Resource-Aware Coverage and Task Assignment in Visual Sensor Networks

Bernhard Dieber, *Student Member, IEEE,* Christian Micheloni, *Member, IEEE,* and Bernhard Rinner, *Senior Member, IEEE*

***Abstract*—A visual sensor network (VSN) consists of a large amount of camera nodes which are able to process the captured image data locally and to extract the relevant information. The tight resource limitations in these networks of embedded sensors and processors represent a major challenge for the application development. In this paper we focus on finding optimal VSN configurations which are basically given by (i) the selection of cameras to sufficiently monitor the area of interest, (ii) the setting of the cameras' frame rate and resolution to fulfill the quality of service (QoS) requirements, and (iii) the assignment of processing tasks to cameras to achieve all required monitoring activities. We formally specify this configuration problem and describe an efficient approximation method based on an evolutionary algorithm. We analyze our approximation method on three different scenarios and compare the predicted results with measurements on real implementations on a VSN platform. We finally combine our approximation method with an expectation-maximization algorithm for optimizing the coverage and resource allocation in VSN with pan-tilt-zoom (PTZ) camera nodes.**

***Index Terms*—Visual sensor network; resource allocation; camera coverage; task assignment; evolutionary algorithm**

I. Introduction

Camera networks have been used for security monitoring and surveillance for a very long time. In 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. *Visual sensor networks (VSNs)* consist of camera nodes, which integrate the image sensor, embedded processor, and wireless transceiver [1]. In a visual sensor network a large number of camera nodes form a distributed system, where the cameras are able to process image data locally and to extract relevant information, to collaborate with other cameras on the application-specific task, and to provide the systems user with information-rich descriptions of captured events [2].

VSNs represent networks of embedded sensors and processors with tight resource limitations. However, VSNs have to process large amounts of visual data in real-time and perform rather complex algorithms to fulfill the application requirements. To explore these requirements in some detail, let's have a closer look at a typical monitoring application for VSN. The objective here is to cover (large parts of) the monitoring area with the cameras while performing various monitoring activities. Typical monitoring activities include motion detection, object detection and tracking. Note that these tasks vary in complexity and may change depending on space and time. The selection of cameras and assignment of monitoring tasks to these cameras is therefore a challenging and important problem for VSNs.

In this paper we focus on camera selection and task assignment in VSNs considering the strong resource limitations. We describe this challenge as a *coverage and resource allocation problem* with the objective to find an optimal configuration of the VSN. A configuration is basically given by (i) the selection of cameras to sufficiently monitor the area of interest, (ii) the setting of the cameras' frame rate and resolution to fulfill the quality of service (QoS) requirements, and (iii) the assignment of processing tasks to camera nodes to achieve all required monitoring activities. In order to solve this coverage and resource allocation problem we model the cameras' capabilities and resources, the observation space and monitoring activities as well as the available processing tasks and their resource requirements. We search for approximate solutions using an evolutionary computing approach which provides a good compromise between search time and solution quality. The solutions are evaluated on our embedded camera platforms which will be deployed in a biologically sensitive environment.

This work contributes to the scientific knowledge in at least the following aspects: The first contribution is based on the specification of the VSN configuration as a camera coverage and task assignment problem. To the best of our knowledge, this is the first formulation which jointly considers coverage, QoS and resource allocation in VSNs. The second contribution includes our evolutionary algorithm which efficiently finds good approximations of the coverage and task assignment problem. Further, we combine our approximation method with an expectation-maximization (EM) algorithm. This combination is able to optimize camera coverage and resource allocation in VSNs with pan-tilt-zoom (PTZ) cameras, supports an accurate spatial modeling and achieves a better camera utilization than with static cameras in dynamically changing environments. Finally, the achieved configurations are mapped and executed on our embedded camera platforms which enables a comparison of estimated and measured re-

Manuscript received XXX; revised YYY.

B. Dieber and B. Rinner are with the Institute of Networked and Embedded Systems, Alpen-Adria Universität Klagenfurt, Austria, (see http://pervasive.aau.at/).

C. Micheloni is with the University of Udine, Italy.

This work is 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 and by the Austrian Research Promotion Fund under grant 825840.

source consumption.

The remainder of this paper is organized as follows. Section II discusses related work with regard to resource-limited camera networks as well as sensor placement and selection. Section III introduces the problem formulation in detail. Section IV describes our evolutionary approach for the approximation of the problem and presents our software tool for specifying and solving the VSN coverage and resource allocation problem. Section V discusses the achieved results using different network settings and scenarios for the monitoring activities. An experimental evaluation on camera platforms is also presented. Section VI concludes this paper with a brief summary and discussion about future work.

II. Related Work

A. Resource-limited camera networks

In many multi-camera networks [3], the available resources are limited. These resource limitations are especially prevalent in *distributed smart cameras* [4] and *visual sensor networks* [5], [2]. Different to the traditional camera networks, the processing of the visual data is distributed among the individual camera nodes. A major reason for this distribution of processing is to avoid transferring raw data and hence to support down-scaling the required communication infrastructure [6].

Optimizing the resource assignment in camera networks has recently gained interest in different fields of research. First, computing platforms and sensors are one such field. Due to the advances in semiconductors, sensing and computing performance have increased quite dramatically while reducing the power consumption. Thus, power-efficient camera nodes have emerged recently; examples include WiCa [7], Meerkats [8] and Citric [9] platforms. Although these platforms vary in the available sensing and processing performance, the trend towards smaller, more capable and power-efficient camera nodes can be clearly identified [6].

Second, dynamic resource management tries to change the configuration of the camera nodes and the network during operation to adapt to changes in the required functionality. Maier et al. [10] introduce an online optimization methods for dynamic power management in surveillance camera networks where individual camera components changed their power modes based on the current performance requirements of the network. Winkler et al. [11] present camera nodes combining low and high power radios. Karuppiah et al. [12] describe a resource allocation framework to coordinate distributed object tracking in camera networks. The fundamental block in this framework is the fault containment unit which provides a service using a redundant set of resources. This set of resources supports local fault compensation but also hierarchical resource updates by exchanging fault information. Dynamic resource allocation is also applied to optimize the transfer of (compressed or raw) video data over a camera network. Shiang et al. [13] focus on how multiple cameras should efficiently share the available wireless network resources and transmit their captured information to a central monitor. They compare a centralized approach, a game-theoretic and a greedy approach for making the resource allocation decisions. The objective of resource optimization is to adapt the available resources such as energy, communication bandwidth, service level and sensing capability, such that a given goal function is maximized [2]. Typical goal functions are maximizing the network lifetime and/or the coverage area. Zou et al. [14] evaluates power consumption models for encoding, transmission and recovery of video data in a camera network and optimize for network lifetime. Yu et al. [15] also maximize the network lifetime by optimizing camera selection for coverage and energy allocation to camera nodes. He et al. [16] present a power-rate-distortion model to characterize relationship between power consumption of a video decoder and its rate-distortion performance. Adaptive resource management is also applied to realize camera selection and handoff for multi-camera tracking applications [17], [18].

Middleware systems are yet another method for resource optimization in camera networks [19]. Molla et al. [20] survey recent research on middleware for wireless sensor networks. These middleware systems focus on reliable services for ad-hoc networks and energy awareness [21]. The spectrum ranges from a virtual machine on top of TinyOS, hiding platform and operating system details, to more data-centric middleware approaches for data aggregation (i.e., shared tuplespace) and data query. Agilla [22] and In-Motes [23], for example, use an agent-oriented approach. Agents are used to implement the application logic in a modular and extensible way and agents can migrate from one node to another. Cougar [24] or TinyDB [25] follow the data-centric approach, integrating all nodes of the sensor network into a virtual database system where the data is stored distributed among several nodes.

B. Sensor placement and selection

Placing and selecting sensors is an intensively studied area for wireless sensor networks. The fundamental question is where to place the sensors—or alternatively what sensor to select—such that the area is appropriately covered while keeping the network connected. The area of interest is often described by a set of critical sites (referred to as control points), and each control point has to be covered by at least k sensor nodes. Optimal node placement is a very challenging problem that has been proven to be NP-hard for most of the formulations of sensor deployment [26]. To tackle such complexity, several heuristics have been proposed to find sub-optimal solutions (e.g., [27]). Placement problems are often represented as an integer programming (ILP) model (e.g., [28]).

In contrast to traditional sensor networks which assume omnidirectional sensors, camera networks facilitate directional sensors which introduce additional complexity to the sensor placement problem [29], [30]. Similar to the omnidirectional case, this problem is often described as an ILP [31], and various heuristics have been proposed to find good approximations [32], [33]. The proposed approaches in literature differ in the assumptions about the sensors (homogeneous vs. heterogeneous; fixed vs. mobile), assumptions about the environment (static vs. dynamic), the sensor coverage modeling and the optimization objectives. For example, sensor coverage is often

modeled as simple 2D trapezoids or segments (e.g., [34] [29] and [33]). Cai et al. [35] evaluate algorithms for solving the multiple directional cover sets (MDCS) problems of setting the directions of sensors into a group of nondisjoint cover sets to extend the network lifetime.

