Efficient Feature Distribution for Object Matching in Visual-Sensor Networks

Vildana Sulić, *Student Member, IEEE*, Janez Perš, *Member, IEEE*, Matej Kristan, *Member, IEEE*, and Stanislav Kovačič, *Member, IEEE*

***Abstract*—In this paper we propose a framework of hierarchical feature distribution for object matching in a network of visual sensors. In our approach we hierarchically distribute the information in such a way that each individual node maintains only a small amount of information about the objects seen by the network. Nevertheless, this amount is sufficient to efficiently route queries through the network without any degradation of the matching performance. A set of requirements that have to be fulfilled by the object-matching method to be used in such a framework is defined. We provide examples of mapping four well-known, object-matching methods to a hierarchical feature-distribution scheme. The proposed approach was tested on a standard COIL-100 image database and in a basic surveillance scenario using our own distributed network simulator. The results show that the amount of data transmitted through the network can be significantly reduced in comparison to naive feature-distribution schemes such as flooding.**

***Index Terms*—computer vision, distributed systems, object matching, visual-sensor networks.**

I. Introduction

In the past decade, digital cameras have become ubiquitous, and in the past few years a noticeable trend towards the use of smart cameras has emerged. Furthermore, with developments in the field of sensor networks, networks of smart cameras (visual-sensor networks – VSNs) have become a hot topic of research. The use of smart cameras includes such diverse areas as surveillance, traffic or environmental monitoring, analyses of sporting events and ambient intelligence. On the other hand, we have witnessed rapidly decreasing prices for the associated hardware, which has allowed the deployment of different types of digital cameras in "smart" objects for personal use, i.e., mobile phones.

In comparison to other types of sensors, cameras provide large amounts of digital data. Whenever a large number of cameras is used together, e.g., in a typical VSN, the problems of data transmission and automatic data processing are significantly amplified. The automatic processing of visual data is the task of computer vision, which deals with extracting the useful information from images and video sequences. Since computer-vision algorithms operate on large amounts of digitized visual data, some type of high-end hardware is usually needed. In terms of network topology, this leads to a star network structure – a powerful processing unit and one or more sensors (cameras).

Such a structure has some advantages, such as the simplicity of the routing. A conceptual problem of such a centralized approach is that it is not scalable, i.e., it does not scale with the number of sensors used. When additional nodes are added to such a configuration, the central processor becomes a major bottleneck. In some cases, the number of visual sensors may go into the hundreds (for example, the London Congestion Charge monitoring system consists of more than 200 cameras). It is obvious that the requirements for transmitting and processing the data in such a large system are correspondingly large.

For systems like this, a decentralized, distributed structure is a natural choice. Distributed systems do not rely on a single central processor, but rather each node takes over its share of the processing load. The transport of data may be based on hop-by-hop routing, which means that the node only knows how to reach its immediate neighbors, while the task of reaching the final destination requires the participation of many nodes. This means that computer-vision algorithms have to be mapped to the *distributed environment*, which is a challenge on its own.

In this paper we deal with a fundamental task in computer vision: the detection and matching of objects. In this task, visual sensors (cameras) acquire images, which are then used to extract the knowledge needed to perform the object matching at some later time. In a distributed network structure, knowledge about the detected objects has to be somehow available to all the nodes in the network.

This can be trivially done by *flooding* the network with information, in the hope that the knowledge will reach each and every node in the network and, therefore, the matching can be done locally. Conversely, information can be stored locally on a sensing node; however, establishing a correspondence between the same object, observed by different nodes at different times, would require flooding as well. We aim to find a tradeoff between these two extremes – a method which does not require the distribution of all the data to each and every node, but still provides a means to obtain a correspondence between the same objects seen by different cameras.

Our approach is to distribute the visual knowledge hierarchically. Complete information about the object (e.g., a complete set of feature vectors) is stored in a node that has captured the original image of an object. All the other nodes receive less-detailed (more abstract) visual information

Manuscript received February 31, 2011; revised February 31, 2011 and February 31, 2011; accepted February 31, 2011. Date of publication February 31, 2011; date of current version February 31, 2011. This work was supported by research program P2-0095 and research grant 1000-07-310221, both by the Slovenian Research Agency.

The authors are with the Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, SI-1000 Slovenia (e-mail: vildana.sulic@fe.uni-lj.si; janez.pers@fe.uni-lj.si)

Digital Object Identifier 10.2222/TCSVT.2011.1111111

(e.g., less-descriptive feature vectors). Since, in general, more abstract visual information requires less storage space and less transmission capacity, significant savings in that respect are possible. The abstract visual information is only used to efficiently route the network queries when a correspondence between objects has to be established (e.g., in the matching phase). In this way, the efficient transmission of information in the matching phase is achieved as well. However, during propagation some information is inevitably lost due to abstraction. To achieve an identical matching performance only complete information (e.g., full feature vectors) is used for the matching. To make this possible, the matching method has to fulfill certain requirements.

