Distributed Online Visual Sensor Network Reconfiguration for Resource-aware Coverage and Task Assignment

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

***Abstract*—A visual sensor network (VSN) consists of resource-limited camera nodes which process the captured image data locally and collaborate with other cameras over a wireless network. The configuration of the network is a very important task in VSNs in order to adapt the available resources to the current requirements of the application. We focus on coverage and task assignment as a key configuration problem for VSNs. Due to rapid changes in the VSN's environment, a dynamic and online reconfiguration is highly needed. In this paper we introduce and evaluate a fast and distributed resource-aware reconfiguration algorithm. Our approach is based on simple optimization primitives to find good approximations and to keep the required data transfer in the VSN small. We analyze the communication complexity and compare our algorithm with a centralized configuration approach based on several scenarios with different complexity. Our experiments show that we can achieve the same configuration quality as with the centralized approach in almost all cases.**

I. Introduction

In a Visual Sensor Network (VSN) a large number of camera nodes form a distributed system. They process image data locally to extract relevant information, collaborate with other cameras on the application-specific task and provide the system's user with information-rich descriptions of captured events [1]. These networks can be found in various applications including assisted living, surveillance and entertainment [2].

VSNs represent networks of embedded sensors and processors with tight resource limitations. Still, VSNs have to process large amounts of visual data in real-time and perform rather complex algorithms to fulfill the application requirements. These requirements are significantly different to typical wireless sensor network applications and pose novel challenges for deployment and operation of VSNs. We focus on one such challenge, i.e., *resource-aware coverage and task assignment* 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 [3], we formally introduced this configuration problem and presented a centralized approximation method for it. This evolutionary algorithm is used to approximate an optimal tradeoff between surveillance quality and resource consumption. This method is well suited for a-priori configuration since all information on observation points and node resources need to be in a central place to run this algorithm. In a dynamic environment, where frequent changes occur, this would lead to high communication effort and significant latency in reconfiguration. To *dynamically change* the coverage and task assignment in VSN, we introduce and evaluate in this paper a fast, distributed algorithm capable of finding good approximations of the configuration problem. It exchanges so called descriptors to inform other nodes of possible solutions and is able to improve a solution over time. Our approach is based on simple optimization primitives and keeps the required messages in the VSN small. Such a resource-aware algorithm is very important for a deployment on the resource-limited camera nodes.

The remainder of this paper is organized as follows: Section II describes relevant related work. In Section III we present a concise problem formulation. Section IV introduces our approach to distributed sensor selection and task assignment. We evaluate our approach in Section V and conclude and point out future work in Section VI.

II. Related Work

Resource limitation is a typical problem many multi-camera networks have to face [4], [5]. In recent years many approaches to improve energy efficiency in visual sensor networks [6], [1] and smart camera networks [2] have been proposed.

Casares et al. [7] try to maximize processor idle time on smart cameras in order to prolong the node lifetime. In [8] additional sensors are used to selectively turn on and off cameras to save energy.

Selecting the best camera for a particular task is crucial in a VSN. Strategies for task assignment and object handoff influence the overall resource consumption [9], [10], [11]. Yu et al. [12] propose a camera selection and energy allocation strategy to deliver a user-requested view in a VSN using a stochastic model of the network lifetime. Cenedese et al. [13] formalize the problem of task assginment of multiagent-driven camera networks and present a decentralized algorithm to solve it. Tessens et al. [14] propose a method to select the camera which currently provides the best view of a certain scene. Shen et al. [15] employ mechanisms of selforganization to select cameras in a distributed smart camera network.

Although the related work discussed above addresses various aspects of resource constrained visual sensor networks, there are significant differences to the approach presented in this paper. We propose a highly-flexible, dynamic and completely

distributed approach not only aiming on assigning tasks to camera nodes in a resource aware manner but also manipulating camera settings, namely frame rate and resolution in order to further reduce the resources needed to perform the network task. It is able to adapt to changes in the environment and can also deal with heterogeneous activity requirements in the network.

The approach presented in this paper builds upon our previous work as presented in [3]. It uses the same models for coverage and resources and thus produces comparable results. It is however, a different approach from an algorithmic point of view. While our previous work presents a centralized evolutionary algorithm, we show in this paper a completely distributed algorithm which is nevertheless able to achieve the same result quality in most cases.

III. Problem Formulation

This section presents a concise definition of our coverage and task assignment problem and briefly sketches a centralized approximation algorithm.