In networks comprised of pan-tilt-zoom (PTZ) cameras, the covered area can be actively controlled by changing the cameras' PTZ parameters [36]. This allows to keep "moving" control points within the coverage area—which is important for various tracking applications. However, to steer PTZ cameras appropriately for tracking applications, accurate coverage modeling and efficient optimization are crucial. Such visual coverage models have be described by Mittal et al. [37] and Karuppiah et al. [12]. Examples for efficient algorithms for PTZ configurations are based on expectation-maximization [38], consensus and game theoretic approaches [39], [40], [41].

Typical sensor network problems (e.g., placement, coverage and routing) have also been described as a multi-objective optimization problem [42]. Rajagopalan et al. [43] discuss evolutionary algorithms for wireless sensor networks to solve not only sensor placement problems but also coverage, routing and aggregation optimization tasks. Although the presented related work has some similarities to the presented approach, there are significant differences. First, we consider the coverage and task assignment problem as finding an optimal *VSN configuration* where the in-networking processing (i.e., task assignment) can be changed as well. Second, resource consumption and sensor coverage are modeled corresponding to the different requirements of the (PTZ) camera nodes. Finally, the hierarchical evolutionary algorithm achieves an efficient approximation of a rather complex configuration problem.

III. Problem Formulation

A. Overview and assumptions

As discussed in Section II research has just recently focused on resource-limited visual sensor networks. A fundamental problem here is to determine an optimal network configuration while satisfying various functional and resource requirements. In contrast to typical optimization problems, we focus here on a combined *sensor selection and resource allocation problem*. The objective is (i) to select a subset of cameras which are able to sufficiently monitor the area of interest, (ii) to set the sensors' frame rate and resolution appropriately and (iii) to assign the necessary monitoring procedures to the camera nodes. This *configuration* of the camera network has to satisfy the resource constraints of all camera nodes and the monitoring constraints of all observation points.

The considered network configuration problem is depicted in Figure 1. A set of n camera sensors 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 (s_i covering t_j) has to deliver the monitoring activity a_{t_j} of the observation point, i.e., a set of image processing procedures

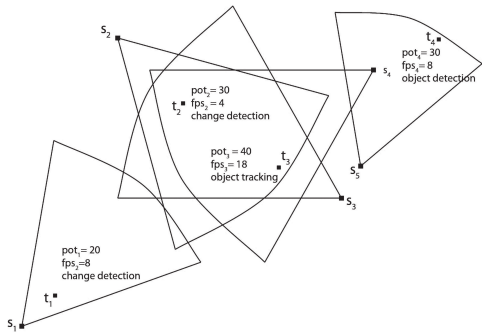


Fig. 1. A graphical sketch of a simple sensor selection and resource allocation problem. Five cameras $s_1, \ldots, s_5$ with fixed FOV (red segments) are placed on the monitoring area to cover four observation points $t_1, \ldots, t_4$. The objective is to find a network configuration, i.e., the set cameras which are required to cover all observation points and the assignment of all necessary image processing procedures to the covering cameras, which satisfies all resource requirements and optimizes some target function. Potential optimization criteria include minimizing global energy usage or maximizing overall network lifetime.

$\tilde{P}_{s_i}$ must be executed at s_i while not exceeding the available resources (processing, memory and energy) of the camera.

A potential solution to the configuration problem depicted in Figure 1 is the selection of cameras s_1, s_3 and s_5. Since t_1 is covered by s_1, this camera must be configured to achieve at least 8 fps and 20 pot and executes a change detection procedure. t_2 and t_3 are covered by s_3; thus, this camera must be configured with at least 18 fps and the resolution to achieve at least 30 pot at t_2 and 40 pot at t_3. Change detection and object tracking procedures must be executed on s_3. Finally, s_5 covers t_4; this camera must be configured to achieve at least 8 fps and 30 pot and execute object detection procedures.

For modeling the network configuration problem we make the following assumptions[1]:

- The camera network consists of directional sensors with a fixed position and fixed field of view (FOV). The frame rate and the resolution of the image sensor can be changed within an a-priori known set of sensor configurations.
- Each camera is able to capture images (at the defined resolution and frame rate), to execute a sequence of image processing procedures and to transfer data/results to other camera nodes in the network. This data transfer is realized in a simple peer-to-peer manner. Complete communication coverage among the nodes and a potential base station is assumed.
- The observation points are static locations in the monitoring area which must be covered by at least one camera's FOV at sufficient resolution, i.e., pixels on target. The pixels on target are determined by the sensor resolution and the distance between camera and observation point.
- We currently only consider convex 2D space without obstacles restricting the camera's FOV.

[1] In Section IV we describe an extension of our network configuration approach to optimize PTZ camera configurations. Naturally, we are able to relax some of these assumptions for this extension, i.e., fixed FOV and 2D space modeling.

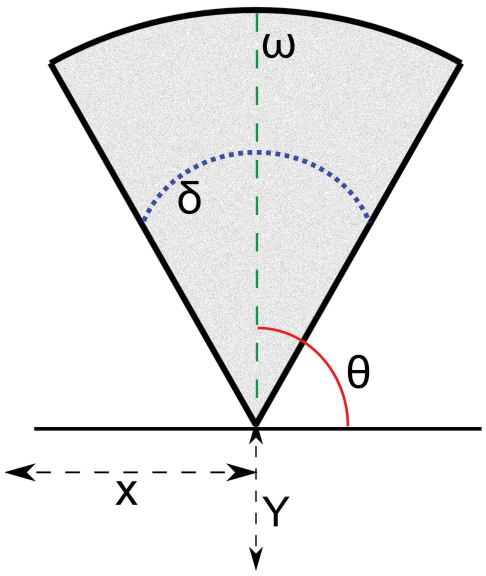


Fig. 2. The 2D model of the camera's field of view (gray area) defined by covering angle δ (blue), covering distance ω (green) and orientation θ (red). The location of the sensor is depicted by x and y.

B. Problem definition

We consider a set of n camera sensors

$$S = \{s_1, \ldots, s_n\}$$

where for each sensor s_i we know its geographical position (x_{s_i}, y_{s_i}), its available resources $r_{s_i} = (c_{s_i}, m_{s_i}, e_{s_i})$ describing processing, memory and energy resources and its l possible data input configurations $D_{s_i} = \{d_{s_i1}, \ldots, d_{s_il}\}$ where d_{s_ij} is a tuple (res_{s_ij}, fps_{s_ij}) representing a certain resolution and the frame rate of the image sensor. $D = \{D_{s_1}, \ldots, D_{s_n}\}$ represents the set of input configurations for all cameras. Furthermore, the orientation θ_{s_i} of the camera and its field of view—expressed by the covering angle δ_{s_i} and the covering distance ω_{s_i}—are known. This 2D model is illustrated in Figure 2. The parameters θ_{s_i}, δ_{s_i} and ω_{s_i} are computed on the basis of the real value of the height, pan angle, tilt angle and focal length assigned to each camera by the EM-based coverage computation.

Further, we define the set of m observation points as

$$T = \{t_1, \ldots, t_m\}$$

where for each t_i we know its geographical position (x_{t_i}, y_{t_i}), the monitoring activity $a_{t_i} \in A$ (where A is the set of all monitoring activities that the sensors are capable of) as well as the required QoS expressed as pixels on target pot_{t_i} and frame rate fps_{t_i}.

An activity represents a high-level monitoring task which must be achieved at an observation point. Examples for such activities are *image compression and streaming*, *change detection*, *object detection*, *person counting* and *object tracking*. These activities are realized by executing some image processing procedures at the covering cameras. In general, there exist many different combinations of single image processing procedures which realize a certain activity. For example, in order to achieve *object tracking*, we can combine a background subtraction procedure (e.g., mixture of Gaussian or frame differencing), an object detection procedure (e.g., connected components) with a tracking algorithm (e.g., CamShift or KLT tracking). Different combinations of these procedures achieve the desired activity, but naturally impose different resource requirements.

For each activity $a \in A$ we define the set $P_a = \{p_{a_1}, \ldots, p_{a_p}\}$ representing alternative procedures for achieving a. Thus, each alternative $p_{a_i} \in P$ is a set of procedures $p_{a_{i_1}}, \ldots, p_{a_{i_b}}$, and the execution of all these procedures is necessary to achieve a. A camera can perform multiple activities simultaneously, and for each activity a_i covered by that camera we must select an appropriate p_k from P_{a_i}. The set of sets $\tilde{P}_{s_j}$ contains all p_{a_i} assigned to s_j . If $\tilde{P}_{s_i} = \emptyset$ no image procedure is assigned to s_i, and this camera can be switched off to save resources.

The function $\tilde{r}(\tilde{P}_{s_i}, d_{s_i}) \rightarrow (\tilde{c}_{s_i}, \tilde{m}_{s_i}, \tilde{e}_{s_i})$ specifies the required processing, memory and energy resources for the individual procedures for a specific data input configuration. The required resources are specified on a single frame basis.

