Improved Adaptivity and Robustness in Decentralised Multi-Camera Networks

Lukas Esterle and Bernhard Rinner
Lakeside Labs, Institute of Networked and Embedded Systems
Alpen-Adria Universität Klagenfurt
Klagenfurt, Austria
Email: {Lukas.Esterle – Bernhard.Rinner}@aau.at

Peter R. Lewis and Xin Yao
CERCIA, School of Computer Science
University of Birmingham
Birmingham, UK
Email: {P.R.Lewis – X.Yao}@cs.bham.ac.uk

***Abstract*—In this paper we present increased adaptivity and robustness in distributed object tracking by multi-camera networks using a socio-economic mechanism for learning the vision graph. To build-up the vision graph autonomously within a distributed smart-camera network, we use an ant-colony inspired mechanism, which exchanges responsibility for tracking objects using Vickrey auctions. Employing the learnt vision graph allows the system to optimise its communication continuously. Since distributed smart camera networks are prone to uncertainties in individual cameras, such as failures or changes in extrinsic parameters, the vision graph should be sufficiently robust and adaptable during runtime to enable seamless tracking and optimised communication. To better reflect real smart-camera platforms and networks, we consider that communication and handover are not instantaneous, and that cameras may be added, removed or their properties changed during runtime. Using our dynamic socio-economic approach, the network is able to continue tracking objects well, despite all these uncertainties, and in some cases even with improved performance. This demonstrates the adaptivity and robustness of our approach.**

I. Introduction

The latest advances in video technology and cameras becoming affordable, allow the deployment of surveillance systems in new locations. Having a large network of cameras to observe a certain area raises several interesting problems. A person responsible for tracking moving objects in such a large network must be familiar with the topology of the network as well as the environment in which the cameras are embedded. In automated multi-camera tracking, knowledge of the topology is essential to provide good tracking results while not consuming too much resource. Different approaches have been proposed to provide the topology of the camera network, allow a seamless handover, and hence to assist the observer. This vision graph is usually defined offline either manually or by calibrating the cameras (semi-)automatically. The majority of the previously proposed approaches use a central server entity to calculate the vision graph in order to facilitate the handover.

When defining the vision graph, and hence the neighbourhood relations, offline or by means of a central server, three major problems arise. Firstly there is a bottleneck when communicating with the server. This gets more relevant when there is higher number of cameras. Secondly the approaches are not robust to changes within the network. Not only does each camera rely on all of their neighbouring cameras to be fully functioning, but also there is the assumption that other cameras do not change their position or angle. Thirdly, the central server is a weak point in the network, and its failure will cause the entire system to be paralysed.

This paper extends our previous work, in which we introduced a socio-economic approach to selecting cameras to track objects within a decentralised network of smart cameras [1]. The scenarios tested in our previous work contained several simplifying assumptions, which were not reflective of the real world. The main contribution of this paper is to demonstrate how the previous approach can be improved substantially, especially in terms of adaptivity and robustness, when two very 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 setups.

In this context, we show how the system can automatically deal with cameras failing as well as being added to the system during runtime and still maintain high performance. Furthermore we will show that the system can cope well with a camera changing its extrinsic parameters while the system is running.

The rest of the paper is structured as follows. Section II discusses previous work related to multi-camera tracking and the generation of vision graphs. Section III briefly recapitulates our approach to select a camera for continuous tracking in a distributed fashion, employing a local utility function to support the decision process and hence generate the vision graph. Furthermore, the relaxation of different assumptions as well as the impact on our algorithm will be explained in this section. In sections IV and V we describe our new experimental setup as well as our conducted experiments and summarise our key findings. The final section discusses the implications of this work and identifies future research areas.

II. Background

Camera networks usually have to deal with very limited resources. On one hand, this is because the hardware itself only provides limited resources such as CPU and memory, and on the other hand because object tracking is computationally expensive and therefore might need a lot of computational

resources. Hence, we do not want to "waste" resources by tracking the same object with multiple cameras. It is therefore important to decide which camera should track each object at a given time.

