homogeneous case. Furthermore, as with the results in Section 3.5.2, the use of decentralised online learning of marketing strategies also extended the Pareto efficient frontier when compared to the homogeneous case. However, in this case, learning was not able to generate outcomes dominating the most Pareto efficient heterogeneous cases. We speculate that this is due to the presence of already highly Pareto efficient outcomes from static heterogeneous configurations, which would be difficult to find while still exploring the space sufficiently. It is likely that dynamic configurations in more complex real world scenarios will yield similarly

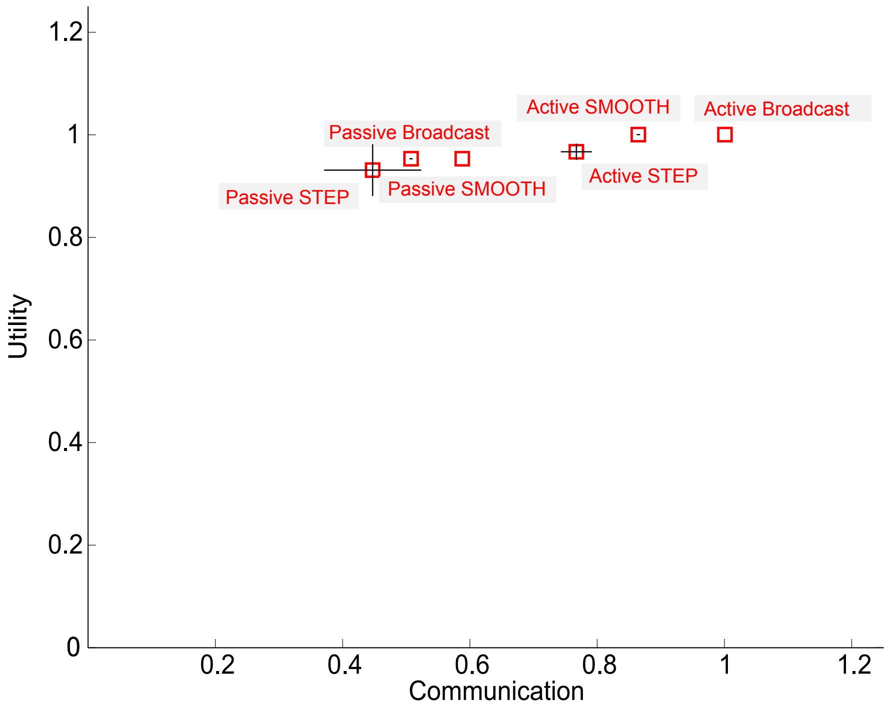


Figure 6.14: Performance (overall utility calculated over duration of scenarios) of our six marked-based algorithms in the real world deployment of Scenario R7. Both, utility and communication values, are normalised by those of the ACTIVE BROADCAST. Experiments have been repeated 30 times; standard deviation is given.

interesting results like those described in Section 3.5.2, but this remains an area for future research. However, as shown in Table 5.1, the learnt configuration is efficient compared to statically homogeneous assigned strategies. Identifying the best heterogeneous configuration requires knowledge of the camera network topology as well as the movement patterns of the objects of interest. Moreover, it is highly time consuming and computationally expensive.