To compute the pixels on target for a certain t_i we use the function $f(D \times T)$ which is based on our simple 2D geographical model. The pixels on target can be calculated by using the angular size of a unit sized object at the given distance $dist_{i,j} = \sqrt{(x_{s_i} - x_{t_j})^2 + (y_{s_i} - y_{t_j})^2}$ between camera s_i and observation point t_j.

By taking into account the camera's resolution res_i and the camera's covering angle δ_i, we can estimate the 1D resolution at the target, i.e., the pixels on target of a unit sized object of $1m$. This simple estimation is based on the ratio between the angular size of the target within the camera's FOV (which can be approximated by $\frac{360}{2\pi \cdot dist_{i,j}}$) and the covering angle δ_i.

$$f(d_i, t_j) = \begin{cases} res_i \cdot \frac{360}{2\pi \cdot dist_{i,j}} \cdot \frac{1}{\delta_i} & \text{if } s_i \text{ is covering } t_j. \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

C. Feasible configuration

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 s_i, the required memory and processing resources of all assigned procedures $\tilde{P}_{s_i}$ must not exceed the available resources. The required resources for the given input data configuration can be computed by $\tilde{r}(\tilde{P}_{s_i}, d_{s_i})$. Thus, the following condition must hold:

$$\forall s \in S : \tilde{c}_s \leq c_s \wedge \tilde{m}_s \leq m_s \quad (2)$$

In order to satisfy the QoS requirements, every observation point must be covered by at least one sensor. This point must be within the field of view of the camera. The sensor must be configured to guarantee a certain number of pixels on target:

$$\forall t \in T \exists s_i \wedge \exists d_{s_i} : f_i(d_{s_i}, t) \geq pot_t \wedge fps_d \geq fps_t \quad (3)$$

where l represents the number of sensor configurations.

Finally, to satisfy the activity constraints, every observation point must be covered by at least one sensor which must execute the set of image processing procedures that achieve the desired activity for that observation point:

$$\forall t \in T \exists s_j \wedge \exists p \in \tilde{P}_{s_j} : p \in P_{a_t} \tag{4}$$

D. Optimization criteria

In general, there are multiple feasible configurations possible for a given network configuration problem. Thus, we are interested in configurations which optimize some criteria. This optimization can be performed in multiple optimization criteria and with different optimization objectives. In this paper, we are focusing on three different criteria: (i) quality, expressed as pixels on target, frame rate and surveillance activity; (ii) energy usage; and (iii) processed data volume. Naturally, different criteria can be defined as well.

Since $\tilde{r}$ calculates the resource usage for processing a single frame, we define the remaining lifetime of a node using the required and available energy as well as the frame rate:

$$L_{s_i} = \frac{e_{s_i}}{\tilde{e}_{s_i} \cdot fps_{s_i}}$$

In terms of energy usage, the optimization can follow different criteria. Examples criteria include:

- Minimum global energy usage:

$$min\left(\sum_{i=1}^{n} \tilde{e}_{s_i} \cdot fps_{s_i}\right) \tag{5}$$

- Maximum lifetime for a specific node s_i:

$$max\left(L_{s_i}\right) \tag{6}$$

- Maximum overall network lifetime:

$$max\left(\min\left(L_{s_i}\right)\right) \tag{7}$$

Considering the data volume processed on a node we can minimize the data volume with respect to resolution and frame rate.

$$min\left(\sum_{i=1}^{n} res_{s_i} \cdot fps_{s_i}\right) \tag{8}$$

To express surveillance quality at the task level, we assign a quality rating to the processing procedures using the function $q(\tilde{P})$. This function then maps a set of image processing procedures to a quality ranking. By accumulating all quality values, we achieve a global quality measure for our surveillance tasks.

$$max\left(\sum_{i=1}^{n} q(\tilde{P}_{s_i})\right) \tag{9}$$

IV. Approximation with Evolutionary Algorithm

A. Approach

The search space for the combined coverage and resource allocation 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 [44] which are inspired by biological processes and apply the ”survival of the fittest” principle in an iterative way [45].

In an evolutionary algorithm, one permutation of the problem space variables is called a chromosome or individual. Many chromosomes form a population (the working set of the algorithm). In every iteration (also referred to as epoch) a certain number of chromosomes is altered by the genetic operators mutation and/or crossover. The mutation generates a copy of a single chromosome and alters some variables of the copied chromosome. The crossover recombines parts of two chromosomes to generate a new one.

During the breed of a new generation by mutation and crossover, the population may grow. To keep the population size constant, the fittest chromosomes must be selected at the end of each epoch. The selection strategy is an elementary part of the evolutionary algorithm and consists of two phases: (i) assigning a fitness value to each individual, and (ii) selecting a subset of the population based on the fitness values and the applied selection strategy. If the selected population is smaller than the targeted population size, dominated individuals may be added to the population again to ensure a maximum diversity.

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).

B. Evolutionary modeling and approximation

We approximate the combined camera coverage and task assignment problem in a hierarchical, evolutionary approach (cp. Figure 3). As illustrated in Figure 3, 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 (Equ. 3). 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 (Equ. 2 and 4). 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

Algorithm coverage_and_assignment()
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 (Equ. 3)
  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 (Equ. 2 and 4)
  SELECT
    perform elitist selection
until termination

Fig. 3. Hierarchical evolutionary algorithm for approximating the camera coverage and task assignment problem.

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.

Note, that the camera coverage and task assignment problem can also be approximated by a standard, non-hierarchically evolutionary algorithm. However, our hierarchical approach helps to significantly reduce the number of calls to the fitness functions which are computationally expensive and dominate the overall runtime of the evolutionary algorithm. In the following, we describe both steps of our approach in more detail. Note, that in both algorithm steps, we generate the initial population randomly. To generate random sensor configurations, we randomly select initial resolutions, frame rates and activities from a given set. To generate an initial set of tasks for a certain activity (task assignment), we randomly select initial procedures that fulfill the given activity.

1) Camera coverage and input data configuration: In the first step we try to select the necessary sensors and set the resolution and frame rate such that every observation point is covered appropriately. The optimization goal is to minimize the data volume processed on all camera nodes.

For the genetic encoding, a chromosome is represented by the status (on/off) and the input data configuration d_{s_i} of each sensor s_i. The available input data configurations are represented in the set D. We only apply mutation as genetic operator which simply corresponds to randomly changing the sensors' status and input data configuration. The fitness function is given by Equ. 3. The decision vector generated by the fitness function is defined by two parameters. The first parameter is equal to the number of observation points "covered" by the chromosome, i.e., the number of observation points which have a properly configured camera covering them. The second parameter corresponds to a cost metric referring to the data volume processed at all sensors. It is calculated by the maximum possible data volume of all nodes normalized by the total data volume processed at all nodes.

$$costratio = \frac{\sum_{i=1}^{n} (res_{max_{s_i}} \cdot fps_{max_{s_i}})}{\sum_{i=1}^{n} (res_{s_i} \cdot fps_{s_i})} \tag{10}$$

2) Task assignment: The "covering" solutions at each epoch of the first step serve as input to the second step. Here we try to find a task assignment for every camera such that each activity of the covered observation points can be achieved and the resource requirements of the cameras are fulfilled. The optimization goal is the energy consumption and lifetime as specified in the optimization objectives (Equ. 5, 6, 7, 9, respectively).

A chromosome is represented by the set of assigned procedures $\tilde{p}_i$ to all activated cameras $s_i \in S'$. The potential assignments of procedures are specified in P. In this step we also perform only mutation as genetic operator. Thus, the assignments of procedures to cameras are changed randomly. The fitness function is defined by Equ. 2 and Equ. 4. For each solution we can calculate the processing, memory and energy requirements, i.e., for each feasible assignment $\tilde{P}$ the required resources $\tilde{c}_{s'_i}, \tilde{m}_{s'_i}, \tilde{e}_{s'_i}$ are computed for all cameras $s'_i \in S'$ by applying the function $\tilde{r}$.

The function $\tilde{r}$ can be realized either by a mathematical model of the resource consumptions or by empirical measurements of the resource consumptions on the target hardware. For our algorithm we adhere to the second approach and measured the required resources for all algorithms and input data configurations on the available camera platforms (cp. Figure 12). These resource values are stored in a table. Thus, the resource function $\tilde{r}$ can then be realized as a simple table lookup.

The fitness function for this algorithm checks if the resource requirements are met on all cameras. The first parameter in the decision vector is the total globally achieved quality calculated according to equation 9. The second parameter is computed according to the resource-related optimization goal. For criterion 5, we calculate the global energy usage e_{global} and use $\frac{1}{e_{global}}$ as fitness value. For criterion 6, we calculate the lifetime for node s_i use it as fitness value. For criterion 7, the minimum lifetime is taken as fitness value. This fitness function also calculates the resulting quality according to Equation 9.

3) Elitist selection: For both steps we use an *elitist selection* [45] method which stores the best found chromosome independently of the main population in order to avoid the loss of already found good chromosomes. In every epoch we add chromosomes to the elite which are not dominated by any other element in this elite. Note that chromosomes may remain in the elite. Thus, if the same chromosome is still in the elite in later epochs, we (re-)use the stored task assignment for that chromosome and can avoid the expensive execution of the second step, i.e., a call of the algorithm "task_allocation()".