The major contribution of this work is a hierarchical feature-distribution scheme for object matching that allows the distribution of reduced feature vectors without any degradation in the matching performance. We also define four requirements (abstraction, storage, existence of a metric, convergence) for the matching method to be used in the proposed scheme and provide an algorithm for the efficient routing of the network queries during the matching.

The remainder of the paper is organized as follows. In Section II we present an overview of the related work. Section III introduces the concept of a hierarchical, feature-distribution scheme, mathematically formulates a set of requirements and proposes algorithms for efficient feature distribution and distributed object matching. Four basic object-matching methods are analyzed and shown to fulfill the stated requirements in Section IV, and in Section V the experiments and results of tests in an application-layer, distributed-network simulator are reported. Section VI provides our conclusions and future work.

II. Related work

Visual-sensor networks (VSNs) are the meeting point of two different technologies. On one side there is the distributed sensor approach, which puts significant constraints on the available processing power and the network's transmission capabilities. On the other side, there are image-processing and computer-vision algorithms, which are both computationally and data intensive. Therefore, the main issues in visual-sensor networks revolve around the task of achieving the maximum performance on hardware with limited capabilities.

In this section we present an overview of the relevant publications. A few review papers on wireless, multimedia, sensor networks, such as by Misra *et al.* [1] or by Akyildiz *et al.* [2], can be found in the literature. The latest developments in multi-camera networks are presented by Aghajan and Cavallaro [3], while a special issue on Distributed Smart Cameras [4] provides an overview of the trends and challenges in the field of distributed smart cameras. Charfi *et al.* [5] highlighted the challenging issues and opportunities in visual-sensor networks. Major research issues in VSNs, such as camera coverage, network-architecture design, energy-aware data processing and communication, are discussed in the same survey. Soro and Heinzelman [6] go one step further, and also address other aspects of VSNs. They focus on the low-power and low-complexity aspects of visual-sensor networks. By emphasizing the unique characteristics of VSNs, such as the resource requirements, local processing, real-time performance, precise location and orientation information, time synchronization, data storage and autonomous camera collaboration, they also provide research directions in VSNs. They divide the research directions into several areas: signal-processing algorithms (problems related to camera calibration and research related to object detection, tracking, and high-level vision processing), wireless networking (network problems related to real-time data communication, camera collaboration and route selection), sensor management, hardware architecture for VSNs (energy consumption, platforms and testbeds) and middleware (which bridges the gap between the application and the low-level network structure).

We divide our overview into the three categories that are the most relevant to our work. In the first category, we explore the relevant achievements in the deployment of computer vision on embedded systems. In the second category, we present publications regarding the efficient transmission of data between nodes, and in the last category, we present relevant research that deals with the problems that originate from the distributed nature of visual-sensor networks.

Embedded computer-vision systems. With the widespread use of digital cameras, the prevalence of wireless connectivity and the integration of visual sensors into objects for personal use (like mobile phones or PDAs), the technology of visual-sensor networks has become widespread [7]. We expect that the prevalent implementation of the networked visual sensor in the future will be in the form of an embedded computer-vision system. Embedded systems impose significant constraints on the choice of computer-vision and pattern-recognition algorithms in VSNs. Due to limited resources, the choice of algorithms used in embedded systems reflects mainly those constraints. For example, Tessens *et al.* [8] proposed a radically simplified foreground-detection algorithm, Bramberger *et al.* [9] used basic background modelling for a traffic-surveillance task and Quaritsch *et al.* [10] suggested the use of the computationally inexpensive CamShift algorithm for object tracking. Wang *et al.* [11] and Fleck *et al.* [12] used adapted object-tracking algorithms to reduce the computational load on the embedded system. On the other hand, Arth and Bischof [13] used a state-of-the-art approach to object recognition; however, they proposed a hybrid solution, where computationally intensive training is done off-line on a conventional PC and the recognition is performed on an embedded camera using a nearest-neighbor search. In this work we are faced with a similar problem (object recognition), but due to the integral nature of our solution all the processing needs to be done on the camera nodes. According to the taxonomy of Soro and Heinzelman [6], the above work, including our own, belongs to the area of signal-processing algorithms for embedded systems.

