Resource-Aware Configuration in Smart Camera Networks

Bernhard Rinner, Bernhard Dieber, Lukas Esterle
Institute of Networked and Embedded Systems
Alpen-Adria Universität Klagenfurt, 9020 Klagenfurt, Austria
http://pervasive.aau.at

Peter R. Lewis, Xin Yao
School of Computer Science
University of Birmingham, UK
http://www.cercia.ac.uk

Abstract

A recent trend in smart camera networks is that they are able to modify the functionality during runtime to better reflect changes in the observed scenes and in the specified monitoring tasks. In this paper we focus on different configuration methods for such networks. A configuration is given by three components: (i) a description of the camera nodes, (ii) a specification of the area of interest by means of observation points and the associated monitoring activities, and (iii) a description of the analysis tasks. We introduce centralized, distributed and proprioceptive configuration methods and compare their properties and performance.

1. Introduction

Camera networks have been used for wide area monitoring for a very long time. In most of these networks, the cameras act as distributed image sensors that continuously stream video data to a central processing unit, where the video is analyzed by a human operator. More recently, camera nodes have become more capable allowing them to process the captured images locally and to perform in-network data analysis. Networks of smart cameras [12, 11] or visual sensor networks [1, 14] have recently received a lot of interest in academia and industry. An important trend in these camera networks is that they are able to modify the functionality to better reflect changes in the observed scenes and in the specified monitoring tasks.

In this paper we focus on different *configuration methods* for smart camera networks. As a configuration, we consider the analysis tasks that have to be performed on the specific cameras at a specific point in time. Thus, we focus on the *where* and *what* and—since we are mainly interested in constructive configuration methods—*how* captured image data is processed in the network. Basically, we specify a configuration of the camera network by three components: (i) a description of the camera nodes, (ii) a specification of the area of interest by means of observation points and the associated monitoring activities, and (iii) a description of the analysis tasks.

The purpose of this paper is twofold. First, we introduce and formally describe the configuration problem in smart camera networks. Second, we identify the configuration design space, summarize our recent work on centralized, distributed and proprioceptive configuration methods (e.g., [4, 5, 7]) and qualitatively compare these approaches.

The remainder of this paper is organized as follows. Section 2 specifies the configuration problem and briefly discusses the design space of potential configuration algorithms. Sections 3, 4 and 5 describe three different configuration methods and present selected experimental results. Section 6 concludes this paper with a brief comparison and outlook.

2. Configuration in Camera Networks

2.1. Problem Definition

Figure 1(a) sketches the configuration problem [4]. A set of n camera nodes S is placed on a 2D space; the coverage area of each camera is represented by a segment. Each observation point t_j from the set $T = t_1, \ldots, t_m$ has to be covered by at least one camera at a given QoS. The QoS is determined by the frame rate (fps) and the pixel resolution at the observation point, i.e., the pixels on target (pot) of a unit sized object. The covering camera (t_j is within s_i's field of view) has to deliver the monitoring activity a_{t_j} of the observation point while not exceeding the available resources (processing, memory and energy) of the camera.

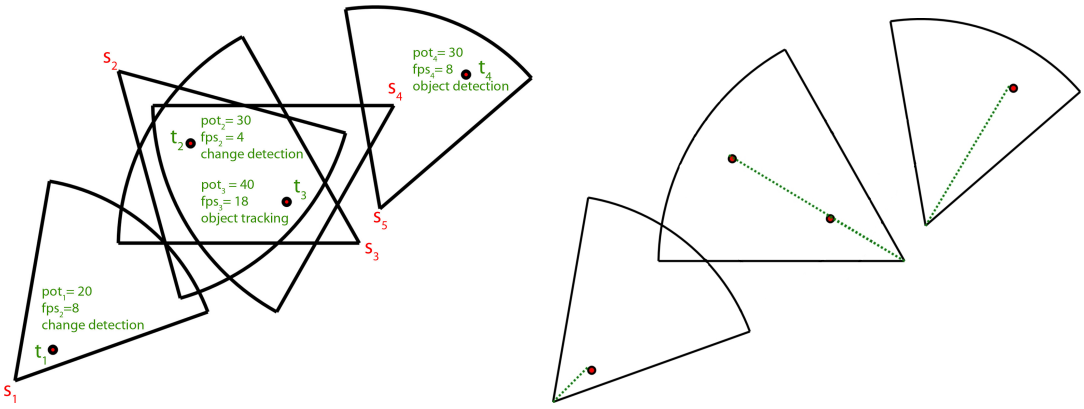
Figure 1(b) depicts a potential solution to the example problem. Three cameras s_1, s_3 and s_5 are sufficient to cover all four observation points and provide the analysis tasks with the requested QoS.