If a feasible task assignment is found, the chromosome remains in the elite, otherwise it will be removed. Since there may be allocations which represent feasible solutions

to the sensor selection and sensor configuration problem, but for which no feasible task allocation exists, this additional step is necessary. By restricting the use of the task allocation algorithm to only the members in the elite (and not to all members in the population) we need to test only a small subset of the whole population. In fact, this approach guarantees, that only the chromosomes with the best performance will be tested for resource and activity requirements.

C. PTZ optimization

Our approach for sensor selection and resource allocation considers only static cameras. To be able to use it in PTZ scenarios, we combine our approach with the expectation maximization algorithm described in [38]. To solve the problem of the optimal coverage as in [38] a set of *relevance maps*, 2-dimensional discrete functions in the form of $m : \mathbb{N}^2 \mapsto \mathbb{R}$ representing a relevance value for each point of a discrete map of the scene, has been computed. Since, assuming a planar scene, the intersection of the field of views (represented as cones) with the ground plane are ellipses, finding the optimal camera configuration can be reduced to a data fitting problem. Such a problem consists in finding a set of ellipses of the cardinality of the number of cameras that best fits the relevance maps and maximizes their coverage. In order to determine only available solutions, in [38] a space projection is performed. In particular, for each camera a surrounding sphere is defined and the corresponding relevance map is projected into such a sphere. In this way, the problem is transformed in finding for each sphere (camera) a circle whose center specifies the pan and tilt angles of the camera and the diameter its FOV angle. In order, to converge to the optimal solution, the maximization step executed by each camera needs to know the probability that a point in the relevance map is in the FOV of other cameras. Since the relevance maps, to handle the occlusions, can be different from camera to camera, this information can be made available by sharing the relevance maps among the cameras in the network. Then, each camera runs locally the EM-based reconfiguration considering all the cameras. Being a deterministic process, all the cameras will reach the same solution that will represent the optimal configuration to maximize the coverage given the relevance maps.

The result of the PTZ optimization is transformed into a suitable input for our evolutionary algorithm by taking the camera positions, orientations and zoom factors into account. Observation points can be generated from activity maps i.e., an observation points are placed in areas of high activity, but also manually placed by users.

D. Simulation environment

To accelerate the development of algorithms and to minimize change effort, we have implemented a generic framework for evolutionary single- and multi-objective optimization problems[2]. It facilitates reuse of core evolutionary algorithms, encoding of the chromosomes, and the specification of fitness functions for decision vectors of arbitrary lengths (≥ 1). Consequently, our framework is able to perform multi-dimensional approximations.

We implemented our algorithms in a way, that every model used in the calculations (e.g., the camera model and the calculation of QoS parameters) can easily be exchanged with more sophisticated models if necessary.

In order to adapt our framework to a new optimization problem the following tasks need to be performed:

1) Define the model and implement a corresponding chromosome along with suitable mutation and crossover operations
2) Define the fitness function

The framework will then run the evolutionary optimization autonomously. The framework comes with predefined selection single- and multi-criteria strategies. However, the selection strategy can be modified if necessary as well.

The core algorithm of the framework performs mutation, crossover and selection in each epoch. Large parts of the algorithm can automatically be parallelized to achieve higher performance on multi-processor systems. The framework is implemented in C#.Net and is compatible to Mono[3] and can thus be used an various platforms including Windows, Linux and MacOS.

Point	pot	fps	$activity$
t_1	10	8	change detection
t_2	10	4	change detection
t_3	20	18	object tracking
t_4	15	8	object detection

TABLE I
THE QUALITY REQUIREMENTS OF OBSERVATION POINTS 1-4 EXPRESSED AS PIXELS ON TARGET, FRAMES PER SECOND AND ACTIVITY.

V. RESULTS

To evaluate our approach, we performed systematic tests of our algorithm using a simple, a medium and a complex scenario. We evaluate the impact of parameters such as population size, mutation rate and number of epochs on the performance of the evolutionary algorithm. We further study the runtime of our algorithm, explore the tradeoff between surveillance quality and resource utilization and compare the predicted resource of the assigned tasks with the measured resource consumption on the target platform. Finally, we evaluate the integration of the PTZ optimization.

A. Scenarios

1) Simple scenario: Our first scenario is the example setup from Section III (see Figure 1). This simple scenario consistis of five cameras and four observation points on a 100×100 meter area. The requirements of the observation points can be seen in Table I. The algorithms used in the task assignment are shown in Table III (these apply to all scenarios).

For this simple scenario, a single optimal solution for sensor selection and sensor configuration exists (if all processing

[2] http://nemo.codeplex.com

[3] http://www.go-mono.org

Sensor	res	fps	$activity$
s_1	SQCIF	8	change detection
s_2	QCIF	4	change detection
s_3	QVGA	18	object tracking
s_4	off	off	off
s_5	VGA	8	object detection

TABLE II
THE OPTIMAL SOLUTION FOR THE SIMPLE SCENARIO ASSUMING NO PREVIOUSLY ALLOCATED RESOURCES ON THE NODES.

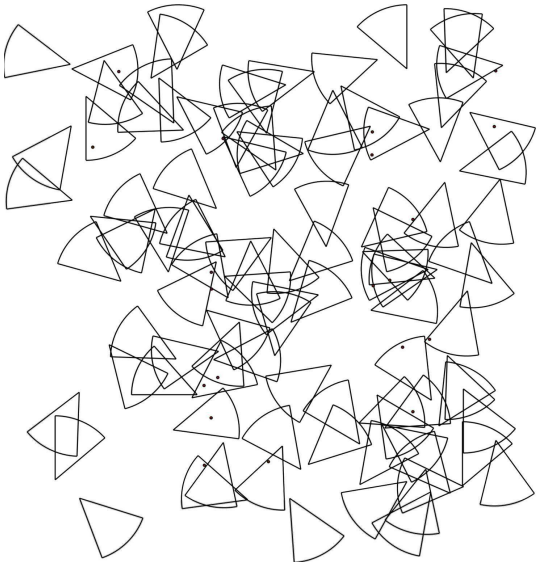


Fig. 4. The randomly scenario with 100 cameras and 20 observation points.

tasks have an equal quality assigned): assuming that all nodes have 100% free resources and at most one activity per sensor, s_4 is switched off and object tracking is assigned to s_3. This is because s_3 is closer to t_3 and can thus cover this observation point at lower resolution. The optimal configuration for this scenario is shown in Table II. The total global data volume is at about 5.6% of the maximum global data volume.

If the amount of free resources of s_3 is reduced to 10% of the available resource, the task of object tracking can only be assigned to s_4 and s_3 should be switched off. We tested multiple such allocations to ensure that the task allocation eliminates solutions which would use more resources than available.

2) Complex scenario: We tested our approach in a more complex scenario of 100 sensors and 20 observation points in an area of 250×250 meters (see Figure 4). The observation points require between 10 and 30 pixels on target.

3) PTZ scenario: We demonstrate the combination of our approach with the PTZ coverage optimization in a medium sized scenario with eight cameras and five observation points on an area of 100×80 meters. Wefirst perform the expectation-maximization algorithm to find the optimal PTZ-parameters of each sensor. Then, we run our evolutionary algorithm to find the optimal sensor configuration and task allocation.

4) Impact of increasing number of solutions: We further evaluate the behavior of our algorithm in scenarios that are within the same level of complexity but which have different

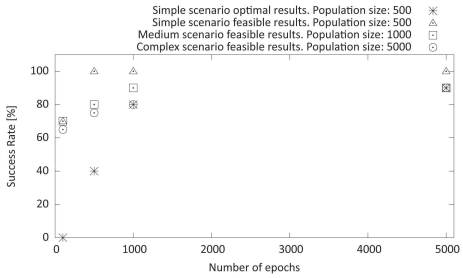


Fig. 6. The relation between number of epochs and the rate of finding optimal and feasible results (success rate), respectively.

number of possible solutions. In a basic scenario of seven observation points we vary the number of covering sensors. Every additional covering sensor increases the number of solutions. This impacts the runtime and the final result. Our basic scenario has five cameras placed such that every point is covered by exactly one camera. We then add cameras to achieve degrees of overlap of two, three and five. Additionally, we constructed two scenarios that have additional non-covering cameras. The scenarios are shown in Figure 5.

For these scenarios we show multi-criteria approximation including the quality ratings of algorithms. We manually assigned a quality rating to each algorithm for our second algorithm stage (see Table III).

Algorithm	Quality
Simple Frame Differencing	0.5
Double Frame Differencing	0.8
Mixture of Gaussians	1
Blobfinder	1
Kalman Tracker	2
CCCR Tracker (openCV)	0.6

TABLE III
QUALITY RATING FOR ALGORITHMS.

B. Evaluation of the evolutionary approximation

1) Number of epochs: Typically, in evolutionary algorithms, the results improve with increasing number of epochs. It can easily be seen that an increased population size will also require increasing the number of epochs to achieve the same results at the same mutation rate.