Efficient data transmission. One of the major issues in visual-sensor networks is dealing with an efficient data transmission between the nodes. The transmission of visual information usually requires a large bandwidth and, therefore, a specifically tailored optimization of the distributed sensor topologies is very desirable. The research into data-

transmission techniques in visual-sensor networks can be roughly grouped into three problem areas [5]. First, there is a need for efficient image- and video-transmission methods for transmission over a single hop, such as by Yu *et al.* [14], Lee *et al.* [15] or Lecuire *et al.* [16]. The motivation behind their work is similar to ours: to transmit visual information as efficiently as possible and, if needed, to degrade the data in a predictable way. Our research differs in the nature of the transmitted data since we transmit visual object features which are then used for efficient routing; however, the core principles are related. Techniques that explicitly deal with multi-hop transmission strategies were proposed, for example, by Wu and Abouzeid [17] and multi-path transmission techniques were used by Charfi *et al.* [18]. Our work addresses multi-hop transmission differently, by examining the contents of data packets for similarities with already transmitted data. Therefore, it provides a mechanism for the efficient multi-hop transmission of pattern-recognition problems, in particular for distributed object matching.

Distributed nature. The next major issue in visual-sensor networks is their distributed nature. Computer-vision algorithms have to be either adapted or totally rebuilt to deal with the specifics of distributed networks. Two major areas of research can be identified here. The first deals with distributed calibration algorithms. Different approaches to the calibration in distributed systems are proposed, such as by Simmons *et al.* [19], by Devarajan and Radke [20], and by Sinha and Pollefeys [21]. In [22] Cheng *et al.* use a vision graph to calibrate the cameras. Once the temporal and spatial relations between the nodes in the network are known, possibilities for other tasks, such as multi-view recognition, are open. For example, Kirishima *et al.* [23] propose a framework for multi-view gesture recognition based on distributed image processing. Zhang *et al.* [24] deal with the key challenges for a multi-camera surveillance system, where each camera unit should share its information through the centralized data-fusion sensor. Our framework does not explicitly deal with multi-view recognition or distributed calibration; however, those approaches can be used to extend the framework with such a functionality.

The other major area of research is the efficient propagation of knowledge, which is also the central theme of our work. This issue has been studied by many authors. Ihler [25] investigated the conceptual problems of inference in distributed sensor networks and Sulić *et al.* [26] presented some conceptual ideas for using hierarchy to distribute the data in VSNs more efficiently. Patricio *et al.* [27] applied a multi-agent framework to a VSN to control the capture parameters of a surveillance system and Gilbert and Bowden [28] presented a system for the incremental, scalable, inter-camera tracking of objects. In the work of Chang *et al.* [29] a camera network system can estimate its topology and auto-organize its own activities according to the content of the scene and the task to be undertaken. This is achieved by using custom encoding, which reduces the amount of information that has to be transmitted across the network. In [22] Cheng *et al.* achieve a reduction in the amount of information by distributing "feature digests" across the network. Leistner *et al.* [30] use the on-line, co-training of classifiers on neighboring cameras. By exchanging homographies between the camera views they are able to reduce the amount of transmitted data. Karakaya and Qi [31] perform target detection and counting using a certainty map. They achieve a reduction in the amount of transmitted information by using local processing and the efficient propagation of data through the network. Park *et al.* [32] build distributed look-up tables from camera viewing angles that are distributed across the network. Such tables are used to select a subset of cameras that is likely to carry out a particular task most effectively. Similarly, our work addresses the problem of distributing object features for distributed object matching. However, to forward network queries across many hops, our framework relies only on the information about a node's direct neighbors, while the tables proposed by Park *et al.* are global in nature. Yang *et al.* [33] perform object recognition in distributed VSNs. They use the local extraction of features, random projection and the properties of chosen image features to significantly reduce the amount of information to be transmitted. In contrast to our work, they assume a star network topology with a single, high-powered base station and allow no communications between the cameras. Nevertheless, in many aspects [32] and [33] are the most closely related work to our own. Our framework provides an implicit routing of network queries to the sensors that are most likely to provide an answer. On the other hand, it exploits certain properties of the distributed features to minimize the amount of transmitted data.

Nevertheless, there is comparatively little research on the conceptual issues associated with VSNs – a significant portion of the related work deals with narrowly defined problems, with implementations often tied to specific hardware. Our work aims to address the conceptual problem of information propagation in sensor networks, which is particulary acute in *visual*-sensor networks. In that respect, it provides benefits for implementations on embedded systems, and provides a way to distribute the information in distributed systems in general.