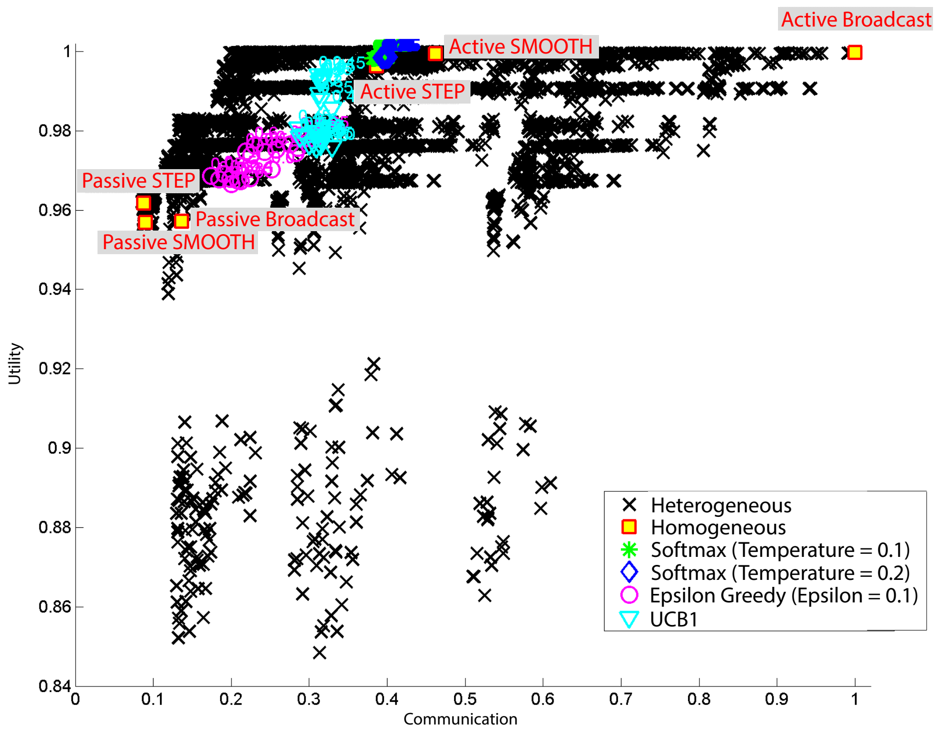


Figure 6.15: Performance of all configurations on scenario $R13$.

Seventh Chapter

Conclusion and Future Work

This thesis has presented a solution for distributed tracking of objects in networks of autonomous smart cameras. In contrast to multi-camera tracking, distributed tracking assigns tracking responsibility of an object only to a single camera at a time. This requires the network to decide on its own which camera is responsible for tracking an object at what time.

In typical handover approaches, either a centralised control is utilised, where *a priori* knowledge for each camera about its environment and neighbourhood relationships is provided, or the employed cameras are calibrated. In contrast, we do not rely on any *a priori* knowledge at all and assume uncertainty regarding location and orientation of the cameras as well as the movement patterns of the objects of interest. Inspired by market-mechanism, we implemented a single-sealed bid auction mechanism on each camera and enabled them to trade tracking responsibilities for objects of interest. Getting to know their trading partners, each camera is further induced with the capability of learning its local neighbourhood, the so-called *vision graph*. This allowed each camera in the system to reduce its own communication overhead, and hence the communication effort of the entire system. Making use of biology-inspired foraging mechanisms, we have implemented artificial pheromones to build up the vision graph, and simultaneously enabling the cameras to forget about neighbours, wherever the response rate to advertised auctions diminishes over time.

To advertise auctions in the network of smart cameras to prospective buyers, we have described six different strategies exploiting the vision graph. Each of these strategies attempts to realise one of two objectives: (i) minimising network-wide communication or; (ii) maximising system-wide tracking performance. The selection of an appropriate configuration using a variety of strategies in the smart camera network turned out to require knowledge of the camera setup, the environment, and the movement patterns of the objects of interest. To neglect *a priori* knowledge of these parameters, we have implemented multi-armed bandit problem solvers in every camera of the network. This enables the cameras to learn on their own which strategy fits them best, given a priority on either minimising communication or maximising tracking performance. Indeed, we were able to show that cameras are able to improve their network-wide performance when learning their own strategy during runtime compared to obstinately assigning homogeneous strategies.

7.1 Summary of Contributions

More specifically, this thesis provides the following contributions:

- Formulation of a market-based approach to assign tracking responsibilities within a network of smart cameras without central coordination. The presented approach enables a camera to transfer the tracking responsibility to a neighbouring camera ‘seeing’ an object best, at a given time step.

- Introduction of artificial pheromones to create links between cameras based on the trading behaviour of tracked objects. This allows the cameras to organise themselves in the network and build up the so-called vision graph without any central server. Additionally, evaporating pheromones let cameras forget about changing environmental situations or movement behaviour of objects.