In this context, detecting and seamlessly tracking objects in a network of cameras requires cooperation among the different cameras. Selecting the next camera to continue tracking of an object and transferring this responsibility as soon as the object leaves the field of view (FOV) of the current camera, is referred to as *handover* [2]. One of the first autonomous handover approaches for object tracking in smart cameras was presented by Quaritsch et al. [3]. This approach relies on a static and *a priori* known vision graph and does not consider dynamics of the camera network. This can be an important issue for the overall performance. Detmold et al. [4] state the importance of the topology of a camera network for higher-level functions such as multi-camera tracking, target following or camera placement optimization. Still the spatial relationships of FOVs of cameras are often defined manually before the network is operational. In recent years, (semi-)automated approaches have been proposed to determine the vision graph (i.e. [5], [6], [7], [8]). These approaches vary in the topology assumptions (e.g., overlapping or non-overlapping FOVs), topology modelling and the extraction of relevant information from individual camera views.

Adding a random camera at a random timestep without incorporating it into the network manually might not gather valuable utility for the entire network. Removing a random camera at a random time can be even worse in case the network topology is not updated, objects could get lost for entire sub-networks. Changing the viewpoint or the position of a camera might make it more valuable to a completely different area of the network, especially when the old, as well as the new neighbours are not notified. While adding a camera to the network might be a minor problem, since it can only increase the overall utility of the network, removing and changing positions and viewpoints do affect the network actively and might occur due to external actions, such as failures or vandalism.

III. Handover

In [1] we proposed a socio-economic inspired handover algorithm, which employs self-interested agents in sealed bid auctions [9] to obtain the rights to track objects. The value of each object is determined by each camera on its own. In our simulation we use the inverted euclidian distance between camera and object as the utility. The trading of an object provides an implicit snapshot of part of the network topology. When a successful trade is completed, pheromone is deposited on the link between the trading cameras. Initially, a broadcasting strategy is used for communication. Over time, pheromone trails are built up, which represent the strength of neighbourhood relationships, an indicator of the likelihood of a future trade with each camera. These pheromone trails may then be exploited in order to optimise the trade-off between tracking performance and communication overhead. The approach is fully decentralised and requires no *a priori* topology information, since this is learnt online.

This dynamic, socio-economic inspired approach is described in detail in [1], where we demonstrated its ability both to learn the vision graph online and to optimise the trade-off between tracking performance and communication overhead. However, we also made several simplifying and unrealistic assumptions about the behaviour of the camera network. In this paper we relax these assumptions 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, we adapt the camera handover algorithm from [1] to more explicitly consider these issues. To deal with non-instantaneous handover and communications, we have introduced step 1c to our algorithm. Furthermore, step 1f has been introduced to deal with camera failures. In 1c we introduced durations of auctions, this allows cameras not having the advertised object within their FOV at the exact same time to participate in the same auction. This facilitates dealing with non-overlapping FOVs. 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.

To calculate the utility obtained by a camera from its set of owned objects, we used the same utility function as described in [1].

As the vision graph is built up dynamically during runtime, we exploit this knowledge by means of one of a number of communication policies, as described in [1]. These policies specify firstly whether the camera advertises objects *actively* at every time step or *passively* when the object is about to leave the FOV and secondly the probability of camera i communicating with camera x. This probability is based on the pheromone level on the relevant link and is then used in step 1a of algorithm 1. These policies are typically able to substantially reduce communication within the network, while incurring only small penalty to the tracking performance.

Besides our socio-economic inspired communication policies, for comparison we also tested a static communication policy, which uses a predefined vision graph. The static communication only advertises objects to its known neighbours with a probability of $P_{static} = 1$. The predefined vision graph represents a vision graph generated manually or semi-automatically before the camera network is online. Furthermore, the predefined vision graph does not employ our ant-inspired approach and therefore does not have the concept of link strength.

IV. Experimental Setup

Since we are interested in performing repeatable experiments to investigate adaptivity and robustness issues, we used a simulation environment with different scripted experimental setups of artificial smart-camera networks. The simulation tool

Algorithm 1 Robust 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 $P(i, x) = 1$ and go to step 1b.**
2) *Vision graph update:* Update τ_{ix} for all x according to equation described in [1].
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.

used is that used to generate experimental results in [1]. In the following subsection, new properties of the simulation tool with particular relevance to this paper will be presented. In the second part of this section, the different experimental scenarios will be described.