More formally, we consider the sets

$$S = \{s_1, \ldots, s_n\} \quad (1)$$

including all sensors and their associated properties such as position or field of view as well as the set

$$T = \{t_1, \ldots, t_m\} \quad (2)$$



(a) Problem specification (b) Possible solution

Figure 1. (a) A graphical sketch of an example configuration problem defined by five cameras ($S_1 \dots S_5$) and four observation points ($t_1 \dots t_4$). (b) A possible solution where all observation points (targets) are covered by three cameras.

including all observation points and their surveillance QoS requirements "pixels on target", "framerate" and "surveillance activity".

An activity represents a high-level monitoring task which must be achieved at an observation point. Examples for such activities include *image compression and streaming*, *object detection*, and *object tracking*. We define the set of all activities that can be performed as

$$A = \{a_1, \dots, a_l\}\,. \tag{3}$$

The function $\tilde{r}(a_{s_i}, res_{s_i}, fps_{s_i}) \rightarrow (\tilde{c}_{s_i}, \tilde{m}_{s_i}, \tilde{e}_{s_i})$ is used to calculate the required processing $\tilde{c}_{s_i}$, memory $\tilde{m}_{s_i}$ and energy resources $\tilde{e}_{s_i}$ to perform a certain activity a_{s_i} on a node s_i with a specified data input configuration (resolution res_{s_i} and framerate fps_{s_i}). The required resources are specified for processing a single frame.

We search for *feasible* configurations of the complete network. This means that all resource requirements, QoS requirements and activity requirements must be satisfied. Thus, for each sensor, the required memory and processing resources must not exceed the available resources. The required resources for the given input data configuration can be computed using $\tilde{r}$. In order to satisfy the QoS requirements, every observation point must be covered by at least one camera. The sensor must be configured to guarantee the specified pixels on target (pot) and framerate (fps). Finally, the activity constraints must be satisfied. Every observation point must be covered by at least one sensor which must execute the desired monitoring activity for that observation point.

In general, there are multiple feasible configurations possible for a given network configuration problem. Thus, we are interested in configurations which optimize some criteria including network lifetime, energy usage or activity performance. Optimization can be performed using multiple criteria with different objective functions.

2.2. Configuration Design Space

The configuration problem can be instantiated along several dimensions which also span the design space for the algorithms to solve it.

The *dynamics of the environment* represents one dimension, i.e., whether the observation points may change in the monitoring area. Modifications of the observation points may be caused externally (e.g., by the operator) or internally (e.g., by automatic data analysis) to focus attention to different areas. In a static environment, offline configuration would be feasible; a dynamic environment requires a dynamic (re-)configuration.

The second dimension is the *distribution of processing and data* of the configuration method. Centralized methods are typically easier to implement, but have limited scalability. The required computation and communication resources are a related issue here.

The *homogeneity of sensors and tasks* represents another dimension. Sensor nodes can have identical or different capabilities wrt. sensing, processing and communication. This is also true for the observation points which can require homogeneous or heterogeneous monitoring activities.

Finally, the *a priori knowledge* about the network and the monitoring activities has a strong influence on configuration method. Such knowledge include information about the network topology, the sensor calibration and the objects and events to detect. Learning is one approach to overcome limited a priori knowledge.

2.3. Configuration Methods

In this paper we discuss three different configuration methods: centralized, distributed and proprioceptive configuration. These methods have different capabilities and requirements; they represent instances within the previously discussed configuration design space.