- Description of two auction strategies which allow the cameras to identify their unique neighbourhood relations with other cameras in the network. Furthermore, presentation of three communication policies exploiting the private vision graph of each camera in order to reduce the number of exchanged messages within the network. Additionally, we compared all possible combinations of auction strategies and communication policies and evaluated their trade-offs regarding network-wide communication overhead and tracking performance.

- A definition of various uncertainties regarding the cameras in a network, such as adding, removing and changing the location and orientation of a camera during runtime. Additionally, an analysis of the robustness of the presented approaches regarding these uncertainties in camera networks was presented.

- A comparison of homogeneous as well as heterogeneous assigned strategies and policies, showing the benefits of heterogeneous assignment of approaches based on the unique neighbourhood relationships of each camera. To overcome the problem of *a priori* knowledge regarding the orientation, location and neighbourhood relations of smart cameras to assign the appropriate strategy to each camera, this thesis also presents an approach using multi-arm bandit problem solver to allow the cameras to self-organise their available algorithms during runtime without any supervised learning or central control.

- Development of a custom simulation environment, called *CamSim*, to allow fast testing and evaluation of the presented approaches, policies and strategies.

- Deployment of a heterogeneous smart-camera network in lab conditions consisting of four custom built smart cameras by the company SLR and two smart cameras built with off-the-shelf components.

7.2 Future Work

There are numerous directions for future work. In this thesis, we considered a rather naive resource model, disregarding the actual cost of tracking multiple objects within a single camera. A camera, having the responsibility of tracking multiple objects at the same time, might want to consider remaining resources when submitting a bid for a new object. Having cameras bidding for an object but not being able to deliver appropriate tracking performance afterwards, impairs the network-wide quality. Hence, integrating resource information of a camera in the local valuation function might prove beneficial for the system-wide outcome. Furthermore, this information would allow for adaptation of α in the reward function of the bandit solvers. This would enable each camera to select an appropriate strategy according to locally available resources.

Another strain of future research could induce cameras with the ability to learn about the movement patterns of the objects of interest. This might have multiple benefits. First, each camera could adapt the amount of pheromones to strengthen a link Δ when trading objects and the evaporation rate of pheromones ρ depending on the amount and frequency of objects passing through the respective FOV. Second, cameras could weigh the expected utility of an advertised object by currently owned objects. In case a new object is advertised but the camera is not able to bid for it, the camera has to make a decision on selling currently owned objects or refrain from bidding for new object. This might be beneficial when the new object is expected to result in better performance than any of the remaining objects. Third, having an idea about the movement patterns as well as movement speed of objects could allow the cameras to adapt their time of auction initiation. This means that a camera could decide per object when to initiate an auction, and hence try to hand over the object to another camera. A similar direction of future research could allow the cameras to explore different durations for auctions on their own. This is advantageous, since auction timings are highly dependent on the location of other cameras in the network.

Finally, in our presented approach, objects are offered either to all cameras in the network or a certain subset. Indeed, there is no discrimination between the location of cameras. In case an object leaves the FOV on the right side of a camera, all neighbouring cameras, regardless whether they are on the left or the right side of the selling camera, will receive the auction invitation. Hence, clustering the vision graph based on the location of sold objects could allow the selling camera to further discriminate the local vision graph, and therefore reduce its personal communication overhead even more.

Bibliography

[1] Refined Demonstrators. Deliverable d1-4, EPiCS - Engineering Proprioception in Computing Systems, June 2013.

[2] W. Abbas and M. Egerstedt. Distribution of Agents in Heterogeneous Multiagent Systems. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC 2011)*, pages 976–981. IEEE Computer Society Press, 2011.

[3] G. Anders, C. Hinrichs, F. Siefert, P. Behrmann, W. Reif, and M. Sonnenschein. On the Influence of Inter-Agent Variation on Multi-Agent Algorithms Solving a Dynamic Task Allocation Problem under Uncertainty. In *Proceedings of the Sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2012)*, pages 29–38. IEEE Computer Society Press, 2012.