III. A HIERARCHICAL FEATURE-DISTRIBUTION SCHEME FOR OBJECT MATCHING

A. Problem formulation

A *node* in a VSN is an object that consist of a visual sensor, a local memory, a central processing unit and one or more connections to the other network nodes. The computational and communication resources of such a node are usually known, but limited. Its primary task is the acquisition of images via its visual sensor and the processing of those images according to a particular task, which may involve communications with other nodes in the VSN. In our case, that task is *object matching* – each seen object is matched against all the available object *features* to establish whether a particular object has been seen before. In the VSN, a single node does not have all the relevant features to make such a decision. The features, belonging to the objects seen by the other network nodes reside in the local memory of those nodes. If a particular node wants to match a previously seen object, all the features from the network have to be requested for

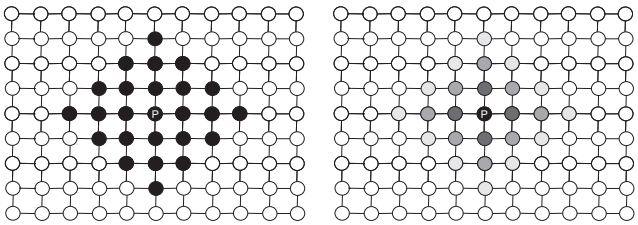


Fig. 1. Illustration of a network state during the feature-distribution process, with "P" denoting the primary node. (a) When features are distributed by flooding, each node receives an exact copy of the original feature vector (black). (b) When a hierarchical distribution scheme is used, the nodes receive progressively less-detailed (more abstract) feature vectors (shades of grey).

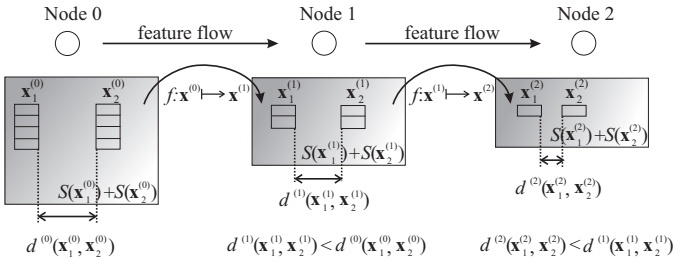


Fig. 2. Illustration of the four requirements. As the feature vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ are propagated through the network, they are transformed by the mapping f on each hop. The required storage space S (grey area) decreases after each hop, along with the distance d between the two feature vectors.

a comparison. In this way, a distributed feature comparison results in a non-negligible amount of network traffic. Our aim is to reduce that amount of traffic while *preserving the matching performance*.

B. Distributed object matching

We define distributed object matching as follows: *given the acquired image of an object, find all the images of visually similar objects that have been acquired by any of the nodes on any previous occasion.* In general, object matching consists of the following two phases:

- Learning phase: the compact representation (model) of the object is extracted from one or more images and stored. We assume that such a compact representation is in the form of a feature vector.
- Matching phase: the same compact representation of an object (a feature vector, in our case) is extracted from the newly acquired image. This vector is compared to the feature vectors stored during the learning phase to obtain a correspondence with one of the learned objects.

In principle, the entire network can be flooded with image data or object features whenever a new image is acquired. In this way, every image would be distributed to each node, which constitutes an extremely wasteful use of network resources. In the next section we propose a scheme that avoids the distribution of complete feature vectors through a novel hierarchical encoding scheme.

C. The hierarchical encoding structure

Our hierarchical encoding scheme is based on a hierarchical reduction of feature vectors. We require that the *primary node* (the visual sensor that has originally seen the unknown object) retains the complete information about the object (e.g., an unmodified feature vector). Its neighbors receive less-detailed, more abstract information, which generally requires less storage and transmission capacity. In this way, the amount of data transmitted across, or stored in the network, can be significantly reduced. This principle is illustrated in Fig. 1.

Such a structure inevitably leads to a loss of information at the point when the features are transformed to their less-detailed representations. In general, this leads to a decrease in the matching performance. To reduce the amount of traffic, while preserving matching performance, we propose four requirements that have to be fulfilled by any object-matching method to be considered for use in the proposed hierarchical scheme. Given that the information about the object is stored as a feature vector $\mathbf{x}$, the requirements are as follows (Fig. 2):

Requirement 1 (Abstraction): There exists a mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}, 0 < n \leq N$, which translates a level n feature vector $\mathbf{x}^{(n)}$ into a higher, more abstract, level $(n+1)$ feature vector $\mathbf{x}^{(n+1)}$. N denotes the highest level of abstraction. In other words, using the mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$, which translates a particular feature vector into a more abstract feature vector, the dimensionality of the feature vectors is reduced.

This requirement assumes that the primary node extracts a level 0 feature vector, $\mathbf{x}^{(0)}$, directly from the acquired image. Its direct neighbors receive more abstract level 1 feature vectors, $\mathbf{x}^{(1)}$, their neighbors receive level 2 feature vectors, $\mathbf{x}^{(2)}$, and so on. The mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$ is done on each of the nodes before transmitting the feature vectors $\mathbf{x}^{(n+1)}$ to the neighbors, until the highest level of abstraction is reached. From this point on, the feature vectors are forwarded unchanged. The maximum level of abstraction can be matching-method dependent (e.g., at some point it may become impossible to further reduce the dimensionality of the feature vector). Another important consideration when selecting N is the network-transmission overhead – from a certain point on the reduction in features does not meaningfully reduce the length of the network packets. Requirement 1 is necessary to build a hierarchical feature structure. If the mapping f does not exist, a hierarchy cannot be achieved.