In the *centralized approach*, the configuration problem is solved on a dedicated computer node where complete knowledge about the camera network, the observation points and the monitoring activities are available. We apply a genetic algorithm to approximate the configuration problem.

In the *distributed approach*, the individual cameras perform simple operations to find solutions for observation points within their FOV and exchange these local solutions to iteratively improve to overall solution. The distributed approach supports fast re-configuration and is therefore well applicable in dynamic environments. Due to limited access to global data and the simpler optimization method there might be a deviation in the solution quality as compared to the centralized approach.

In the *proprioceptive approach*, the camera nodes act autonomously within the network. They collect and maintain information about their state and progress, which enables them to reason about their behavior (self-awareness) and utilize this knowledge to effectively and autonomously adapt their behavior to changing conditions (self-expression) [8]. We use a socio-economic approach to dynamically configure the camera network without a priori knowledge and demonstrate it in multi-object, multi-camera tracking.

Algorithm 1: Centralized configuration [4].

Algorithm centralized_configuration()
INPUT: S, T, A
OUTPUT: active sensors S' with assignments for D' and $\tilde{P}$
ENCODING: for every sensor s_i its status and input config. $d_i \in D_i$

set initial population
for every epoch **do**
  MUTATE sensor status and input configuration
  EVALUATE coverage
  SELECT
    call task_allocations("covering" solutions)
    perform elitist selection
until termination

Algorithm task_allocation()
INPUT: sensor selections satisfying "coverage"
OUTPUT: feasible solutions with ranking
ENCODING: for every sensor $s_i \in S'$ its assigned procedures $\tilde{p}_i$

set initial population
for every epoch **do**
  MUTATE procedure assignment
  EVALUATE resources and activity
  SELECT
    perform elitist selection
until termination

3. Centralized Configuration

The search space for the configuration problem is typically very large and thus, a combinatorial search strategy becomes infeasible. Since this search problem is also multi-dimensional, the solution is no single point in the search space but a set of Pareto-optimal solutions. A popular approach to tackle multi-dimensional optimization problems is the use of evolutionary algorithms [3] which are inspired by biological processes and apply the "survival of the fittest" principle in an iterative way [16].

The execution of an evolutionary algorithm can be influenced by changing the targeted population size (i.e., the number of individuals present after selection), the mutation rate and the crossover rate (i.e., the number of mutation and crossover operations in one epoch). Evolutionary algorithms make heavy use of random variables (e.g., to select a chromosome to mutate).

3.1. Evolutionary Modeling and Approximation

We approximate the configuration problem in a hierarchical, evolutionary approach [4]. As illustrated in Algorithm 1, the algorithm takes the sets of sensors S, observation points T and activities A as inputs and returns a set of selected sensors $S' \subseteq S$ with assigned sensor configuration D' and procedures $\tilde{P}$.

In the first step, we focus on the coverage problem and search only for sensor selections and input configurations satisfying the coverage requirements. At the end of each epoch, these "covering" solutions are passed over to a second evolutionary algorithm searching for feasible task assignments. This second step focuses on the resource and activity constraints. Thus, the joint output of both steps satisfies all conditions for *feasible* solutions which are ranked according to the specified fitness functions. Although our problem formulation considers scenarios with uncovered observation points (i.e., points which are outside the FOV of all cameras) as infeasible, our algorithm implementation is able to eliminate these points in a preprocessing step and still present solutions for all covered points.

4. Distributed Configuration

The basic idea of the distributed algorithm is that camera nodes autonomously act in a greedy manner to cover observation points (also called targets) in their FOV. They then exchange messages (so called descriptors) describing the required resources to cover an observation point to inform other nodes of their local solution. Improved solutions are identified by comparing the exchanged descriptors. By performing periodic re-evaluation of the assignments (the targets covered by that camera), the solution can be improved iteratively.

In the basic distributed algorithm we do not require information about camera neighborhood; the cameras exchange

descriptors with broadcast messages[1]. Each camera stores the best descriptor for a certain target, be it a local solution or the solution of a remote node. This stored descriptor is broadcast whenever the camera receives a worse descriptor. We support multi-hop dissemination of descriptors using this mechanism. This mechanism also improves the robustness against message loss in unreliable networks.