[4] O. Ardaiz, P. Artigas, T. Eymann, F. Freitag, L. Navarro, and M. Reinicke. The Catallaxy Approach for Decentralized Economic-based Allocation in Grid Resource and Service Markets. *Applied Intelligence*, 25(2):131–145, 2006.

[5] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47:235–256, 2002.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. Speeded-up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[7] H. Bay, T. Tuytelaars, and L. van Gool. Surf: Speeded up robust features. In *Proceedings of the Ninth European Conference on Computer Vision (ECCV 2006)*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg, 2006.

[8] H. Bester. Bargaining versus price competition in markets with quality uncertainty. *The American Economic Review*, 83(1):278–288, 1993.

[9] J. Black, T. Ellis, and P. Rosin. Multi View Image Surveillance and Tracking. In *Proceedings of the Workshop on Motion and Video Computing, 2002*, pages 169–174. IEEE Computer Society Press, 2002.

[10] A. Campbell, C. Riggs, and A. S. Wu. On the Impact of Variation on Self-Organizing Systems. In *Proceedings of the Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2011)*, pages 119–128. IEEE Computer Society Press, 2011.

[11] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Computer Survey*, 41(3):15:1–15:58, 2009.

[12] A. Chandra, P. S. Oliveto, and X. Yao. Co-evolution of Optimal Agents for the Alternating Offers Bargaining Game. In *Proceedings of the European Conference on the Applications of Evolutionary Computation (EvoApplications 2010)*, volume 6024 of *Lecture Notes in Computer Science*, pages 61–70. Springer Berlin Heidelberg, 2010.

[13] Z. Cheng, D. Devarajan, and R. J. Radke. Determining Vision Graphs for Distributed Camera Networks Using Feature Digests. *EURASIP Journal on Applied Signal Processing*, 2007(1):220–220, Jan. 2007.

[14] A. Cichowski, C. Madden, H. Detmold, A. Dick, A. van den Hengel, and R. Hill. Tracking Hand-off in Large Surveillance Networks. In *Proceedings of the 24th International Conference Image and Vision Computing New Zealand (IVCNZ 2009)*, pages 276–281. IEEE Computer Society Press, 2009.

[15] S. H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation.* World Scientific Publishing Co., Inc., 1996.

[16] S. H. Clearwater, R. Costanza, M. Dixon, and B. Schroeder. Saving energy using market-based control. In *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pages 253–273. World Scientific Publishing Co., Inc., 1996.

[17] D. Cliff. Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments. Technical Report HPL-97-91, HP Labs, 1997.

[18] D. Cliff and J. Bruten. Simple Bargaining Agents for Decentralized Market-Based Control. Technical Report HPL-98-17, HP Laboratories, 1998.

[19] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[20] H. Detmold, A. van den Hengel, A. Dick, A. Cichowski, R. Hill, E. Kocadag, K. Falkner, and D. S. Munro. Topology Estimation for Thousand-Camera Surveillance Networks. In *Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras*, pages 195–202. IEEE Computer Society Press, 2007.

[21] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[22] B. Dieber, L. Esterle, and B. Rinner. Distributed Resource-aware Task Assignment for Complex Monitoring Scenarios in Visual Sensor Networks. In *Proceedings of the Sixth International Conference on Distributed Smart Cameras (ICDSC 2012)*, pages 1–6. IEEE Computer Society Press, Oct 2012.

[23] B. Dieber, C. Micheloni, and B. Rinner. Resource-Aware Coverage and Task Assignment in Visual Sensor Networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(10):1424–1437, 2011.

[24] M. Dorigo and T. Stützle. *Ant Colony Optimization.* The MIT Press, 2004.

[25] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet Advertising and the Generalized Second Price Auction: Selling Billions of Dollars Worth of Keywords. Technical report, National Bureau of Economic Research, 2005.

[26] T. Eichstädt. *Einsatz von Auktionen im Beschaffungsmanagement.* Gabler, 2008.

[27] T. Ellis, D. Makris, and J. Black. Learning a Multi-camera Topology. In *Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 165–171. IEEE Computer Society Press, 2003.