Figure 1 shows our configuration problem. n camera sensors are placed on a 2D space; the coverage area of each camera is represented by a segment. Observation points represent the tasks to be performed by the network. Each has to be covered by at least one camera at a given QoS (determined by the frame rate fps and the pixels on target pot) performing a certain monitoring activity for the observation point while not exceeding the available resources (processing, memory and energy).

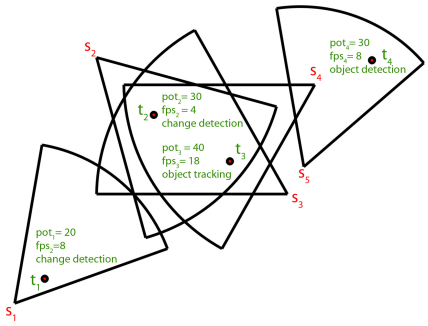


Fig. 1. A simple scenario

For modeling the network configuration problem we make the following assumptions: *i)* 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. *ii)* Each camera captures images (at the defined resolution and frame rate) and executes a sequence of image processing procedures and transfers 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. *iii)* The observation points are static locations and must be covered by at least one camera's FOV at sufficient pot and fps. pot is determined by the sensor resolution and the distance between camera and observation point. Note, that observation points themselves are not moving but they rather indicate spots of high activity of a certain kind. However, they may be added or removed at runtime if activity hotspots change or an external operator redefines them. *iv)* We currently only consider a convex 2D space without obstacles restricting the camera's FOV.

A. Problem Definition

We define $S = \{s_1, \ldots, s_n\}$ as the set of all sensors and their associated properties such as position or field of view and $T = \{t_1, \ldots, t_m\}$ as the set of all all observation points and their surveillance QoS requirements "pixels on target" (pot), "framerate" (fps) and "surveillance activity" (a).

An activity represents a high-level monitoring task such as *image compression and streaming*, *change detection*, *object detection*, *person counting* and *object tracking*. We define the set of all activities that can be performed as $A = \{a_1, \ldots, a_l\}$. 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 where all requirements regarding resources, QoS and activity are satisfied. No sensor may exceed its memory and processing capabilities. The required resources for the given input data configuration can be computed $\tilde{r}$.

1) 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. In this paper, we are focusing on (i) quality (pot, fps, a) and (ii) energy usage. Naturally, different criteria can be defined as well.

2) Requirements for a Distributed Algorithm: Runtime reconfiguring is key for reacting to occurring events and task changes. Monitoring tasks (i.e., the observation points and their requirements) may change frequently depending e.g. on the activity in the area. A centralized algorithm needs to collect information about all nodes in a single node which requires expensive communication.

In general, observation points can be in the FOV of multiple cameras. This coverage can be easily represented as a bipartite graph. Strongly connected coverage graphs pose a challenge for distributed algorithms, since an update of the task assignment on a single camera may introduce updates on all connected (via shared observation points) cameras. Such chain of updates may then also trigger an update on the originating camera which may result in update loops.

IV. Distributed Algorithm

	Step				
Node	**0**	**1**	**2**	**3**	**4**
A	d_A			d_B	
B			d_B		
C		d_C			d_B

Fig. 2. A sample communication pattern of the algorithm in a slotted representation. Node A defines the observation point, Node B has the best solution for it. d_X denotes a message which carries the solution found by node X.

The basic idea of the algorithm presented here is that autonomous camera nodes act greedily to cover observation

points (also called targets) in their FOV. They exchange *descriptors* describing the required resources to cover an observation point. Improved solutions are found by comparing descriptors. Periodic re-evaluation of the assignments (the targets covered by that camera) improves the solution iteratively. We assume that the information on a new observation point is disseminated by a single node. The node might for example use activity maps [16] to determine where a new observation point must be placed. In this case, this node will be the first to transmit information about the new observation point. If an external operator defines new targets, this information will enter the network at a single sensor node, thus this node will further disseminate this information. We do not assume any information on 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 (for networks which span farther than the broadcast range of a node) of descriptors using this mechanism. This mechanism also improves the robustness against message loss.

Algorithm distributed_coverage_and_assignment()

On define new observation point t:
  if t can be covered:
    Calculate required settings and resources
    Broadcast descriptor
  fi

On receive descriptor d for target t:
  if t is not in FOV **or** d is stored as best solution
    end
  fi
  if no queue for t exists:
    create queue q_t for t and set timer
  else
    add d to q_t
    restart timer
  fi

On timer for q_t elapsed:
  take best descriptor d from q_t
  if better descriptor d_s available (local / stored solution)
    Broadcast d_s
  else
    Store d as remote best descriptor for t
    Broadcast d
  fi
  Delete q_t

Do periodically:
  Select target t from targets covered by this node
  Send out descriptor for t

Fig. 3. Event-based pseudo code of the distributed algorithm. "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.