Requirement 2 (Storage): If $S(\mathbf{x})$ is the storage space required for the feature vector $\mathbf{x}$ in bits, then it should hold that: $S(\mathbf{x}^{(n)}) \geq S(\mathbf{x}^{(n+1)})$.

If this requirement is not fulfilled, the hierarchical encoding scheme does not provide any improvements in terms of network traffic and data storage.

Requirement 3 (Existence of a metric): There exists a metric $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)})$, which provides a measure of the *similarity* between two feature vectors $\mathbf{x}_1^{(n)}$ and $\mathbf{x}_2^{(n)}$ of the same level n.

The existence of the metric is essential, both for the object matching itself and for the hierarchical feature-encoding scheme. The distance $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)})$, when compared to the threshold T, determines whether the objects are *similar*,

$d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}) \leq T$, or not, $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}) > T$. If the metric does not exist, then a comparison between the objects cannot be obtained.

Requirement 4 (Convergence): Given two vectors $\mathbf{x}_1^{(n)}$ and $\mathbf{x}_2^{(n)}$ that are similar, $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}) \leq T$, the corresponding vectors on the next level $n+1$ should be at least as similar as the vectors on the previous level, $d^{(n+1)}(\mathbf{x}_1^{(n+1)}, \mathbf{x}_2^{(n+1)}) \leq d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)})$.

The fourth requirement guarantees the following: whenever the comparison between the feature vectors $\mathbf{x}_1^{(n)}$ and $\mathbf{x}_2^{(n)}$ results in a distance $d^{(n)}$ above the threshold T, we can be sure that there will be no match between the same feature vectors $\mathbf{x}_1^{(m)}$ and $\mathbf{x}_2^{(m)}$ on any of the lower ($m < n$) levels. This has an important consequence for the matching phase, when searching for the match across the network. The search can be terminated early in the directions where there are no possibilities of finding a match. A violation of Requirement 4 would cause a node with the feature $\mathbf{x}^{(n)}$ to block access to the node that may have the matching features $\mathbf{x}^{(m)}$, given that $m < n$.

Any matching method that fulfills the stated requirements can be implemented hierarchically. However, due to Requirement 4, the matching performance of the chosen matching method would remain exactly the same in the hierarchical implementation. It is possible, nevertheless, to implement the matching method in such a framework, even if it violates Requirement 4. In this case, the matching performance would drop in proportion to the ratio of feature vectors $\mathbf{x}$ for which the method would violate the requirement. The consequence would be an increase in the number of cases where a previously observed and known object would be declared unknown by the network.

D. The propagation of feature vectors

Let us assume that the primary node has acquired an image of a new object and has already discovered that the observed object has not been seen before. Equivalently, we may assume that the network is in learning-only mode.

First, the feature vector is extracted and a unique[1] identification number (ID) is generated and attached to the feature vector. The actual procedure of feature extraction depends on the matching method used. The extracted vector is then stored locally and marked as being a level 0 vector, $\mathbf{x}^{(0)}$. Then, using the mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$, the next level feature vector $\mathbf{x}^{(1)}$ is prepared, assigned the same ID and broadcasted to all the direct neighbors of the primary node. Each receiving node attaches a tag to the received vector, which uniquely determines the origin of the feature vector. Due to Requirement 2, the level 1 feature vectors $\mathbf{x}^{(1)}$ require less storage space than the vectors $\mathbf{x}^{(0)}$. The process of applying the mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$ and broadcasting to the node's neighbors is repeated on each receiving node, until every node has at least *some* information about the object seen by the primary node. Communications between the nodes and the unique IDs ensure that the nodes refuse to accept any duplicated feature vectors.

[1]For example, generated from the appropriate hardware identifier (such as a MAC address) and the sequential number of the acquired image.

E. Object matching

The task of object matching may be performed concurrently with learning (e.g., the network tries to match the object before proceeding with the learning). For now, let us assume that the network is in matching-only mode. The algorithm for object matching is presented in Algorithm 1. During object matching we call the node that has acquired the image the *querying node*. The Algorithm 1 is recursive in its nature. In the matching phase, the querying node generates a network *query* packet, if the object has not been seen locally (no match on level 0), but it has been seen by any of the other nodes (match on other, higher levels). A match means that two feature vectors are similar on the same level. The query packet, generated by the querying node, contains unmodified level 0 feature vectors of an unknown object, and is transmitted to those querying node's neighbors, which provided matching feature vectors on some higher level. Upon receiving the query packet, every node runs the Algorithm 1 from the Line 2 on.