[28] U. M. Erdem and S. Sclaroff. Look There! Predicting Where to Look for Motion in an Active Camera Network. In *Proceedings of the IEEE Conference on Vision and Signal-based Surveillance*, pages 105–110. IEEE Computer Society Press, 2005.

[29] L. Esterle. CamSim. `https://github.com/EPiCS/Camsim`, 2013.

[30] L. Esterle, H. Caine, P. R. Lewis, X. Yao, and B. Rinner. CamSim: A Smart Camera Network Simulator. In *Proceedings of the Seventh IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2013)*, pages 1–2. IEEE Computer Society Press, 2013.

[31] L. Esterle, P. R. Lewis, M. Bogdanski, B. Rinner, and X. Yao. A Socio-economic Approach to Online Vision Graph Generation and Handover in Distributed Smart Camera Networks (ICDSC 2011). In *Proceedings of the Fifth ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–6. IEEE Computer Society Press, 2011.

[32] L. Esterle, P. R. Lewis, B. Rinner, and X. Yao. Improved Adaptivity and Robustness in Decentralised Multi-Camera Networks. In *Proceedings of the Sixth ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–6. IEEE Computer Society Press, 2012.

[33] L. Esterle, P. R. Lewis, X. Yao, and B. Rinner. Socio-economic vision graph generation and handover in distributed smart camera networks. *ACM Transactions on Sensor Networks*, 10(2):20:1–20:24, 2014.

[34] M. Esteva and J. Padget. Auctions without Auctioneers: Distributed Auction Protocols. In *Agent Mediated Electronic Commerce II*, volume 1788 of *Lecture Notes in Computer Science*, pages 220–238. Springer Berlin Heidelberg, 2000.

[35] T. Eymann. Co-evolution of Bargaining Strategies in a Decentralized Multi-agent System. In *Proceedings of the Symposium on Negotiation Methods for Autonomous Cooperative Systems*, pages 126–134. Association for the Advancement of Artificial Intelligence, 2001.

[36] R. Farrell and L. S. Davis. Decentralized discovery of camera network topology. In *Proceedings of the Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2008)*, pages 1–10. IEEE Computer Society Press, 2008.

[37] C. Frank and K. Römer. Algorithms for Generic Role Assignment in Wireless Sensor Networks. In *Proceedings of the Third International Conference on Embedded Networked Sensor Systems*, pages 230–242. ACM Press, 2005.

[38] R. A. Gagliano and P. A. Mitchem. Valuation of network computing resources. In *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pages 28–52. World Scientific Publishing Co., Inc., 1996.

[39] M. Gagliolo and J. Schmidhuber. Algorithm Portfolio Selection as a Bandit Problem with Unbounded Losses. *Annals of Mathematics and Artificial Intelligence*, 61(2):49–86, 2011.

[40] E. H. Gerding, R. K. Dash, D. C. K. Yuen, and N. R. Jennings. Optimal Bidding Strategies for Simultaneous Vickrey Auctions with Perfect Substitutes. In *Proceedings of the Eighth Workshop on Game Theoretic and Decision Theoretic Agents*, page 10–17, 2006.

[41] E. H. Gerding, R. K. Dash, D. C. K. Yuen, and N. R. Jennings. Bidding Optimally in Concurrent Second-price Auctions of Perfectly Substitutable Goods. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 267–274, 2007.

[42] E. H. Gerding and J. A. La Poutré. Bilateral Bargaining with Multiple Opportunities: Knowing your Opponent's Bargaining Position. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 36(1):45–55, 2006.

[43] G. Gerrard and R. Thompson. Two Million Cameras in the UK. *CCTV Image*, 42:10–12, 2011.

[44] P. Gradwell, M. Oey, R. Timmer, F. Brazier, and J. Padget. Engineering Large-scale Distributed Auctions. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 1311–1314. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[45] A. Gupta, D. O. Stahl, and A. B. Whinston. The Economics of Network Management. *Communications of the ACM*, 42(9):57–63, 1999.