[1]Neighboring cameras are connected via two edges in the coverage graph. If we know the neighbors, we can multicast the descriptors to these nodes and reduce the communication in the network

A. Events in the algorithm

Figure 3 shows a pseudocode description of the distributed algorithm. Nodes react to events such as the occurrence new targets or the reception of new descriptors.
Figure 2 shows a sample execution with three nodes. After node A broadcasts an initial descriptor for the new observation point, node C is the first to answer but the best solution is found by node B. In steps 3 and 4, the nodes A and C confirm the best solution of B by broadcasting the corresponding descriptor once again. This is also done to enable multi-hop communication and to compensate for message loss.

1) Processing of New Targets: Nodes detect hotspots which need additional surveillance activity and define new observation points with the respective surveillance requirements. New observation points may also be defined by external operators. For a new observation point, a node calculates the necessary resolution, framerate and activity needed to cover this points and the additional resource usage for those settings. A target may be covered "for free" if a node already covers another target that requires the same (or higher) settings. Else, the additional resource demand is used as basis for a descriptor (the difference between the current resource usage and the predicted total usage if the target is covered). The corresponding descriptor is sent to the other nodes.

2) Processing a Descriptor: Received descriptors are compared, the best of all for a certain observation point will be stored at the node and broadcast. Descriptors are buffered for a configurable period of time and the best of all received descriptors is selected. Immediately replying to new descriptors results in a higher amount of messages. A small random factor is additionally added to this interval to desynchronize the nodes. This is done to prevent nodes from processing their results at the same time. Upon receiving a descriptor, the node first checks if it covers the respective observation point. If so, it either adds it to an existing queue or creates a new one for this target.
After the buffering interval has elapsed, a local descriptor (the costs for it to cover it locally) is compared to the best stored and the best buffered one. The best descriptor is stored locally and is broadcast to the other nodes as a confirmation. If the node's local descriptor it the best, it will cover the given target. A node that has no observation points assigned can shut down its sensory layer to save energy.

3) Periodic Activities: Each node periodically tries to hand off one of the targets it covers to enable better solutions from other nodes. This incrementally increases the quality of the solution. We call this mechanism the runtime re-evaluation of observation points.
This mechanism is also used as a keep-alive signal to detect faulty nodes. This increases the robustness of the system and reduces the number of messages exchanged at runtime.

V. Evaluation and Results

In this section, we present a thorough evaluation of the presented algorithm. We first show a theoretical evaluation on the communication complexity which is then used to evaluate practical simulation results.

A. Communication Complexity

In this section, we approximate the communication complexity of the presented algorithm. To do this, we assume the communication to be done in a slotted fashion. We assume that only one node can transmit its message at a given time and

that all other nodes overhear this message and that the order in which nodes send is random.
For this evaluation, we define node neighborhood as the joint observation of a certain target by multiple cameras, i.e., it is the set of all cameras which have this target in their FOV. Using the function

$$c(s,t) = \begin{cases} 1 & \text{if } t \text{ is in the field of view of } s. \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

we define the neighborhood of cameras respective to an observation point t as

$$N_t = \{s \in S | c(s,t) = 1\} \quad (2)$$

Note, that we define best and worst cases for a single observation point only. To calculate the respective numbers for a scenario consisting of multiple observation points, the sum of all observation points must be taken. The two messaging patterns for the best and worst cases are shown in Figure 4.

1) Best Case: In the best case, the node which initially defines a new target also has the best solution for it. In this case, it broadcasts its descriptor and all neighboring nodes reply to confirm. This corresponds to a best case communication complexity of $|N_t|$ messages for a single observation point t where $|N_t|$ is the number of cameras in the neighborhood wrt. this target point.

	Step			
Node	0	1	2	3
A	d_A			
B				d_A
C			d_A	
D		d_A		

	Step									
	0	1	2	3	4	5	6	7	8	9
A	d_A		d_B			d_C				d_D
B		d_B			d_C				d_D	
C				d_C				d_D		
D							d_D			

Fig. 4. The best case (left) and worst case (right) communication complexity.