In Figure 6 we show the change in number of feasible and optimal results with increased number of epochs. By choosing a suitable population size, a predictable rate of feasible results can be achieved. Depending on the complexity of the task, a larger number of epochs may be required.

2) Population size: The population size represents the number of different permutations present at a certain point in time. A larger population size increases the probability that the population contains good individuals.

For our algorithm, it is necessary to increase the population size with increasing complexity of the scenario. As our results show, a population size of 5000 individuals is sufficient to achieve good results even for the complex scenario. For less complex tasks, population sizes of between 100 and 1000 are

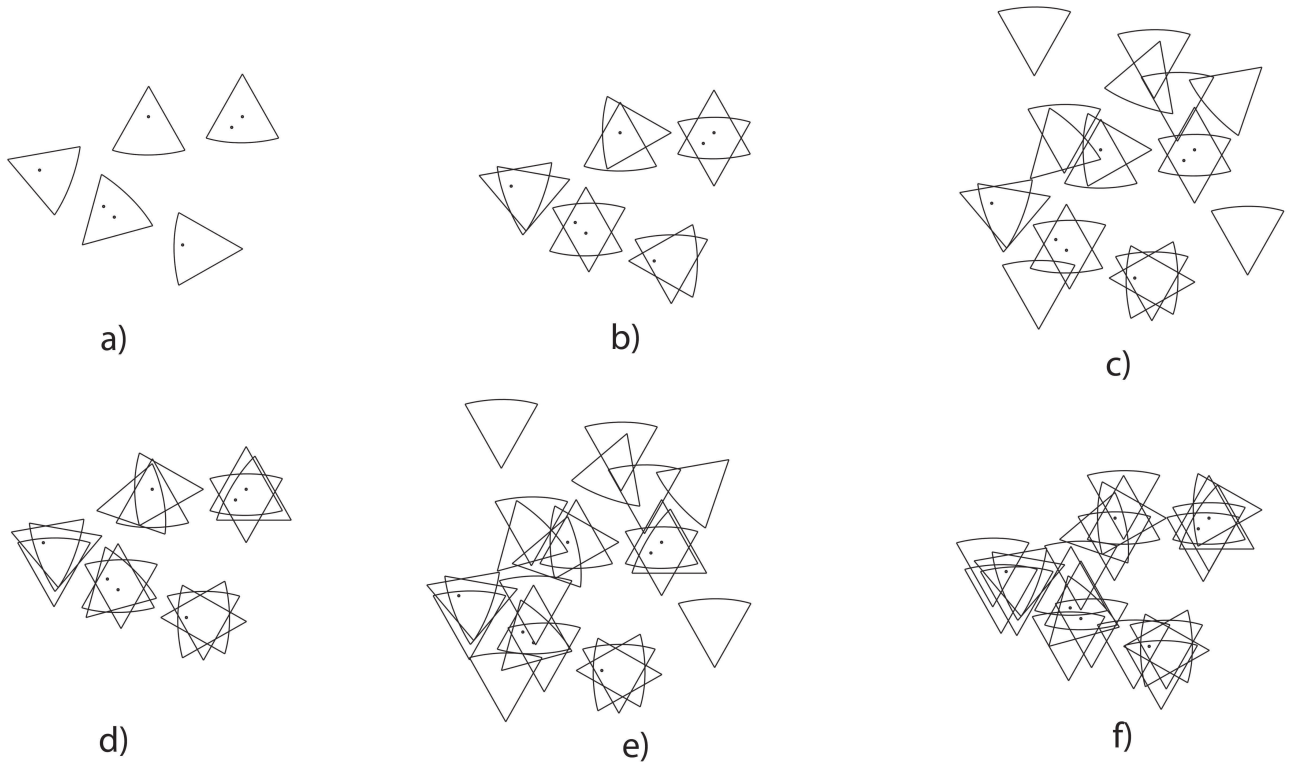


Fig. 5. Scenarios with a degree of overlap of a) one, b) two, c) two with noncovering sensors, d) three, e) three with noncovering sensors and f) five.

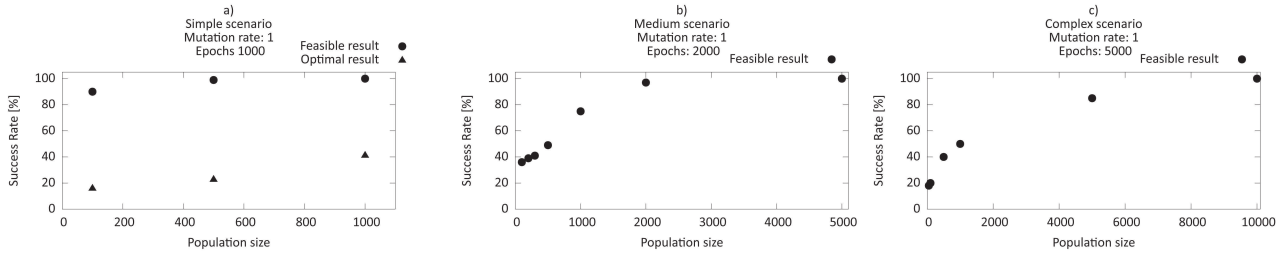


Fig. 7. Relation between population size and the rate of finding feasible results (success rate) for a) Simple b) Medium and c) Complex scenario.

sufficient. ¡ Figure 7 shows the impact of larger population sizes on the resulting rate of finding feasible solutions. It can be seen that the solution quality of complex scenarios can be improved by increasing the population size.

3) Mutation rate: Since the mutation rate determines how many chromosomes are altered per epoch, it greatly influences the number of epochs needed to find good results.

Figure 8 shows the influence of mutation rate on the achieved rate of feasible solutions. It can be seen that larger mutation rates not necessarily yield higher success rates. Thus, the mutation rate must be chosen carefully.

C. Algorithm runtime

Evlolutionary algorithms typically have a very large search space which causes long runtimes. We show that our algorithm has a linear runtime w.r.t. population size (Figure 9a), number of epochs (Figure 9b) and mutation rate (Figure 9c).Note that the runtime values shown are independent of the scenario complexity, i.e., to run 1000 epochs at 0.5 mutation rate and population size 1000 takes the same amount of time for the complex and the simple scenario, respectively.

We performed the tests on a standard PC equipped with an Intel Core2Duo processor with 2.5 GHz. For each scenario we ran at least 1500 test runs at different combinations of mutation rate and population size and took dumps of the algorithm state at certain epochs.

Runtimes for scenarios with increasing degree of FOV overlap are shown in Figure 10. It can be seen that an increasing number of solutions has a small impact on the runtime (whereas this also means that feasible solutions might be found earlier). Increasing the scenario complexity by adding non-covering cameras however, has almost no impact on the runtime.

D. Surveillance Quality

By assigning quality ratings to algorithms, we can explore the tradeoff between surveillance quality and resource utiliza-

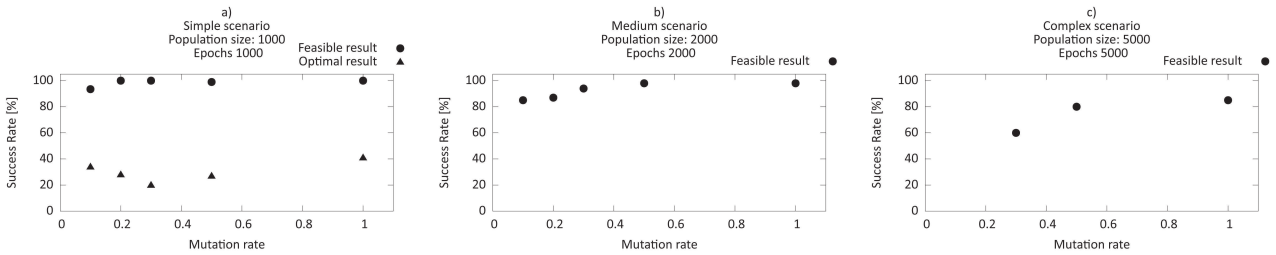


Fig. 8. The relation between mutation rate and the rate of finding feasible results (success rate) for a) Simple b) Medium and c) Complex scenario.