[46] D. Hausheer and B. Stiller. PeerMart: The Technology for a Distributed Auction-based Market for Peer-to-Peer Services. In *Proceedings of the IEEE International Conference on Communications*, volume 3, pages 1583–1587, 2005.

[47] E. Hjelmås and B. K. Low. Face Detection: A Survey. *Computer Vision and Image Understanding*, 83(3):236 – 274, 2001.

[48] O. Javed, S. Khan, Z. Rasheed, and M. Shah. Camera Handoff: Tracking in Multiple Uncalibrated Stationary Cameras. In *Proceedings of the Workshop on Human Motion, 2000*, pages 113–118. IEEE Computer Society Press, 2000.

[49] J. Ketcham, V. L. Smith, and A. W. Williams. A Comparison of Posted-offer and Double-auction Pricing Institutions. *The Review of Economic Studies*, 51(4):595–614, 1984.

[50] H. Kim, J. Romberg, and W. Wolf. Multi-Camera Tracking on a Graph Using Markov Chain Monte Carlo. In *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2009)*, pages 1–8. IEEE Computer Society Press, 2009.

[51] P. Klemperer. Auction Theory: A Guide to the Literature. *Journal of Economic Surveys*, 13(3):227–286, 1999.

[52] P. R. Lewis, L. Esterle, A. Chandra, B. Rinner, J. Torresen, and X. Yao. Static, dynamic and adaptive heterogeneity in socio-economic distributed smart camera networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2014. accepted.

[53] P. R. Lewis, L. Esterle, A. Chandra, B. Rinner, and X. Yao. Learning to be different: Heterogeneity and efficiency in distributed smart camera networks. In *Proceedings of the Seventh IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2013)*, pages 209–218. IEEE Computer Society Press, 2013.

[54] P. R. Lewis, P. Marrow, and X. Yao. Resource Allocation in Decentralised Computational Systems: An Evolutionary Market Based Approach. *Journal of Autonomic Agents and Multi-Agent Systems*, 21(2):143–171, 2010.

[55] Y. Li and B. Bhanu. A Comparison of Techniques for Camera Selection and Handoff in a Video Network. In *Proceedings of the Third ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–8. IEEE Computer Society Press, 2009.

[56] Y. Li and B. Bhanu. Utility-based Camera Assignment in a Video Network: A Game Theoretic Framework. *IEEE Sensors Journal*, 11(3):676–687, 2011.

[57] F. Lopes, M. Wooldridge, and A. Q. Novais. Negotiation Among Autonomous Computational Agents: Principles , Analysis and Challenges. *Artificial Intelligence Review*, 2008(29):1–44, 2008.

[58] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[59] C. C. Loy, T. Xiang, and S. Gong. Multi-camera Activity Correlation Analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 1988–1995. IEEE Computer Society Press, 2009.

[60] D. Makris, T. Ellis, and J. Black. Bridging the Gaps between Cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, pages 205–210. IEEE Computer Society Press, 2004.

[61] Z. Mandel, I. Shimshoni, and D. Keren. Multi-Camera Topology Recovery from Coherent Motion. In *Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2007)*, pages 243–250. IEEE Computer Society Press, 2007.

[62] P. R. Milgrom and R. J. Weber. A Theory of Auctions and Competitive Bidding. *Econometrica*, 50(5):1089–1122, 1982.

[63] M. S. Miller, D. Krieger, N. Hardy, C. Hibbert, and E. D. Tribble. An automated auction in atm network bandwidth. In *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pages 96–125. World Scientific Publishing Co., Inc., 1996.

[64] T. B. Moeslund and E. Granum. A Survey of Computer Vision-based Human Motion Capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[65] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based Object Detection in Images by Components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001.

[66] B. Möller, T. Plötz, and G. A. Fink. Calibration-free Camera Hand-Over for Fast and Reliable Person Tracking in Multi-Camera Setups. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR 2008)*, pages 1–4. IEEE Computer Society Press, 2008.