Algorithm 2 shows a pseudocode description of the distributed algorithm. Nodes react to events such as the occurrence of new targets or the reception of new descriptors[2]. Due to this simple protocol, we can quickly react to changed environmental circumstances and detected events in the monitored area. The distributed algorithm is fast and has good scalability and is therefore feasible for online reconfiguration in dynamic environments [5].

Algorithm 2: Distributed configuration [5].

```
Algorithm distributed_configuration()
  /* executed on every camera node */
On receive new observation point t:
  if t can be covered:
    Calculate required (res, fps, activity)
    Calculate required resources for (res, fps, activity)
    Broadcast descriptor
  fi
On receive descriptor d:
  if better descriptor d_s available (local / stored)
    Broadcast d_s
  else
    Store d as remote best descriptor for t
    Broadcast d
  fi
Do periodically:
  if uncovered target in range
    if it can be covered
    Calculate required (res, fps, activity)
    Calculate required resources (res, fps, activity)
    Broadcast descriptor
    fi
  fi
Do periodically:
  Offer most expensive target that is covered by this node
```

4.1. Comparison of Centralized and Distributed Configuration

In the following we briefly compare the centralized and the distributed configuration algorithm. This comparison is based on the achieved solution quality of both algorithms, i.e., the configuration algorithms optimized the overall energy consumption in four test scenarios. This comparison is adopted from [5] which discusses both algorithms in more detail.

The results of these test series are shown in Table 1. We performed 500 simulation runs per scenario and show the average, minimum and maximum deviation from the reference result of the central algorithm. We also present the average, minimum and maximum number of messages needed to initially cover all points. We further show, how often the algorithm found the optimal result initially and after an re-evaluation phase of 100 message. Note, that even if the optimal result is not found, the result is still valid but requires a higher amount of resources. Additionally we show the average, minimum and maximum deviation from the reference result after the re-evaluation.

It can be seen that the algorithm performs very well in the simple and medium complex scenarios. It is able to find the optimal result for scenario a already in the initial assignment phase. Also, the number of messages needed is very low making the algorithm very suitable for resource-limited networks (especially, if we consider that one message only has a few bytes of size).

We can see that the algorithm found the optimal result for scenario d initially in 4.5% of our test cases but was able to increase this rate after the re-evaluation to nearly 42%. Thus, the algorithm finds results fast and the re-evaluation allows the improvement of the solution already with a small number of additional messages.

To evaluate the performance of the distributed algorithm in unreliable networks, we performed simulations where some broadcast messages were lost randomly. The results for 5% message loss are shown in Table 2. These results are still very close to the case with no message loss. The benefit of a continuous improvement after the initial assignments can also be clearly seen (cf. last column).

5. Proprioceptive Configuration

Recently, tracking applications have been developed on smart camera networks where the processing is distributed among the camera nodes (e.g., [10, 13]). While these distributed approaches apply different configuration strategies for managing the tracking responsibilities, they rely on topology knowledge and/or require iterative information exchange among the cameras. Our proprioceptive approach, first introduced in [7], overcomes these limitations and is able to achieve robust, flexible and scalable multi-camera configuration with low computation and communication overhead.

The tracking handover can be seen as an instance of the configuration problem in smart camera networks. The moving objects' position represent the observation points which are in this case highly dynamic. The monitoring activities (the object tracking) and the camera nodes are typically homogeneous. The goal is to select the best camera for track-

[1] Cameras are neighbors if they have a common observation point in their FOV. If we know the neighbors, we can multicast the descriptors to these nodes and reduce the communication in the network

[2] "On x" indicates the occurrence of event x on the node. Events for new observation points or new descriptors are shown along with optional periodic activities for optimization.