Algorithm 1 : Object matching

Input: Image
Output: Object correspondence

1: Extract object features $\mathbf{x}^{(0)}$.
2: // Local search
3: **for** All levels in the local storage **do**
4: Apply the mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$ and calculate the next level feature $\mathbf{x}^{(n+1)}$ from $\mathbf{x}^{(n)}$.
5: Compare $\mathbf{x}^{(n+1)}$ with all the vectors of level $n+1$ from the local storage.
6: **if** No match is found **then**
7: Terminate the search, object is unknown. Optionally, proceed with learning.
8: **else if** Match is found on the level 0 **then**
9: Object has been seen locally.
10: **else**
11: // Some other node might have seen the object.
12: // Proceed with network search.
13: **for** All matching vectors **do**
14: Examine tags, attached to the locally stored matching feature vector.
15: Forward level 0 features of the unknown object to the neighbor, who provided locally stored matching feature vector.
16: // Upon receiving forwarded features, neighboring nodes start from Line 2 of the Algorithm 1.
17: **end for**
18: **end if**
19: **end for**

Neighboring nodes process the forwarded feature vectors of an unknown object in the exactly same way, as if they would acquire the image of an unknown object by themselves. On each node the result of the processing is either:

- The object is unknown (if there is no local match). This result is not reported to the querying node (the node that originally saw an object).
- The object is known (a match on level 0 is found). This result is reported to the querying node.
- The object might have been seen by the network (*possible* match found on one of the higher levels).

The effect of this algorithm is that, during the matching phase, the feature vectors of an unknown object in the unmodified form (level 0 feature vectors) are forwarded from node to node along the trail left by the propagated feature vectors in the learning phase. If the querying node does not receive any replies from the other nodes in a *reasonable amount of time*, the object is unknown to the network. A reasonable amount of time in this context is network and application dependent. In theory, the querying node should wait for at least double the amount of time that a packet needs to reach the most distant node in the network. A more efficient solution could be to introduce a hop-counter to each feature vector, to enable the calculation of more accurate timeout values at the querying nodes.

The efficiency of our approach stems from the fact that the feature vectors are only forwarded in a direction where there is a possibility that the object has been seen (according to Requirement 4).

F. Self-organization and fault-tolerance

Other desirable properties in visual-sensor networks are self-organization and fault-tolerance. The proposed hierarchical feature-distribution scheme is essentially a high-level, self-organizing mechanism. As described above, the routing of the query packets during the matching phase depends on the information stored in the network during the learning. This may provide a certain degree of fault-tolerance, as follows.

- In the case of a node malfunction during the learning, the feature vectors will be simply propagated around it, provided that there are alternative connections available.
- In the case that a non-primary node enters an off-line state after learning, the matching process may be significantly disrupted due to the disruption in the routing of the query packets. The seriousness of the disruption may be reduced if we allow the nodes to accept duplicate feature vectors in the learning phase. These duplicate feature vectors would provide multiple alternative directions for the routing of query packets if one of the node's neighbors goes off-line.
- In the case that a primary node goes off-line after learning, the match on level 0 cannot be provided and no result is reported to the querying node. This could be alleviated by modifying the feature distribution scheme, by requiring that all the primary nodes distribute level 0 feature vectors to their immediate neighbors. In this way the robustness could be increased at the expense of the efficiency. Another way of increasing the robustness would be to allow the nodes with level 1 feature vectors to provide answers to the querying node if the primary node is off-line. In this case the matching performance would decrease.

Obviously, our framework imposes a trade-off between routing efficiency and fault-tolerance. The appropriate balance between the two would need a systematic treatment and, therefore, is beyond the scope of this paper.

IV. Illustrative examples

Our hierarchical feature-encoding framework does not rely on any particular object-matching method. It only sets the requirements that enable a given matching method to be implemented in a distributed way. To be able to illustrate the performance of our approach, we selected four matching methods. These are principal component analysis (PCA), 2D Haar transform, template matching, and histogram matching. In the following sections we illustrate the appropriate mappings $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$ and the metrics $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)})$, for which these four methods fulfill the stated requirements.

A. Methods based on orthogonal projection and Euclidean distance

All the methods described in this section use Euclidean distance as their metric and in this respect they share some of the properties regarding the use in our framework.

1) Principal component analysis and 2D Haar transform: PCA (also known as the Karhunen-Loéve transform) is a vector-space transform that reduces multidimensional data sets to lower dimensions, while minimizing the loss of information. This is achieved by finding a linear basis of reduced dimensionality for the data (a set of eigenvectors) in which the variance in the data is a maximum [34]. In our case, we obtain the eigenvectors in advance, and they remain fixed throughout the learning and matching phase.