2) Worst Case: The worst case arises if the node which initially defines a new target has the least optimal solution for it and if the subsequent communication is performed in reverse order of solution quality (i.e., nodes with worse solutions communicate first). In this constellation we reach the worst case communication complexity of

$$\frac{|N_t| \cdot (|N_t| + 1)}{2}$$

In Figure 4 we assume that the solution quality corresponds to the alphabetical order of the node ID, i.e. that node D has the best solution. However, node A transmits an initial descriptor. Node B replies with the next best solution which is then confirmed by A before C transmits its solution. It can easily be seen that any other transmission order would result in a lower number of required messages. If, for example, D transmits its solution immediately after the initial descriptor of A, the total number of messages required is 5.

B. Experimental Evaluation

In a real implementation and in simulations we evaluate our presented algorithm. We use the central algorithm presented in [3] as a benchmark for the distributed approach and evaluate the number of messages needed. Third, we show the algorithm's behavior in scenarios with message loss.

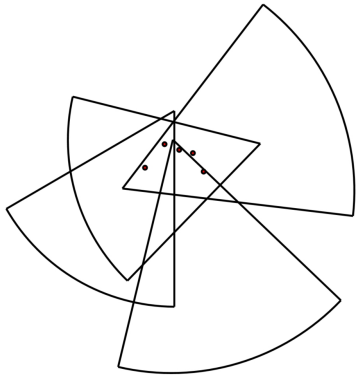


Fig. 5. The camera and observation node placement in the practical scenario

In [3] the appropriateness of our resource models has been demonstrated. We have shown that the predictions of our energy and resource usage models closely match the values measured on real hardware. To achieve results for large scenarios, we rely on simulations. However, we first show the results from a real deployment to show that results from simulation and real deployments correspond.

Experimental setting: As a practical application, we have defined a scenario with four Pandaboard-based[2] embedded cameras deployed to cover an area of approx. 50×30 meters with five, partially shared observation points (Figure 5). The observation points require either simple background subtraction, object detection or tracking. We use OpenCV[3] algorithm implementations (BgStatModel, BlobDetector, TemplateTracking). We measure the time until a stable solution is reached and compare the solution to the simulation results.

Simulation setting: We have defined six scenarios with typical deployment scenarios for VSNs with *i)* separated clusters with overlapping fields of view (e.g., surveillance of separate rooms in buildings or intersections of streets), *ii)* connected fields of view (e.g., continuous surveillance of pathways or large halls) and *iii)* large-scale complex networks.

The first two scenarios—which have separated clusters—are used to show the behavior of the algorithm in low-complexity settings. While scenario b defines the same observation points (Figure 6(a)) as scenario a, it has more sensors covering those points. In scenario a 10 sensors are deployed while 15 sensors cover the observation points in scenario b.

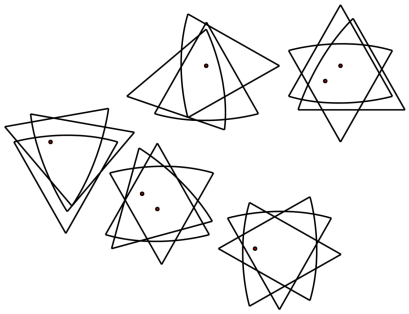
The third scenario (Figure 6(b)) has connected clusters and is thus more complex to solve. Every node in scenario c transitively shares its field of view with every other sensor. In total, scenario c has 15 sensors and observation points respectively.

Additional to the three fixed test scenarios we evaluate the behavior of the proposed approach in a very large network. We have generated a random scenario of 100 sensors and 25 observation points. By adding more observation points we have derived two further scenarios containing 33 and 50 observation points respectively.

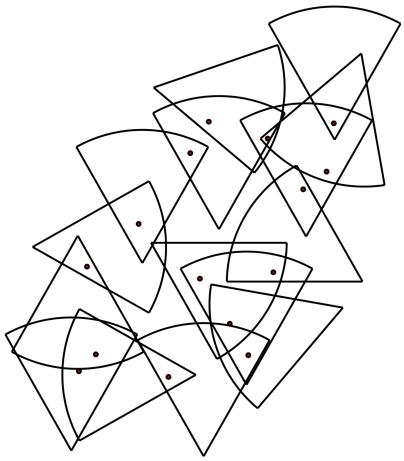
In all scenarios we use the resource usage prediction of the central algorithm as a benchmark. We measure the per-

[2]http://www.pandaboard.org

[3]http://opencv.willowgarage.com



(a) The low complexity scenario b). Scenario a) has the same observation points but one camera less per cluster.