Scenario	Deviation [%]	# Messages	Opt. runs initial - after +100 msgs. [%]	Deviation after +100 msgs. [%]
a	0 - 0 - 0	17.5 - 14 - 21	100 - 100	0 - 0 - 0
b	1.5 - 0 - 36.8	31.2 - 22 - 43	45.1 - 45.5	1.3 - 0 - 12.2
c	5.1 - 0 - 15.5	29.4 - 21 - 43	16.2 - 54.8	3.7 - 0 - 7.8
d	11.9 - 0 - 64.9	54.1 - 44 - 78	4.5 - 41.8	6.7 - 0 - 55.5

Table 1. Comparison of the centralized and distributed configuration algorithm using 4 different scenarios [5]. Average, minimum and maximum deviation in predicted solution quality after the initial assignment (second column) and number of messages of the initial assignment (third column) as well as the ratio of optimal solutions found initially and after re-evaluation (fourth column) are shown. The last column depicts the average, minimum and maximum deviation from the optimal result after an re-evaluation of 100 messages.

Scenario	Deviation [%]	# Messages	Opt. runs initial - after +100 msgs. [%]	Deviation after +100 msgs. [%]
c	21.9 - 0 - 229	27.9 - 21 - 38	16.8 - 57.8	3.3 - 0 - 12.6
d	25 - 0 - 218	51.2 - 38 - 64	2.2 - 45.2	7.1 - 0 - 55.5

Table 2. Comparison of centralized and distributed configuration algorithm including a message loss of 5% [5]. Average, minimum and maximum deviation in predicted solution quality after the initial assignment (second column) and number of messages of the initial assignment (third column) as well as the ratio of optimal solutions found initially and after re-evaluation (fourth column) are shown. The last column depicts the average, minimum and maximum deviation from the optimal result after an re-evaluation of 100 messages.

ing the object at any time.

The proprioceptive approach is based on ideas from both social and economic systems. A market mechanism is used to allocate objects to cameras for tracking. Subsequently, an ant colony inspired mechanism is used to learn the camera network's topology during runtime. As the vision graph is built up over time, it is then used to direct communication within the market mechanism in a more efficient manner. Market principles have long been shown to be useful in the control of distributed computing systems (e.g., [9, 2]) as have ant colony inspired algorithms (e.g., [6]). However, our approach is the first use of artificial pheromones to enable targeted marketing which is completely different from optimization. This provides us with the ability to efficiently manage the trade-off between communication overhead and performance at run-time.

5.1. Utility and Market Mechanism

In the proprioceptive configuration each camera is controlled by a self-interested autonomous agent, with a utility function which it attempts to maximize. A camera i, has a set of *owned* objects O_i. The instantaneous utility of camera i is then given by [7]

$$U_i(O_i, p, r) = \sum_{j \in O_i} [c_j \cdot v_j \cdot \phi_i(j)] - p + r \qquad (4)$$

where $\phi_i : O_i \rightarrow \{0, 1\}$ and is 1 if camera i attempts to track object j and 0 otherwise. c_j and v_j represent the underlying tracking confidence and visibility of object j respectively. Cameras also make payments to each other in exchange for ownership rights over objects. For example, camera b may pay camera s in order to buy the right to track an object owned by s. If both cameras agree, then the object is removed from O_s and added to O_b. In equation 4, p denotes the sum of all payments made by camera i at that time, while r denotes the sum of all payments received.

We use Vickrey auctions [15] as the chosen market mechanism. When attempting to sell an object it owns, a camera hosts a single item Vickrey auction for that object. The advantage of the Vickrey auction is that it has a dominant strategy for bidders: to bid one's truthful valuation, regardless of the strategies of the other bidders. In common with other market-based control systems (e.g., [9]), the currency used is not real money, it is an artificial quantity used for system control.

The basic process which occurs in this approach is therefore as follows. Each camera advertises information to other cameras about the objects it currently owns. Upon seeing such an advertisement, a camera determines the utility it is likely to gain if it owned the object, and had the right to track it. The receiving camera may then bid for the object, privately to the advertising camera. Since we use a Vickrey auction, each camera may place only one bid and the dominant strategy of each camera is to bid a value equal to its truthful valuation of the object in terms of its contribution to the camera's utility (see equation 4).