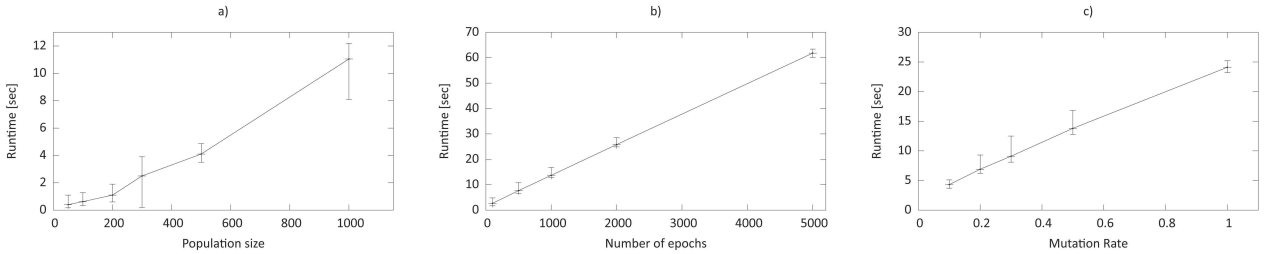
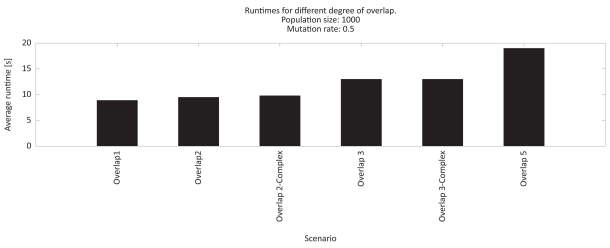


Fig. 9. Runtime with respect to population size, number of epochs and mutation rate for a) Simple b) Medium and c) Complex scenario..



Node	Tasks
1	SingleGaussian
2	FrameDoublediff
3	FrameDoublediff, Blobfinder, Kalman
4	off
5	FrameDoublediff, Blobfinder

TABLE IV
THE RESULT OF THE TASK ALLOCATION FOR THE SIMPLE SCENARIO.

Fig. 10. The runtimes for our overlap scenarios.

tion. Figure 11a shows the Pareto front for the scenario of medium complexity for an elite size of 20 (i.e., we choose 20 non-dominated solutions from the pareto front).

Figure 11b shows an example Pareto front for the scenario with five overlapping cameras per observation point. We used an elite size of 100 for this experiment. This shows that there is a large number of possible solutions that our algorithm is able to find. All those solutions must be regarded as equally good tradeoffs between quality and resource usage. From those, possible solution one has to be selected. This may be done according to a predefined weighting of quality versus resource usage or by any other metric or selection function.

E. Measurements of resource usage

The result of our evolutionary algorithm is a feasible camera configuration and a task allocation along with a prediction of the resource usage. To evaluate the accuracy of the resource prediction, we experimentally tested the resource usage of the assigned tasks on a real platform. Based on the hardware platform used in our tests (see below), we constructed the mapping $\tilde{r}$ from measuring algorithm performance on this hardware. We did this by running the algorithms with videos of different resolutions as inputs while measuring resource and energy usage.

We can then use $\tilde{r}$ as a lookup table in the algorithm to predict the resource usage of a certain combination of algorithms. We implemented the application according to the task assignment in Table IV and measured the CPU load and power usage. For these tests, the application read images from a video file of the calculated resolution and executes the assigned tasks.

We use Atom-based embedded boards as target platforms. We tested all algorithms on pITX-SP 1.6 plus board manufactured by Kontron[4]. The board is shown in Figure 12 and serves as processing platform for our camera nodes. It is equipped with a 1,6 GHz Atom Z530 and 2GB RAM.

As it can be seen from Table V, our predictions match with the measured results.

F. Integration of PTZ configuration

In PTZ scenarios we want to *i)* find feasible configurations and task allocations and *ii)* to select the subset of sensors which is required to cover the observation points. The sensors which are not required, can then be commanded to basic coverage while the other sensors can focus on detection and tracking at the observation points.

[4]http://www.kontron.com

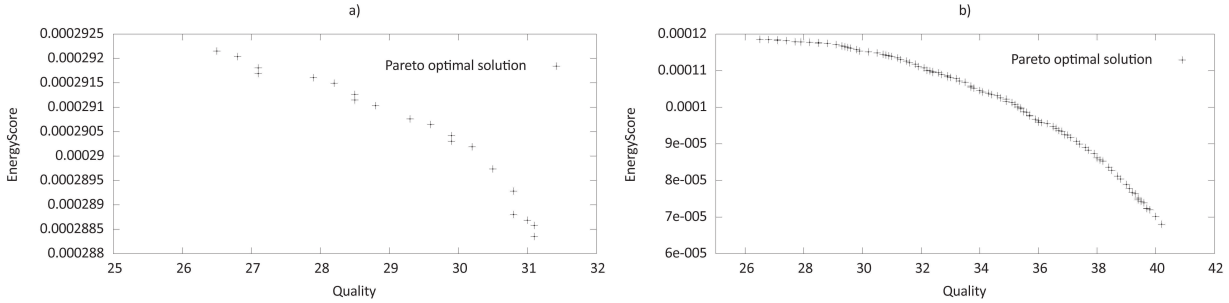


Fig. 11. Pareto fronts for a) medium scneario and b) scenario with 5 overlapping cameras. The resource optimization goal is Global minimum energy usage

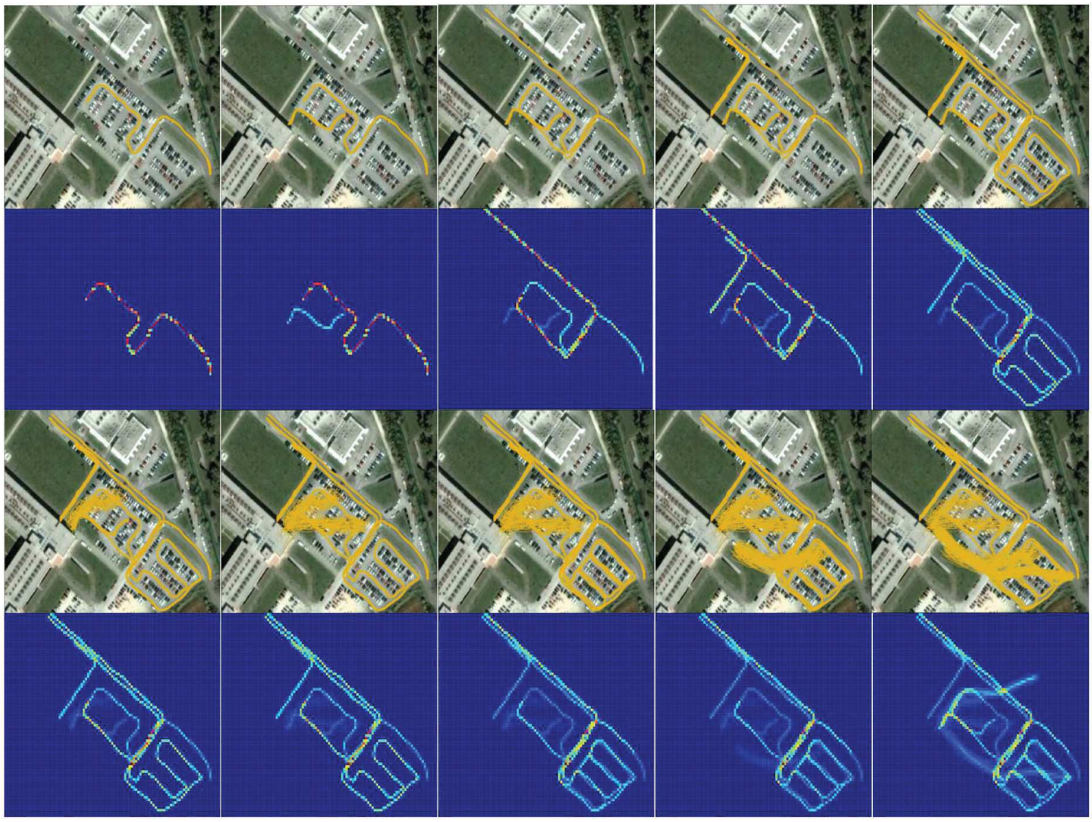


Fig. 14. Scenario evolution. First row shows the evolution of the environment by introducing trajectory clusters TR1, TR2, TR3, TR4, TR5. Second row show the evolution of the activity map as consequence of the evolution of the environment represented in the image above. Third row as for first row but for introducing clusters TR6, TR7, TR8, TR9, TR10. Fourth row as for second row but concerning the evolution presented in the third row.

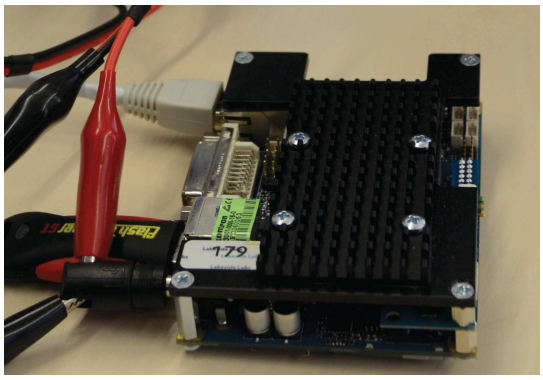


Fig. 12. The pITX-SP hardware platform used in our tests.

Node	CPU_p[%]	$Power_p$[W]	CPU_m[%]	$Power_m$[W]
1	4.32	0.09	3.6	0.1
2	0.34	0.01	0.21	0.01
3	17.55	0.35	16.2	0.3
5	12.28	0.25	11.6	0.2

TABLE V
THE PREDICTED AND MEASURED RESOURCE USAGE FOR NODES IN THE SIMPLE SCENARIO. CPU_p AND $Power_p$ ARE PREDICTED VALUES, CPU_m AND $Power_m$ ARE MEASURED VALUES.