[67] K. Morioka, S. Kovacs, J.-H. Lee, and P. Korondi. A Cooperative Object Tracking System with Fuzzy-Based Adaptive Camera Selection. *International Journal on Smart Sensing and Intelligent Control*, 3(3):338–358, 2010.

[68] P. Mörters and Y. Peres. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2010.

[69] L. Mottola and G. Picco. Logical Neighborhoods: A Programming Abstraction for Wireless Sensor Networks. In *Distributed Computing in Sensor Systems*, volume 4026 of *Lecture Notes in Computer Science*, pages 150–168. Springer Berlin Heidelberg, 2006.

[70] E. F. Nakamura, H. S. Ramos, L. A. Villas, H. A. de Oliveira, A. L. de Aquino, and A. A. Loureiro. A Reactive Role Assignment for Data Routing in Event-based Wireless Sensor Networks. *Computer Networks*, 53(12):1980–1996, 2009.

[71] C. R. Plott and V. L. Smith. An Experimental Examination of Two Exchange Institutions. *The Review of Economic Studies*, pages 133–153, 1978.

[72] A. Prasath, A. Venuturumilli, A. Ranganathan, and A. A. Minai. Self-Organization of Sensor Networks with Heterogeneous Connectivity. In G. Ferrari, editor, *Sensor Networks: Where Theory Meets Practice*, pages 39–59. Springer, 2009.

[73] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl. Autonomous Multicamera Tracking on Embedded Smart Cameras. *EURASIP Journal on Embedded Systems Volume 2007*, 2007(1):35–45, 2007.

[74] F. Qureshi and D. Terzopoulos. Multi-camera Control through Constraint Satisfaction for Persistent Surveillance. In *Proceedings of the IEEE Conference on Vision and Signal-based Surveillance*, pages 211–218. IEEE Computer Society Press, 2008.

[75] M. Reisslein, B. Rinner, and A. Roy-Chowdhury. Smart Camera Networks (Guest Editors' Introduction). *Computer*, 47(5), 2014.

[76] B. Rinner, B. Dieber, L. Esterle, P. R. Lewis, and X. Yao. Resource-aware Configuration in Smart Camera Networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2012)*, pages 58–65. IEEE Computer Society Press, 2012.

[77] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf. The evolution from single to pervasive smart cameras. In *Proceedings of the Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2008)*, pages 1–10. IEEE Computer Society Press, 2008.

[78] B. Rinner and W. Wolf. Introduction to Distributed Smart Cameras. *Proceedings of the IEEE*, 96(10):1565–1575, 2008.

[79] D. Rojković, T. Crnic, and I. Cavrak. Agent-based Topology Control for Wireless Sensor Network Applications. In *Proceedings of the MIPRO, 2012 - 35th International Convention*, pages 277–282. MIPRO, 2012.

[80] K. Römer, C. Frank, P. J. Marrón, and C. Becker. Generic Role Assignment for Wireless Sensor Networks. In *Proceedings of the 11th ACM SIGOPS European Workshop*, pages 1–6. ACM Press, 2004.

[81] N. Salazar, J. A. Rodriguez-Aguilar, and J. L. Arcos. Self-Configuring Sensors for Uncharted Environments. In *Proceedings of the Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2010)*, pages 134–143. IEEE Computer Society Press, 2010.

[82] B. Song, C. Soto, A. K. Roy-Chowdhury, and J. A. Farrell. Decentralized Camera Network Control Using Game Theory. In *Proceedings of the Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2008)*, pages 1–8. IEEE Computer Society Press, 2008.

[83] P. Stone and M. Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[84] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[85] M. Szymanski. Videoüberwachung in Bayern - Späh-Angriff mit 17.000 Kameras. *Süddeutsche Zeitung GmbH* `http://www.sueddeutsche.de/bayern/videoueberwachung-in-bayern-spaeh-angriff-mit-kameras-1.1610655`, 2013-02-27. Accessed: 2013-10-02.