5.2. Pheromone-based Vision Graph Generation

The market mechanism allocates objects to cameras without requiring any topology information, since each camera can advertise its objects to all other cameras in a broadcast fashion. However, by introducing vision graph information, the process can be made more efficient in terms of communication overhead. This topology information need not be known *a priori*, since the ant colony inspired algorithm is able to learn it online, by observing trading behavior. This is a key advantage of combining social and economic approaches to solve the configuration problem. The fundamental idea is that as a camera learns its neighborhood relations, it may reduce total communication, by advertising

its objects only to those cameras which are likely to buy. At the same time, it can still obtain a high utility, and adapt to changes in the network topology by updating its model of the vision graph during runtime.

As alluded to, the learnt vision graph information is distributed and local information is stored in cameras. We define for each camera i an adjacency list, E_i, the set of all links (or edges) originating at that camera. Each element of E_i is the tuple (i, 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 initialized 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, τ_{ix} is increased by a value Δ. The strength of the links also decreases over time, allowing the system to re-learn in response to changes in topology or cameras' fields of view. The pheromone learning method is described in full in [7].

As τ values are learnt, the inefficient broadcast behavior of cameras can then be made more efficient. Specifically, when advertising an object, camera i sends a message to camera x with probability $P(i, x)$, otherwise it does not communicate with camera x at that time.

We have investigated a number of strategies for determining $P(i, x)$, based on the learnt τ values. These are described in [7], where this socio-economic approach was first introduced.

5.3. Autonomous Camera Control

Combining the camera's utility function, decision process, trading behavior and vision graph generation, we specify that each camera in the system behaves according to Algorithm 3.

As indicated in step 4, the handover algorithm should be run regularly enough 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.

5.4. Experimental Study

To test our approach, we created a 2D simulation framework with static cameras; the cameras' fields of view are modeled as segments. Each camera is independently controlled by an autonomous software agent capable of communicating with other such agents via message passing. In the simulation we assume perfect tracking (i.e., every object within the FOV is properly detected and identified) and calculate the visibility of an object based the inverse Euclidean distance between the camera and the object and the simulated position of within the FOV. In each simulation run, the total cumulative utility across all cameras was recorded (the social welfare) as a measure of tracking performance.

Algorithm 3: Proprioceptive handover algorithm [7].

Algorithm proprioceptive_configuration()
/ executed on every camera node*/*

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 camera's utility 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 camera's utility.
2. *Vision graph update of camera i*: Update τ_{ix} for all x.
3. *Tracking decisions of camera i*: Select which objects in O_i to track in order to maximize $U_i(O_i)$.
4. Repeat at regular intervals.

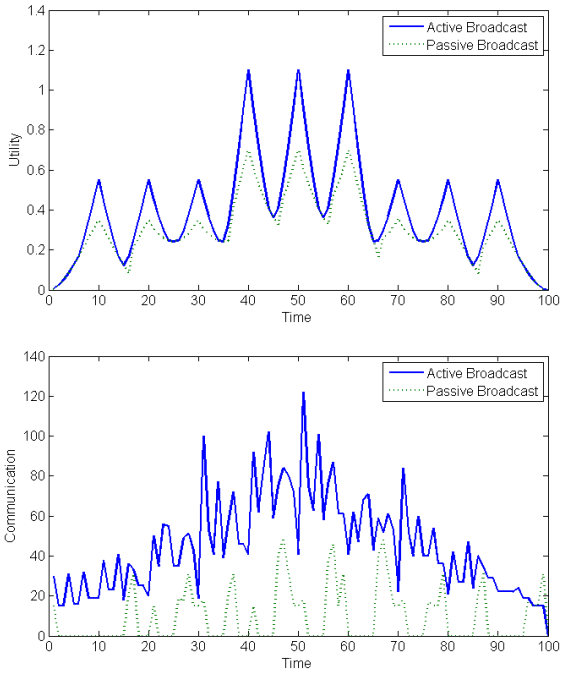


Figure 2. System utility (above) and communication usage (below) over time, during a typical run of a scenario with three objects. Active and passive broadcast algorithms are compared.

The number of messages sent between cameras was also measured.