1) PTZ optimization: To test the automatic configuration of the pan, tilt and zoom parameters of the proposed PTZ network a map representing the university area has been selected. On such a map eight different cameras have been deployed to

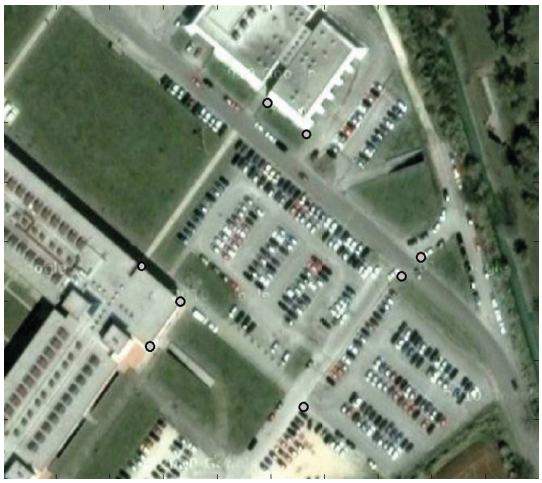


Fig. 13. Deployment of the cameras (black circles) on the monitored environment. Each camera is placed at height 14m.

Scenario Evolution	EM Iterations	Coverage
Empty	174	97, 5%
TR1	67	98, 5%
TR2	93	98, 4%
TR3	83	99%
TR4	46	98.5%
TR5	66	97.9%
TR6	109	98.3%
TR7	64	97.8%
TR8	49	87.7%
TR9	73	99.8%
ALL	98	97.2%

TABLE VI
EM BASED RECONFIGURATION PERFORMANCE. THE TABLE SHOWS THE REQUIRED ITERATIONS TO CONVERGE TO THE OPTIMAL SOLUTION AND THE COVERAGE ACHIEVED BY THE SOLUTION.

cover the entire environment. In Figure 13 a representation of the testbed area together with the deployment configuration of the PTZ network is presented.

Initially, a camera configuration has been achieved by running the EM based network configuration on a homogeneous activity map (e.g., each cell of the map has the same activity density). Then, ten different trajectory clusters have been defined as shown in Figure 14.

As clusters have been added to the scenarios, the EM based reconfiguration has been executed on the new data. Hence, while the scenario evolves by considering new trajectories, thus new activities occurred inside the monitored environment, the PTZ network adapts its parameters to focus on the areas with higher probability of activity. In Table VI, the number of iterations required by each camera to compute its parameters together with the final coverage of the monitored area are presented in relation to the inclusion of the trajectory clusters. It is worth noticing, that the number of iteration is quite low (around 100) and that the coverage of the area is always kept as close as possible to 100%. This means that the reconfiguration can be done distributedly on each camera with lower computational requirements and that the new configuration better fits the activity probability without reducing the area coverage.

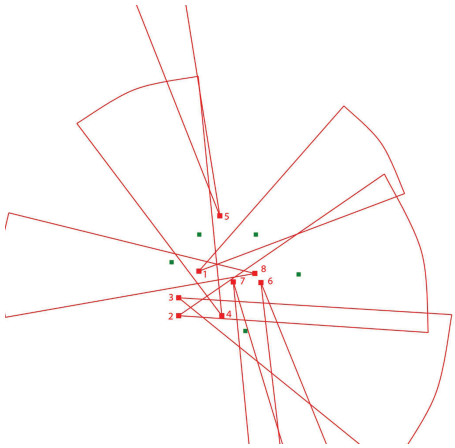


Fig. 16. Input for the sensor selection and resource allocation optimizer.

The PTZ network reconfiguration process on the adopted data achieved the configuration presented in Figure 15. Table VII presents the PTZ parameters for the initial configuration achieved on an empty map and for the final configuration. It is interesting to notice how all the parameters have changed significantly, i.e., the FOV of the cameras for the final configuration is much narrower than that for the initial configuration. The FOVs of the cameras have been narrowed of about 45% on average. This means that the magnification of each camera has been increased thus the resolution with which the objects of interest moving inside the areas of activity is increased as well.

Camera	Pan Initial	Pan Final	Tilt Initial	Tilt Final	FOV Initial	FVV Final
1	35.02	8.868	77.820	80.859	35.516	27.6258
2	-21.28	-22.370	80.170	80.967	44.7594	35.2792
3	14.93	37.427	79.223	82.467	38.4954	38.8806
4	111.50	94.758	78.845	81.499	41.8634	31.0868
5	138.57	148.612	80.868	82.202	34.1896	12.4892
6	-75.38	-133.639	80.619	81.615	37.2104	14.9248
7	-78.34	-97.754	81.414	80.794	45.5406	11.0936
8	178.30	174.446	79.416	82.220	36.8536	24.3778

TABLE VII
PTZ PARAMETERS OF THE CAMERAS AFTER THE INITIALIZATION AND THE LAST CONFIGURATION. THE FIELD OF VIEW (FOV) IS EXPRESSED IN DEGREES AND IT DESCRIBES THE ANGLE OF THE MINIMUM CONIC FIELD OF VIEW THAT INSCRIBES THE REAL CAMERA FIELD OF VIEW.

2) Sensor selection and resource allocation: Taking the result of the PTZ optimization as input, we have run the sensor selection and resource allocation optimizer. In the areas of high activity, we have placed observation points requiring object detection or object tracking. The resulting input for algorithm is shown in Figure 16.

Table VIII shows the results for the sensor selection and sensor configuration. Table IX shows a resulting task allocation. The resulting Pareto front for the task allocation is shown in Figure 11a.

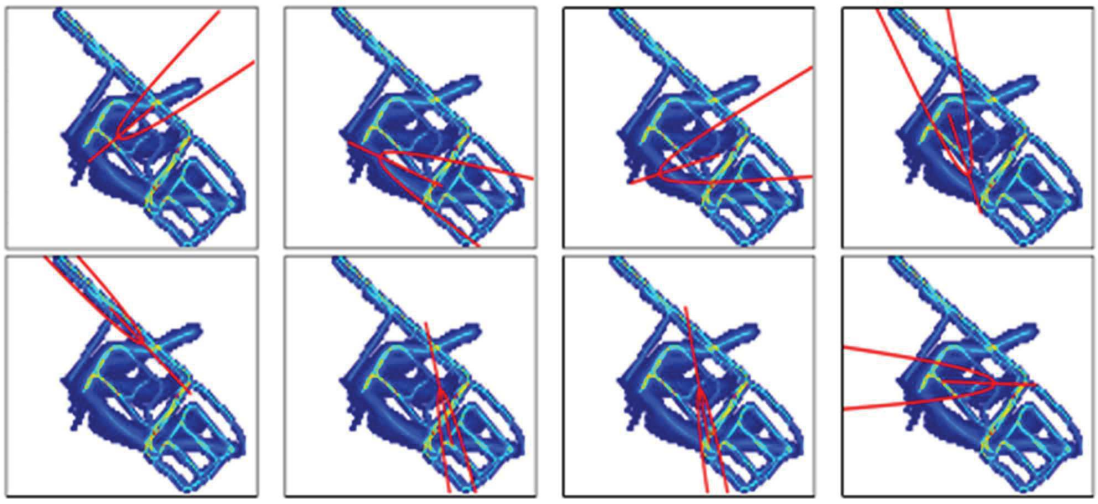


Fig. 15. Configuration of the eight cameras (one to eight from left to right top to bottom) after the EM based reconfiguration algorithm on the final activity map.

Sensor	res	fps	$activitiy$
1	SQCIF	4	object detection
2	VGA	18	object tracking
4	QVGA	4	object detection
7	SQCIF	12.5	object tracking
8	QCIF	2	object detection

TABLE VIII
THE RESULT FOR SENSOR SELECTION AND SENSOR CONFIGURATION. SENSORS 3, 5, 6 ARE OFF.

Sensor	Tasks	CPU [%]	Mem. [MB]	Power [W]
1	FDD, BF	0.24	0.77	0.005
2	FDD, BF, K	70.22	18	1.4
4	FDD, BF	1.65	1.49	0.03
7	FDD, BF, K	2.03	0.73	0.04
8	FDD, BF	1.01	1.49	0.02

TABLE IX
THE RESULTING TASK ALLOCATION. FDD: FRAME DOUBLE DIFF. BF: BLOBFINDER. K: KALMAN. SENSORS 3, 5, 6 ARE OFF.

VI. CONCLUSION

In this paper we have presented a formulation and an approximation method for the camera selection and task assignment problem for visual sensor networks. We have analyzed our approximation method on different scenarios and compared the predicted results with measurements on real implementations on a VSN platform. The tradeoff between surveillance quality and resource utilization has further been demonstrated with our multi-criteria approximation algorithm which achieves a Pareto-front of non-dominating results. We have finally combined our approximation method with an expectation-maximization algorithm for optimizing the coverage and resource allocation in VSN with Pan-Tilt-Zoom (PTZ) camera nodes.