The Haar transform is a linear transformation into the subspace of Haar functions (Haar wavelets). In our case, the 2D Haar transform was used and the feature vectors were obtained by unwrapping the transformation result into a column vector.

Requirement 1: With PCA, properly constructed feature vectors contain feature values that are already ordered by decreasing importance in terms of the reconstruction of the original data. This opens up the possibility of the mapping function $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$, which can be defined as dropping a certain number of features with the lowest importance from the feature vector. The same mapping was also used for the Haar feature vectors, and in this case it results in the dropping of the features that correspond to the highest frequencies of the Haar wavelets.

Requirement 2: It is fulfilled, since dropping any number of dimensions from the feature vector decreases the required storage space.

Requirement 3: Considering the metric $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)})$, the Euclidean distance is used when comparing PCA-based or Haar-based feature vectors.

Requirement 4: It is easy to show that Requirement 4 holds as well, if the Euclidean distance is used. Ignoring one of the dimensions from the Euclidean space never increases the

distance between the two points. At most, the distance remains the same. This also holds for high-dimensional cases. That means, regardless of the order of the features, the distance will always decrease with the decreased dimensionality of the feature vectors, and Requirement 4 is fulfilled.

2) Template matching: Template matching is a global object representation that uses the instance of an object, a template, to search for the same (or similar) object in the image. For the sake of clarity we limit ourselves to a direct comparison of two images of the same dimensions. In this case, the feature vector contains pixel values of the original image.

Requirement 1: We can define the mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$ as a simple subsampling operation[2] that reduces the image dimensions by calculating 2×2 pixels averages. This is only one possibility. Nevertheless: other convolution kernels, such as Gaussian, could be used.

Requirement 2: The resulting image dimensions are halved, both the image and the corresponding feature vector require only a quarter of the original storage space and Requirement 2 is fulfilled.

Requirement 3: We can define the metric $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)})$ as the Euclidean distance between the feature vectors.

Requirement 4: Let us use metric $d^{(n)}$ to compare two images $A^{(n)}$ and $B^{(n)}$. The original images $A^{(n)}$ and $B^{(n)}$ have the dimensions $k \times k$ and the resized images $A^{(n+1)}$ and $B^{(n+1)}$ have the dimensions $k/2 \times k/2$. Let $\mathbf{x}_A^{(n)}$, $\mathbf{x}_B^{(n)}$, $\mathbf{x}_A^{(n+1)}$, $\mathbf{x}_B^{(n+1)}$ be the level n and $(n+1)$ feature vectors, respectively. Since the following inequality holds:

$$\begin{aligned} &d^{(n)}(\mathbf{x}_A^{(n)}, \mathbf{x}_B^{(n)}) \geq d^{(n+1)}(\mathbf{x}_A^{(n+1)}, \mathbf{x}_B^{(n+1)}), \\ &\sqrt{\sum_{i=1}^{k}\sum_{j=1}^{k}(A^{(n)}(i,j) - B^{(n)}(i,j))^2} \\ &\quad \geq \sqrt{\sum_{i=1}^{k/2}\sum_{j=1}^{k/2}(A^{(n+1)}(i,j) - B^{(n+1)}(i,j))^2}, \end{aligned} \tag{1}$$

Requirement 4 is fulfilled. The proof can be found in Appendix A.

B. Histogram matching

Let I be an image. The intensity histogram with P bins $\mathbf{h}_x = \{h_i\}_{i=1}^{P}$ sampled within the image I, is defined as:

$$h_i = f_h \sum_{\mathbf{u} \in U} \delta_i(p(\mathbf{u})), \tag{2}$$

where $\mathbf{u} = (x, y)$ denotes a pixel within the image region I. $\delta_i(\cdot)$ is the Kronecker delta function positioned at histogram bin i and $p(\mathbf{u}) \in \{1...P\}$ denotes the histogram bin index associated with the intensity of a pixel at location $\mathbf{u}$ and f_h is a normalizing constant such that $\sum_{i=1}^{P} h_i = 1$. Each histogram bin contains a normalized count of image pixels within a certain range of grey levels. Accordingly, the elements of the feature vectors are simply normalized histogram bin counts.

[2]Note that image resizing can be implemented as a special case of the Haar transform; here we follow a traditional formulation with a convolution kernel.

Requirement 1: We can define the mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$ as an operation that combines adjoining bins, giving a coarser representation of the original image.

Requirement 2: Since a smaller number of bins requires less storage space, Requirement 2 is fulfilled.