A. Simulation Environment

As in our previous description of our simulation environment we keep a constant number of objects within our scenario. Therfore, if not otherwise specified, objects to be tracked can move in a straight line until they reach the border of the environment and bounce back randomly and continue in that direction until another boundary is reached. Again each camera is controlled independently, by an autonomous software agent capable of communicating with other such agents via message passing. At this stage, we assume perfect tracking (i.e. every object within the FOV of a camera is properly detected, identified, and tracked). To calculate the visibility of an object, we used the inverse Euclidean distance between the camera and the object and the simulated position of the object within the FOV of the camera.

We introduced two main extensions to our simulator: (i) the path of objects can be defined within the simulation description and (ii) events can be defined in the simulation description to occur at a certain timestep of the simulation. These events are related to cameras and have three different types: *add*, *remove*, and *change*. These events occur at a certain timestep where *add* relates to adding a camera at a certain position with predefined parameters to the environment, *remove* represents removing an existing camera from the environment and *change* indicates a change of the parameters of an existing camera during runtime respectively.

B. Experimental Scenarios

For our experiments we considered three general scenarios and executed these scenarios with a variety of objects, paths and events. The different scenarios are illustrated in figure 1. For the first and second scenarios we defined paths for the object to traverse along. These paths are illustrated as blue lines. For scenario three, the objects move in a straight line in a random direction. For each scenario we defined different experiments using our events. We performed each experiment with and without a predefined vision graph to show the robustness of our approach.

For our three distinctive scenarios we conducted experiments where we added a camera during runtime, removed an camera from the test environment and changed the extrinsic parameters of a camera. In table I an overview of the conducted experiments is given.

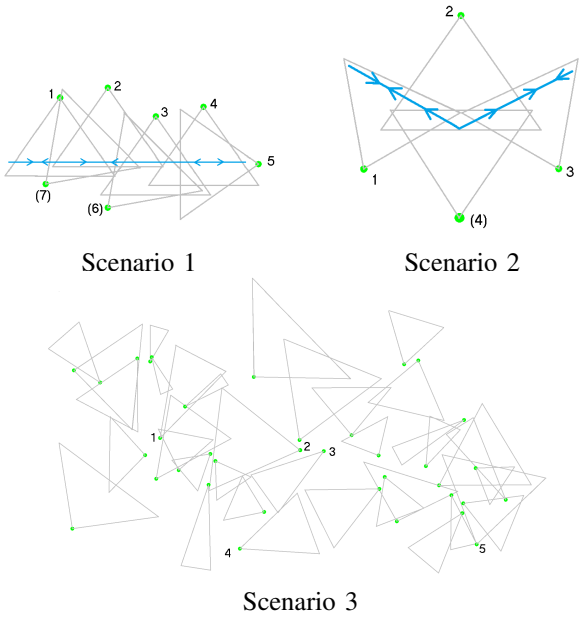

Fig. 1. Three sample experimental setups we used to evaluate our approach. The arrows in scenarios 1 and 2 indicates a path alongside objects traversed during the experimental run. Numbers in brackets indicate cameras not being in the initial network but were added or moved to the illustrated location dynamically.

In our experiments we illustrate the effects of camera-based events using our four different communication approaches: broadcasting, SMOOTH, STEP, as described in [1], and the static communication approach, when applied to both the active and passive schedules. In the case of our dynamically

TABLE I

OVERVIEW OF THE PERFORMED EXPERIMENTS USING THE DIFFERENT SCENARIOS. SCENARIOS 1 AND 2 HAVE FIXED PATHS WHILE IN SCENARIO 3 THE OBJECTS MOVE IN A STRAIGHT PATH UNTIL THEY REACH THE BORDER OF THE SIMULATION ENVIRONMENT FROM WHERE THEY BOUNCE BACK IN A RANDOM DIRECTION. SCENARIOS 1 AND 2 CONTAIN 4 OBJECTS WHILE IN SCENARIO 3 22 OBJECTS ARE MOVING WITHIN THE SIMULATION ENVIRONMENT. FURTHERMORE, ALL PERFORMED ACTIONS OCCURRED IN TIMESTEP 518 WHERE NO OBJECT WAS VISIBLE BY THE AFFECTED CAMERA(S).

Nr.	Scenario	Action
1	Scenario 1	Add Camera (6)
2	Scenario 1	Remove Camera 3
3	Scenario 1	Change Position 3 to (7)
4	Scenario 2	Add Camera (4)
5	Scenario 2	Remove Camera 2
6	Scenario 2	Change Orientation of Camera 2 by -55 degree
7	Scenario 3	Remove Cameras 1, 2, 3, 4, 5

built vision graph, for consistency all parameters were set to be the same as in [1]. For the new aspects of our algorithm, we used $t_a = 3$ and $t_b = 6$ timesteps as the duration for auctions and time-out in case of broadcasting, SMOOTH and STEP communication approaches.