[86] K. Van Moffaert, T. Brys, A. Chandra, L. Esterle, P. R. Lewis, and A. Nowé. A Novel Adaptive Weight Selection Algorithm for Multi-Objective Multi-Agent Reinforcement Learning. In *Proceedings of the Annual International Joint Conference on Neural Networks (IJCNN)*, pages 1 – 9. IEEE Press, 2014. In press.

[87] J. Vermorel and M. Mohri. Multi-Armed Bandit Algorithms and Empirical Evaluation. In *Proceedings of the European Conference on Machine Learning (ECML 2005)*, pages 437–448. Springer, 2005.

[88] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[89] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and S. Stornetta. Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992.

[90] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Auction Protocols for Decentralized Scheduling. *Games and Economic Behavior*, 35(1—2):271–303, 2001.

[91] L. While, P. Hingston, L. Barone, and S. Huband. A Faster Algorithm for Calculating Hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, 2006.

[92] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics*, 1(6):80–83, 1945.

[93] W. Wolf, B. Ozer, and T. Lv. Smart Cameras as Embedded Systems. *Computer*, 35(9):48–53, 2002.

[94] A. Yilmaz, O. Javed, and M. Shah. Object Tracking: A Survey. *ACM Computing Surveys*, 38(4):1–45, 2006.

[95] C. Zhang and Z. Zhang. A survey of recent advances in face detection. Technical report, Tech. rep., Microsoft Research, 2010.

Appendix

A full example of an XML-base CamSim scenario file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <root>
        <simulation max_x="30.0" max_y="30.0" min_x="-30.0"
            min_y="00.0">
            <cameras>
                <camera ai_algorithm="epics.ai.PassiveAINodeMulti"
                    heading="-180.0" name="Cam_01" range="30.0"
                    viewing_angle="50.0" x="-18.0" y="30.0" comm="0"
                    FP="60.0" TP="80.0"
                    bandit="epics.bandits.UCB1"/>
                <camera ai_algorithm="epics.ai.ActiveAINodeMulti"
                    heading="-180.0" name="Cam_02" range="30.0"
                    viewing_angle="50.0" x="10.0" y="30.0"
                    failing="30.0" comm="0"/>
            </cameras>
            <objects>
                <object features="1.0" heading="90.0" speed="1.0"
                    x="-30.0" y="2.0"
                    move="epics.movement.Brownian"/>
            </objects>
            <events>
                <event timestep="0" participant="object" name="1.0"
                    event="error"/>
                <event timestep="0" participant="camera" name="2.0"
                    event="change" x="5.0" y="20.0"/>
                <event timestep="0" participant="object" name="3.0"
                    event="add" heading="90.0" speed="1.5" x="-30.0"
                    y="2.0"/>
            </events>
            <visiongraph static="false">
                <graphnode name="Cam_01">
                    <neighbour name="Cam_02" />
                </graphnode>
                <graphnode name="Cam_02">
                    <neighbour name="Cam_01" />
                </graphnode>
            </visiongraph>
        </simulation>
    </root>
```

Symbols

Element	Meaning
C	Set of cameras
O	Set of objects
m	Amount of objects
n	Number of cameras
k	Number of objects per camera
i, q	Cameras $\in C$
j	Object $\in O$
O_i	Objects owned by camera i
	$\rightarrow i$ responsible for tracking objects in O_i
c_j	Confidence of object j
v_j	Visibility of object j
$\phi_i(j)$	Decision of camera i on attempt to track object j
$G_v(C, E)$	Vision graph network wide
E	Set of edges / links between cameras
$e_{i,q}$	Edge between camera i and p
τ_{ix}	Weight between i and x
$N_G(i)$	Local neighbourhood graph for camera i
Δ	Amount of pheromone put on a link after
	exchanging tracking responsibilities
ρ	Evaporation rate.
S	Set of strategies.
s	Strategy $\in S$.
α	Regularisation factor for reward function.
γ	Number of strategies.
α	Weight for reward function of bandit solvers
ϵ	Degree of randomness for bandit solver EPSILON-GREEDY
τ	Temperature defining randomness in SOFTMAX
δ	Inverted Euclidean distance between an object and a camera.
ψ	Normalised angular position of an object to a camera.