(b) High complex scenario with circular cluster interdependencies.

Fig. 6. Scenarios b and c.

Scenario	Best Case	Worst Case	Average Overlap
real	14	27	2.8
a	14	21	2
b	21	42	3
c	38	70	2.53

TABLE I. BEST AND WORST CASE COMMUNICATION COMPLEXITY ALONG WITH THE AVERAGE NUMBER OF FOV OVERLAP PER OBSERVATION POINT FOR SCENARIOS a - d.

formance of the algorithm in terms of deviation from the benchmark (the higher the deviation, the more resources will the solution demand during runtime). We also measure the number of messages the algorithm needs. From 500 simulation runs per scenario we show the deviation from the reference result of the central algorithm and the number of messages needed to initially cover all points. We also show, how often the algorithm finds the optimal result initially and after a re-evaluation phase of 100 message. Additionally we show the deviation from the benchmark result after the re-evaluation.
The experiments are performed in two steps:

1) Disabled re-evaluation allows estimating the quality of results achieved when the algorithm terminates after all points have been covered.
2) Doing re-evaluation for 100 messages shows the improvement made by this mechanism.

Additionally, we do the same evaluation again for scenario c with message loss rates of 5% and 15%. In this experiment we first took the initial result and also evaluated the result after an extra 100 messages as well.

Targets	Best Case	Worst Case	Average Overlap
25	48	77	1.92
33	68	116	2.06
50	102	169	2.04

TABLE II. BEST AND WORST CASE COMMUNICATION COMPLEXITY ALONG WITH THE AVERAGE NUMBER OF FOV OVERLAP PER OBSERVATION POINT FOR THE COMPLEX SCENARIOS WITH 25, 33 AND 50 OBSERVATION POINTS.

C. Evaluation of the Distributed Algorithm

Deviation experiment vs. simulation	2.45%
Time to find solution	12s
Messages *experimental*	26
Messages *sim*	21.3
CPU node 1 (*simulation* - *experimental*)	90.6% - 95.3%

TABLE III. RESULTS OF THE REALWORLD DEPLOYMENT COMPARED TO THE SIMULATED RESULTS

1) Experimental results: In Table III we show the results from our practical test. It can be seen that the simulation achieves results very similar to a real system both, in terms of achieved results and required messages. We also show how the actual processor load on node 1 corresponds to the predicted load. The experimental system takes 12 seconds to find the solutions for all observation points.

2) Simulation results: We compare the distributed to the central algorithm. The central algorithm has global information as input while the distributed algorithm finds a solution only using information available on the local node
The results of this test series are shown in Table IV. It shows that the algorithm performs very well for simple and medium complex scenarios finding the optimal result for scenario a already in the initial assignment phase. The low number of messages needed makes the algorithm suitable for resource-limited networks (especially, considering that one message only has a few bytes of size).
Note, that in scenarios b, c cases with more messages than in the worst case occured. This is a consequence from the slotted communication assumption in our theoretical evaluation. In the real—unslotted—implementation with unsynchronized buffering, there are rare cases with suboptimal descriptor buffering times resulting in slightly more messages than the worst case. The algorithm found the optimal result for scenario c initially in 4.5% of our test cases. This rate increased after the re-evaluation to nearly 42%, i.e., the algorithm finds results fast and the re-evaluation allows the improvement of the solution.

A distributed algorithm that may be used in networks with non-reliable message transport must be able to deal with message loss. We performed simulations with 5% and 15% message loss to evaluate our approach.
The results for scenarios with 5% and 15% message loss

	Scenario		
	a	b	c
Deviation initially [%]			
Average	0	1.5	11.9
Min - Max	0 - 0	0 - 36.8	0 - 64.9
Number of messages			
Average	17.5	31.2	54.1
Min - Max	14 - 21	22 - 43	44 - 78
Optimal results in initial run [%]			
	100	45.1	4.5
Opt results after +100 messages [%]			
	100	45.5	41.8
Deviation after +100 messages [%]			
Average	0	1.3	6.7
Min - Max	0 - 0	0 - 12.2	0 - 55.5

TABLE IV. RESULTS OF THE DISTRIBUTED ALGORITHM COMPARED TO THE BENCHMARK WITH NO MESSAGE LOSS.

(Table V) are still very close to the results with no loss. While the average deviation from the optimal result increases and reaches a rather high deviation in some rare cases, the added value of performing a continuous improvement after the initial assignments can be seen very clearly since the improvements made after additional 100 messages is very large.