To determine the visibility of an object, we calculate the inverse Euclidean distance between the camera and the object and the simulated position of within the FOV. At this stage, we assume perfect tracking (i.e. every object within the FOV is properly detected and identified).

V. RESULTS

We conducted multiple experiments in each scenario. In each simulation run, the total cumulative utility across all cameras was recorded (the social welfare) as a measure of tracking performance. The number of messages sent between cameras was also measured. We compared the robustness we compare our results with a static communication policy based on an *a priori* known vision graph. For scenario 1 and 2 the *a priori* vision graphs have only been defined for cameras with an overlapping FOV but for scenario 3 non-overlapping FOVs were also considered as long as the FOVs of the cameras were neighbouring each other. Due to a lack of space we cannot report exhaustively on the results from each experiment here, however some key results are highlighted.

The results of experiment 1 are shown in Figure 2 where we added a new camera during runtime. The occurrence of the event is indicated with a red vertical line again at time step 518. The increased accumulated utility using the active SMOOTH and STEP approach is apparent. Since the camera was placed at a location which was already covered by a different camera, the improvement was rather small.

Figure 3 shows results for experiment number 3 employing our active approach. 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. The vertical line shows the time at which the event happened. The drop in utility gain for the static approach after

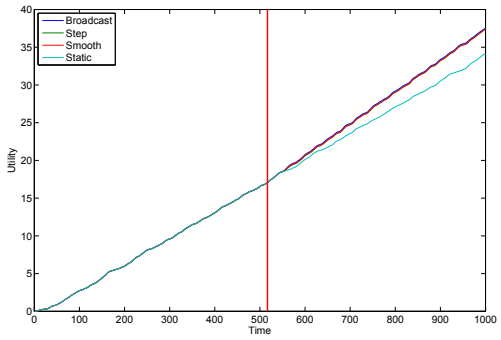

Fig. 2. Cumulative sum of the entire network utility over time for a typical simulation run of experiment 4 comparing our active socio-economic approaches with a static handover. The red vertical line indicates the timestep when the event occured. The simulation ran for 1000 timesteps.

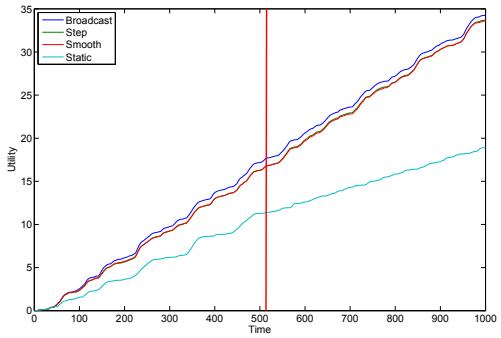

Fig. 3. Cumulative sum of the entire network utility over time for a typical simulation run of experiment 3 (Scenario 1 with change event) and using our passive approaches. The vertical line indicates the timestep when the event occurred. The simulation ran for 1000 timesteps. We changed the position of a single camera within the environment to show the ability of our approach to deal with changes of the extrinsic parameters of cameras.

the event occurred is apparent, demonstrating its inability to adapt to the change. While the static approach loses overall utility, the SMOOTH and STEP policies are able to keep a high utility after the event, indicating their robustness to change.

Figure 4 illustrates the results of scenario 2 with a camera failure event (experiment 5), when passive approaches were used. Here the drop of the accumulated utility is obvious for the static approach, while the socio-economic approaches are able to relearn the vision graph online and continue tracking the object within the entire network.

To compare the adaptivity and robustness of the different variants of the socio-economic 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 shows the performance of the different communication strategies in scenario 1 without any events. The lower illustration plots the results of the same experimental setup but this time having a change event

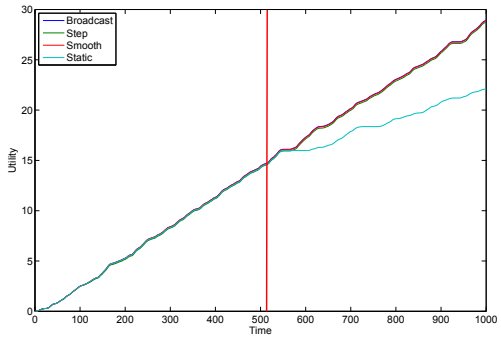

Fig. 4. Cumulative sum of the entire network utility over time for a typical simulation run of Scenario 2 with an error event (experiment 5) and using our active approaches. The red vertical line indicates the timestep when the event occured. The simulation lasted for 1000 timesteps.