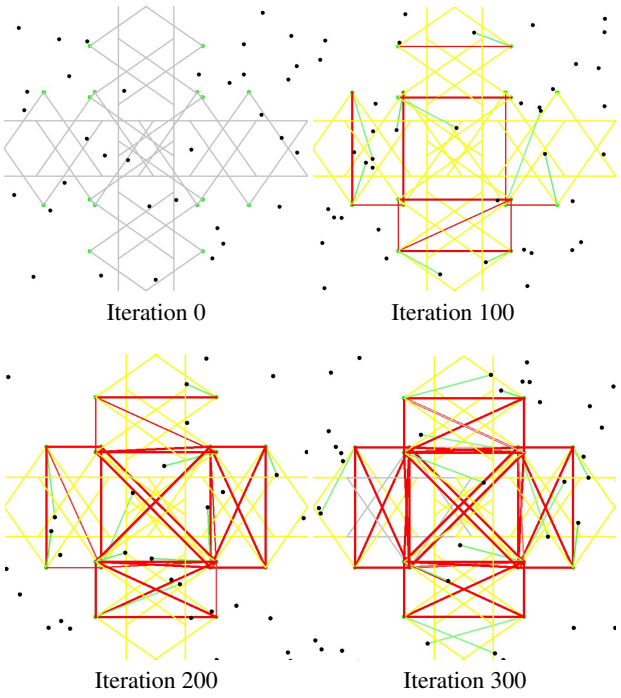


Figure 3. The vision graph is built up through trading interactions, over time, as the system operates [7]. Red lines indicate links in the vision graph; thickness indicates strength. A cameras' FOV turning yellow indicates that it has detected an object.

Initially, two simple broadcast approaches, which we refer to as *active* and *passive*, were tested in the simulation environment. In both approaches, each advertisement message is broadcast 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 a perfect tracking utility since the camera with the highest utility for an object has ownership of it, but on the other hand the communication between the cameras is significant. Contrary to this, the *passive* approach minimizes the communication by sending advertisement messages only when an object is about to leave the FOV of its current owner.

Figure 2 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 scenario with three moving objects. The spikes in utility occur when the objects move into the areas of high visibility in front of each of the cameras. As can be clearly seen, the *active* approach uses significantly more communication.

Figure 3 illustrates the pheromone-based approach to building the vision graph online during runtime. The state of the vision graph is shown at four points through the simulation, from initialization where no adjacency information is known. As the objects are traded between cameras, the links (indicated by thicker red lines) are constructed. Over time, unused links reduce in strength.

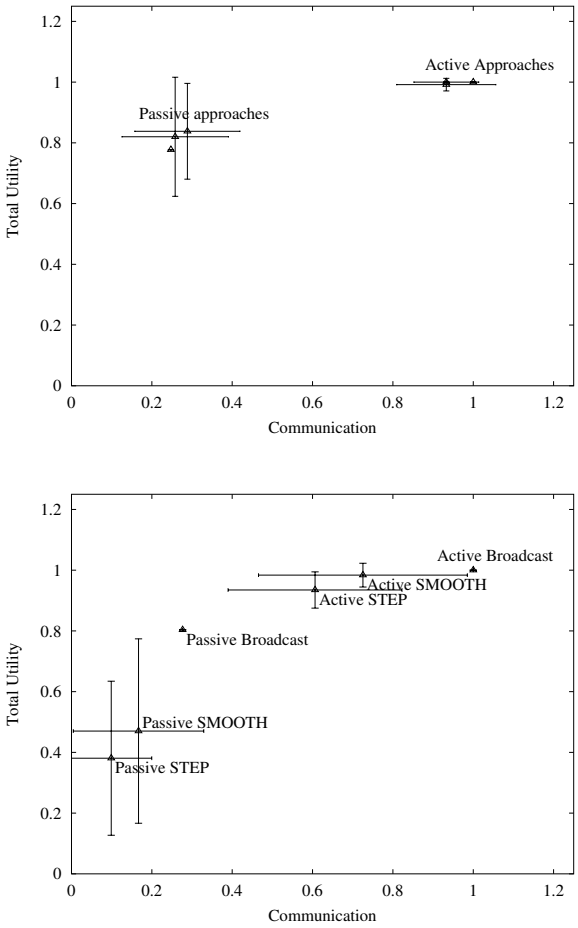


Figure 4. Performance (overall utility calculated across 1000 time steps) of each of the six algorithms on a simple (above) and complex (below) scenario [7]. Both utility and communication values are normalized by those from the active broadcast algorithm.