Our results demonstrate that feasible and near-optimal solutions can be found within few seconds even for our complex test scenario. Furthermore, the predicted resource utilization matches very well with the measured resource utilization. For our five camera scenario, the deviation for the CPU utilization was less than 1.3 %, and the deviation for the power consumption was less than 0.05 W. Our PTZ camera scenario shows that our method can be used to find camera configurations for complex monitoring activities, i.e., to select PTZ cameras which can best cover specific observation points and cameras which can cover wider areas. As with our standard method, the combined method also approximates the task assignment for all cameras.

There are several possibilities for improving our approach. Thus, future work includes (i) the improved modeling of the VSN resources such as communication bandwidth and delay, (ii) further evaluations using both manually generated test data and scenarios from actually deployed VSNs, and (iii) the integration in an VSN application. Further, we will present a distributed solution for this problem as part of our future work. It will also be able to dynamically update the network according to changed environmental parameters like moving objects or changed PTZ configurations.

REFERENCES

[1] B. Rinner, M. Quaritsch, W. Schriebl, T. Winkler, and W. Wolf, "The Evolution from Single to Pervasive Smart Cameras," in *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-08)*, Stanford, USA, 2008, pp. 1–10.

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

[3] Hamid Aghajan and Andrea Cavallaro, Ed., *Multi-Camera Networks*. Elsevier, 2009.

[4] B. Rinner and W. Wolf, "Introduction to distributed smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565–1575, October 2008.

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

[6] B. Rinner and W. Wolf, "Toward Pervasive Smart Camera Networks," in *Multi-Camera Networks*, H. Aghajan and A. Cavallaro, Eds. Elsevier, 2009, pp. 483–496.

[7] R. Kleihorst, A. Abbo, B. Schueler, and A. Danilin, "Camera Mote with a High-Performance Parallel Processor for Real-Time Frame-Based Video Processing," in *Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras ICDSC '07*, 25–28 Sept. 2007, pp. 109–116.

[8] C. B. Margi, X. Lu, G. Zhang, R. Manduchi, and K. Obraczka, "Meerkat: A Power-Aware, Self-Managing Wireless Camera Network for Wide Area Monitoring," in *International Workshop on Distributed Smart Cameras (DSC-06)*, 2006.

[9] P. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan, A. Y. Yang, C. Yeo, L.-C. Chang, J. Tygar, and S. S. Sastry, "Citric: A low-bandwidth wireless camera network platform," in *Proceedings of the Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-08)*, Stanford, CA, USA, September 2008, pp. 1 – 10.

[10] A. Maier, B. Rinner, and H. Schwabach, "Online Multi-Criterion Optimization for Dynamic Power-Aware Camera Configuration in Distributed Embedded Surveillance Clusters," in *Proceedings of the IEEE 20th International Conference on Advanced Information Networking and Applications (AINA 06)*, Vienna, Austria, Apr. 2006, pp. 307–312.

[11] T. Winkler and B. Rinner, "Pervasive Smart Camera Networks exploiting heterogeneous wireless Channels," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2009, pp. 296 – 299.

[12] D. R. Karuppiah, R. A. Grupen, Z. Zhu, and A. R. Hanson, "Automatic resource allocation in a distributed camera network," *Machine Vision and Applications*, vol. 21, pp. 517–528, 2010.

[13] H.-P. Shiang and M. van der Schaar, "Information-Constrained Resource Allocation in Multicamera Wireless Surveillance Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 4, pp. 505–517, 2010.

[14] J. Zou, C. Ta, R. Zhang, and H. Xiong, "Modeling and Optimization of Network Lifetime in Wireless Video Sensor Networks," in *Proceedings of the IEEE International Communications Conference*, 2010, pp. 1 – 6.

[15] C. Yu and G. Sharma, "Camera Scheduling and Energy Allocation for Lifetime Maximization in User-Centric Visual Sensor Networks," *IEEE Transactions on Image Processing*, vol. 19, no. 8, pp. 2042–2055, 2010.

[16] Z. He and D. Wu, "Resource Allocation and Performance Analysis of Wireless Video Sensors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, pp. 590–599, 2006.

[17] C.-H. Chen, Y. Yao, D. Page, B. Abidi, A. Koschan, and M. Abidi, "Camera Handoff with Adaptive Resource Management for Multi-Camera Multi-Target Surveillance," in *IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, 2008, pp. 79 – 86.

[18] E. Monari and K. Kroschel, "Task-Oriented Object Tracking in Large Distributed Camera Networks," in *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2010.

[19] B. Rinner and M. Quaritsch, "Embedded middleware for smart camera networks," in *Multi-Camera Networks*, H. Aghajan and A. Cavallaro, Eds. Elsevier, 2009.

[20] M. M. Molla and S. I. Ahamed., "A survey of middleware for sensor network and challenges," in *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*, 2006, pp. 1 – 6 pp.

[21] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Issues in designing middleware for wireless sensor networks," *Network, IEEE*, vol. 18, no. 1, pp. 15–21, 2004.

[22] C.-L. Fok, G. C. Roman, and C. Lu, "Rapid Development and Flexible Deployment of Adaptive Wireless Sensor Network Applications," in *Proceedings of the IEEE International Conference on Distributed Computing Systems, 2005*, 2005, pp. 653–662.

[23] D. Georgoulas and K. Blow, "Making Motes Intelligent: An Agent-Based Approach to Wireless Sensor Networks," *WSEAS on Communications Journal*, vol. 5, no. 3, pp. 525–522, March 2006.

[24] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards Sensor Database Systems," in *Mobile Data Management*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2001, pp. 3–14.

[25] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, March 2005.

[26] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 6, pp. 621 – 655, 2008.

[27] J. Pan, L. Cai, Y. T. Hou, Y. Shi, and S. X. Shen, "Optimal Base-Station Locations in Two-Tiered Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 5, pp. 458–473, 2005.

[28] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks," *IEEE Transactions on Computer*, vol. 51, no. 12, pp. 1448–1453, 2002.

[29] Y. Osais, M. St-Hilaire, and F. R. Yu, "On Sensor Placement for Directional Wireless Sensor Networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Dresden, Germany, 2009, pp. 1 – 5.

[30] S. Soro, "Application-Aware Resource Management in Wireless and Visual Sensor Networks," Ph.D. dissertation, School of Engineering and Applied Sciences, University of Rochester, 2007.

[31] Y. Osais, M. St-Hilaire, and F. R. Yu, "The Minimum Cost Sensor Placement Problem for Directional Wireless Sensor Networks," in *Proceedings of the IEEE Vehicular Technology Conference*, 2008, pp. 1 – 5.

[32] G. Fusco and H. Gupta, "Selection and Orientation of Directional Sensors for Coverage Maximization," in *Proceedings of the IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2009, pp. 1 – 9.

[33] X. Han, X. Cao, E. L. Lloyd, and C.-C. Shen, "Deploying Directional Sensor Networks with Guarranteed Connectivity and Coverage," in *Proceedings of the IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, San Francisco, CA, USA, 2008, pp. 153–160.

[34] E. Hörster and R. Lienhart, "Approximating Optimal Visual Sensor Placement," in *Proceedings of the International Conference on Multimedia and Expo*, 2006, pp. 1257–1260.

[35] Y. Cai, W. Lou, M. Li, and X.-Y. Li, "Energy Efficient Target-Oriented Scheduling in Directional Sensor Networks," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1259–1274, 2009.

[36] 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*, vol. 27, no. 5, pp. 78–90, 2010.

[37] A. Mittal and L. S. Davis, "A General Method for Sensor Planning in Multi-Sensor Systems: Extension to Random Occlusion," *International Journal of Computer Vision*, vol. 76, pp. 31–52, 2008.

[38] C. Piciarelli, C. Micheloni, and G. L. Foresti, "Occlusion-aware multiple camera reconfiguration," in *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*, 2010, pp. 88–94.

[39] C. Soto, B. Song, and A. K. Roy-Chowdhury, "Distributed multi-target tracking in a self-configuring camera network," in *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, USA, 20-25 Jun 2009, pp. 1486–1493.

[40] 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*, 2009, pp. 1 – 8.

[41] F. Qureshi and D. Terzopoulos, "Smart Camera Networks in Virtual Reality," in *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*, Vienna, Austria, 2007, pp. 1640 – 1656.

[42] C. Rotar, M. RÎsteiu, I. Ileana, and C.-T. Hutanu, "Optimal sensors network layout using evolutionary algorithms," in *Proceedings of the 10th WSEAS international conference on Automation & Information*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 88–93.

[43] R. Rajagopalan, C. K. Mohan, K. G. Mehrotra, and P. K. Varshney, "Multi-objective evolutionary algorithms for sensor network design," in *MultiObjective Optimization in Computational Intelligence Theory and Practice*, L. T. Bui and S. Alam, Eds. Information Science Reference, 2008, pp. 208–238.

[44] C. Coello Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28 – 36, feb 2006.

[45] E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on evolutionary multiobjective optimization," *Metaheuristics for Multiobjective Optimisation*, vol. 535, no. 535, pp. 21–40, 2004.