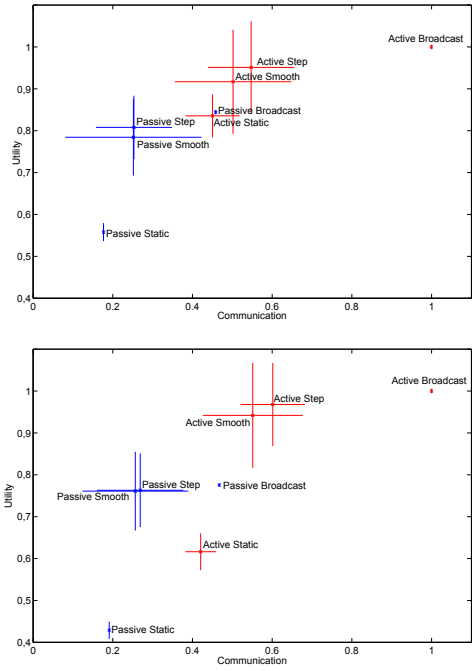

Fig. 5. Overall utility of Scenario 1 calculated over 1000 time steps. Mean and standard deviation have been calculated over 30 runs. The upper graph shows the results for an experiment with no events while the lower graph shows the result for an experiment where the position of a camera was changed after 518 time steps.

after 518 timesteps (experiment 3). While the active approach using static communication performs almost as well as the passive approach in a setting without any changes, the drop of performance for the static approaches is obvious when the change is present; they are not as robust to the change.

In Figure 6 we compare the performance of the different approaches when adding or removing cameras from the network during runtime. All three illustrations show results of scenario 2. The top figure shows the results for an experiment where no events occur, while the middle figure shows the results where a camera has been added to the network during runtime (experiment 4). Even in the presence of a very simple dynamic such as this, the drops in performance of the static approaches are about 10% while the dynamic socio-economic approaches maintain their high performance after the events. The bottom plot in figure 6 illustrates the performance when a camera is removed from the network during runtime (experiment 5). While the utility drops for the static communication approaches by about 20%, the socio-economic approaches are able to maintain a high overall cumulative utility, due to their ability to relearn the changed vision graph online.

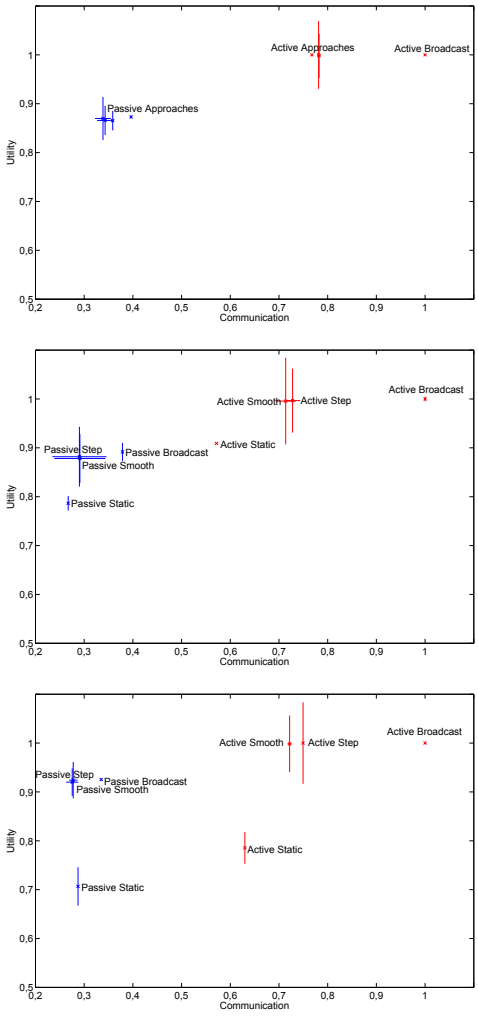

Fig. 6. Overall utility of Scenario 2 calculated over 1000 time steps. Communication and utility are shown on the x and y axes respectively, normalised by the maximum obtained values; *active broadcast* always obtains a utility of 1 with a communication overhead of 1. Mean and standard deviation are 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 7 shows the performance of experiment 7 where we used scenario 3. In the upper graph no events occur but the lower figure illustrates the same scenario when multiple