Figure 4 shows the overall performance of six variants of the approach on two test scenarios, with one object in the environment. Due to the stochastic nature of the object's trajectory and the communication algorithms, mean and standard deviation are shown for each approach, calculated over 30 independent runs. More details on the experimental evaluation can be found in [7].

These results clearly show that the greatest difference between outcomes in the simpler scenario is obtained when switching between *active* and *passive* approaches, while the difference between broadcast and multicast communication schedules has little effect. However, in the more complex scenario, the different approaches yield different outcomes

Property	Centralized	Distributed	Proprioceptive
Environment	static	dynamic	highly dynamic
Processing	centralized	distributed	
Reconfiguration rate	offline	low	high
Configuration goal	global coverage		individual objects
Sensors	cameras, heterogeneous hardware		
Tasks	heterogeneous		homogeneous (tracking)
A-priory knowledge	topology and resources	local resources	none

Table 3. A qualitative comparison of centralized, distributed and proprioceptive configuration algorithms.

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.

6. Discussion

In this paper, we have introduced the configuration problem in smart camera networks and the associated design space for the configuration algorithm. We have compared three different algorithms—centralized, distributed and proprioceptive configuration—which exemplify different parts in the design space. Table 3 highlights the main characteristics of these three approaches. The centralized approach uses an evolutionary algorithm to globally approximate a configuration in a fixed environment with complete knowledge. The distributed approach is able to perform online re-configurations whereas the proprioceptive approach achieves scalable and robust tracking handover in a self-adaptive fashion without relying on any a priori topology knowledge.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement no 257906. This work has also been supported by Lakeside Labs GmbH, Klagenfurt, Austria and funded in part by the European Regional Development Fund (ERDF) and the Carinthian Economic Promotion Fund (KWF) under grant KWF 20214/18354/27107.

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer Networks*, 51:921–960, 2007. 1

[2] S. H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, Singapore, 1996. 5

[3] C. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, Feb 2006. 3

[4] 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*, 21:1424–1437, 2011. 1, 3

[5] B. Dieber and B. Rinner. Distributed Online Reconfiguration of Visual Sensor Networks for Resource-aware Coverage and Task Assignment. Technical report, Klagenfurt University, 2012. 1, 4, 5

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization*. The MIT Press, 2004. 5

[7] 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 (ICDSC)*, pages 1–8, Ghent, Belgium, August 2011. 1, 4, 5, 6, 7

[8] P. R. Lewis, A. Chandra, S. Parsons, E. Robinson, K. Glette, R. Bahsoon, J. Torresen, and X. Yao. A survey of self-awareness and its application in computing systems. In *Proceedings of the 2011 IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*, Oct 2011. 3

[9] 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. 5

[10] Y. Lin and B. Bhanu. A Comparison of Techniques for Camera Selection and Handoff in a Video Network. In *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1 – 8, 2009. 4

[11] C. Micheloni, B. Rinner, and G. L. Foresti. Video Analysis in PTZ Camera Networks - From master-slave to cooperative smart cameras. *IEEE Signal Processing Magazine*, 27(5):78–90, 2010. 1

[12] B. Rinner and W. Wolf. A Bright Future for Distributed Smart Cameras. *Proceedings of the IEEE*, 96(10):1562–1564, October 2008. 1

[13] B. Song, A. T. Kamal, C. Soto, A. K. Roy-Chowdhury, and J. A. Farrell. Tracking and Activity Recognition Through Consensus in Distributed Camera Networks. *IEEE Transactions on Image Processing*, 19(10):2564–2579, 2010. 4

[14] S. Soro and W. Heinzelman. A Survey of Visual Sensor Networks. *Advances in Multimedia*, pages 1–21, 2009. 1

[15] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, 1961. 5

[16] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. *Metaheuristics for Multiobjective Optimisation*, 535(535):21–40, 2004. 3