Loss	Deviation [%]	# Messages	Opt. runs initial - re-eval. [%]	Deviation re-eval. [%]
5%	25 - 0 - 218.3	51.2 - 38 - 64	2.2 - 45.2	7.1 - 0 - 55.5
15%	56.7 - 0 - 285.0	50.0 - 38 - 67	0.4 - 36	9.2 - 0 - 55.5

TABLE V. RESULTS OF THE DISTRIBUTED ALGORITHM FOR SCENARIO *c* COMPARED TO THE CENTRAL EA WITH VARYING MESSAGE LOSS.

3) Large-scale and Complex Networks: For very complex scenarios, we show in Table VI that the algorithm yields results very close to the optimum and that the average number of messages needed per observation point remains stable.

# OP	Dev. [%]	# Msgs.	Msgs. / OP
25	0.36 - 0 - 1.5	63.5 - 57 - 70	2.54
33	0.3 - 0 - 1	93.5 - 81 - 110	2.83
50	0.73 - 0 - 2.6	138.9 - 117 - 152	2.73

TABLE VI. THE RESULTS FOR 25, 33 AND 50 OBSERVATION POINTS (# OP).

VI. CONCLUSION AND FUTURE WORK

In this paper we have presented a purely distributed approach to resource-aware sensor selection and task assignment in visual sensor networks. Our results show that the algorithm is simple, fast, scalable and reliable even if messages lost. The algorithm achieves results very close to the central evolutionary algorithm presented in previous work, in most cases it is able to find the optimal result. The algorithm scales linearly and is thus applicable in small and large networks. We are exploring VSN reconfiguration in even more dynamic environments. A first approach to include object handover in the reconfiguration has recently been published in [17].

ACKNOWLEDGMENTS

This work is supported by Lakeside Labs GmbH, Klagenfurt, Austria and is funded in part by the European Union Seventh Framework Programme under grant agreement n° 257906, the European Regional Development Fund (ERDF) and the Carinthian Economic Promotion Fund (KWF) under grant KWF 20214/18354/27107

REFERENCES

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

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

[3] B. Dieber, C. Micheloni, and B. Rinner, "Resource-Aware Coverage and Task Assignment in Visual Sensor Networks," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 10, pp. 1424–1437, Oct.

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

[5] F. Al Machot, C. Tasso, B. Dieber, K. Kyamakya, C. Piciarelli, C. Micheloni, S. Londero, M. Valotto, P. Omero, and B. Rinner, "Smart Resource-Aware Multimedia Ssensor Network for Aautomatic Ddetection of Complex Events," in *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, 30 2011-Sept. 2, pp. 402–407.

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

[7] M. Casares and S. Velipasalar, "Adaptive Methodologies for Energy-Efficient Object Detection and Tracking With Battery-Powered Embedded Smart Cameras," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 10, pp. 1438–1452, oct. 2011.

[8] J. Nayak, L. Gonzalez-Argueta, B. Song, A. Roy-Chowdhury, and E. Tuncel, "Multi-target tracking through opportunistic camera control in a resource constrained multimodal sensor network," in *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, sept. 2008, pp. 1 –10.

[9] L. Esterle, P. R. Lewis, X. Yao, and B. Rinner, "Socio-economic vision graph generation and handover in distributed smart camera networks," *ACM Transactions on Sensor Networks*, 2013, to appear.

[10] 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.

[11] Y. Li and B. Bhanu, "A comparison of techniques for camera selection and handoff in a video network," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, 30 2009-sept. 2 2009, pp. 1 –8.

[12] 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.

[13] A. Cenedese, F. Cerruti, M. Fabbro, C. Masiero, and L. Schenato, "Decentralized task assignment in camera networks," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, dec. 2010, pp. 126 –131.

[14] L. Tessens, M. Morbee, H. Lee, W. Philips, and H. Aghajan, "Principal view determination for camera selection in distributed smart camera networks," in *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, sept. 2008, pp. 1 –10.

[15] E. Shen and R. Hornsey, "Camera selection using a local image quality metric for a distributed smart camera network," in *Sensors, 2011 IEEE*, oct. 2011, pp. 1217 –1220.

[16] 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.

[17] B. Dieber, L. Esterle, and B. Rinner, "Distributed resource-aware task assignment for complex monitoring scenarios in visual sensor networks," in *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, 2012, pp. 1–6.