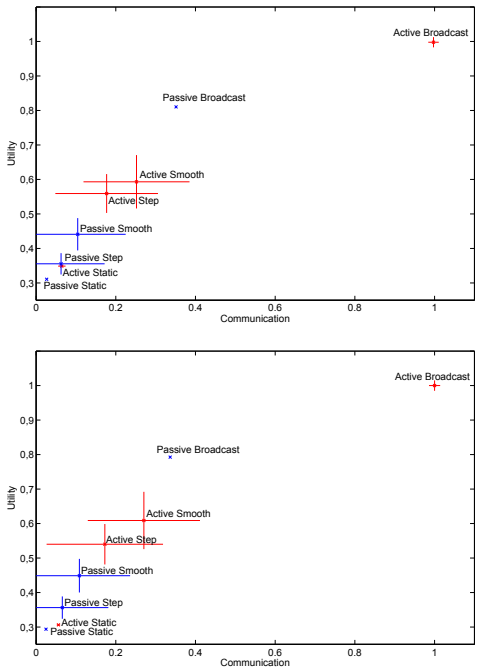

Fig. 7. Overall utility of Scenario 3 calculated over 1000 time steps. Mean and standard deviation have been calculated over 30 runs. The upper graph shows the results for an experiment with no events while the lower graph shows the result for a experiment 7 where multiple cameras were removed after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

cameras fail after 518 timesteps. 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 of the static approaches is rather low, indicating that the relative robustness of the approaches varies with the scenario's properties.

VI. Conclusions and Future Work

In this paper we demonstrated the adaptivity and robustness of distributed smart-camera networks which use an improved version of the socio-economic algorithm presented in [1]. The approach relies on self-interested, autonomous cameras which trade tracking responsibilities for objects using Vickrey auctions.

The main contribution of this paper concentrated on relaxing several simplifying assumptions, which were not reflective of the real world. We showed how to improve the our previous approach substantially, expecially in terms of robustness and adaptivity. Therefor we relaxed the assumptions of instantaneous handover, instantanious communication between cameras, and introduced events to simulate uncertainties in our network. The cameras learn neighbourhood relationships online, clustering into groups to reduce communication within the network. Our improved fully decentralised approach increases the adaptivity and robustness of the camera network when compared to a static approach based based an *a priori* known vision graph, while not requiring a centralised component. 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 after a camera fails.

We showed that different variants of our approach are able to retain or even increase their utility after change events occur. For example, when comparing cumulative network utility 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 approach dropped by 10-20% depending on the event.

There is still a lot of room for future work. A first area is to perform a quantitative analysis of robustness properties on a wider range of scenarios and uncertainties. We would like to understand more deeply the types of uncertainties which can and cannot be dealt with by our algorithm, and how its parameter settings affect this. Another direction is to reconfigure networks based on the knowledge of the vision graph if resources are going to be depleted. Finally, we are currently working on combining long-term reconfiguration algorithms with our novel approach to allow dynamic reconfiguration based on the trading of tracking responsibilities.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement n° 257906.

References

[1] 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," in *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*, Ghent, Belgium, Aug 2011.

[2] 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*, Como, Italy, 2005, pp. 105–110.

[3] 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*, vol. 2007, p. 10, 2007.

[4] 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 ACM/IEEE International Conference on Distributed Smart Cameras*, Vienna, Austria, 2007, pp. 195–202.

[5] Y. Li and B. Bhanu, "Utility-based Camera Assignment in a Video Network: A Game Theoretic Framework," *IEEE Sensors Journal*, vol. 11, no. 3, pp. 676–687, 2011.

[6] T. J. Ellis, D. Makris, and J. K. Black, "Learning a multi-camera topology," in *Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003, pp. 165–171.

[7] Z. Mandel, I. Shimshoni, and D. Keren, "Multi-Camera Topology Recovery from Coherent Motion," in *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*, Vienna, Austria, 2007, pp. 243–250.

[8] 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*, Como, Italy, 2009.

[9] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.