Requirement 3: The metric $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)})$ can be the Hellinger distance [35] between the histograms:

$$d^{(n)}(\mathbf{x}_A^{(n)}, \mathbf{x}_B^{(n)}) = \sqrt{1 - \rho(h_A^{(n)}, h_B^{(n)})}, \tag{3}$$

where $d^{(n)}(\mathbf{x}_A^{(n)}, \mathbf{x}_B^{(n)})$ is the distance between the feature vectors, $\mathbf{x}_A^{(n)}$ and $\mathbf{x}_B^{(n)}$, $\rho(h_A^{(n)}, h_B^{(n)})$ is the Bhattacharyya coefficient, $\rho(h_A^{(n)}, h_B^{(n)}) = \sum_{i=1}^{P} \sqrt{h_{iA}^{(n)} h_{iB}^{(n)}}$. Note that there are other possible choices for the metric d, e.g., the Chi-square distance or the histogram-intersection-based distance [36].

Requirement 4: Let us assume that we have two images $A^{(n)}$ and $B^{(n)}$ with the corresponding histograms $h_A^{(n)}$ and $h_B^{(n)}$. Let $\mathbf{x}_A^{(n)}$, $\mathbf{x}_B^{(n)}$ and $h_A^{(n)}$, $h_B^{(n)}$ be the level n feature vectors and level n histograms and $\mathbf{x}_A^{(n+1)}$, $\mathbf{x}_B^{(n+1)}$ and $h_A^{(n+1)}$, $h_B^{(n+1)}$ be the feature vectors and the level $n+1$ histograms, respectively. Since the following inequality holds:

$$\begin{aligned} d^{(n)}(\mathbf{x}_A^{(n)}, \mathbf{x}_B^{(n)}) &\geq d^{(n+1)}(\mathbf{x}_A^{(n+1)}, \mathbf{x}_B^{(n+1)}), \\ \sqrt{1 - \rho(h_A^{(n)}, h_B^{(n)})} &\geq \sqrt{1 - \rho(h_A^{(n+1)}, h_B^{(n+1)})}, \\ \rho(h_A^{(n)}, h_B^{(n)}) &\leq \rho(h_A^{(n+1)}, h_B^{(n+1)}), \end{aligned} \tag{4}$$

Requirement 4 is fulfilled. The proof can be found in Appendix B.

V. Experiments and results

In our experiments we tested three types of *feature-distribution methods*:

- $\mathbf{M}_{\mathrm{flood-at-match}}$ corresponds to the scenario where the captured visual information is stored locally and each task of finding an object has to be broadcasted across the network by flooding, for each new image acquired. Such a method of distribution requires little or no network traffic during the learning phase; however, it produces a large amount of data transmitted in the matching phase.
- $\mathbf{M}_{\mathrm{flood-at-learn}}$ corresponds to the scenario where the captured visual information from each sensor is distributed to all the nodes for local storage. Again, flooding is used for this purpose. The detection of similar objects is then performed locally by each sensor as new images are acquired. In contrast to the first method, this feature-distribution method produces a large amount of data transmitted during the learning phase and requires very little or no traffic during the matching phase.
- $\mathbf{M}_{\mathrm{hier}}$, the feature-distribution method proposed in this paper.

We used the standard COIL-100 database that consists of images of 100 different objects; each one is rotated by a 5-degree angle interval, corresponding to 72 images per object. This sums up to 7,200 images for the whole database [37]. We have chosen the COIL-100 database since the view variations represent possible points of failure for the tested matching methods. We expected that there would be a certain amount of

matching failures, which is important for a complete evaluation of the proposed feature-distribution scheme.

Four basic object-matching methods, template matching, histogram matching, PCA and 2D Haar transform, were used for our testing of the different feature-distribution methods. For the template matching, the maximum number of levels N was set to 5, which resulted in minimum image dimensions of 4×4 pixels at the level $n = 5$. The reason for this is that we encountered significant rounding errors when 2×2 images were used (the features, which were essentially image pixels, were stored as unsigned 8-bit values). For the histogram matching, the first 10 bins (out of 256) were set to zero immediately after the level 0 features were extracted from the image. This was done to decrease the effect of the black background in the images from the COIL-100 database, which dominated the histogram comparison if unmodified histograms were used. The PCA projection space was built beforehand using only every eighth object orientation. The remaining images were then used in the matching. The Haar transformation matrix was calculated beforehand. The matching accuracy was measured in terms of false positives (FPs) and false negatives (FNs). A FP occurred when the distance between the feature vectors of two different objects was lower than a threshold T and a FN occurred when the distance between feature vectors corresponding to the same object was higher than T. Additionally, the number of computations required to perform the mappings f and calculate the metrics d was measured.

A. Simulator

To test the proposed concept of a hierarchical feature-distribution scheme, we designed a distributed network simulator. It runs on a standard desktop computer and is written in Matlab. The simulator deals only with the application layer of the network communication. This means that it ignores network-related phenomena, such as network delays and limitations in the network bandwidth. The simulator measures the amount of traffic transmitted between the nodes, the number of nodes (hops) over which the traffic is transmitted and the processing load due to the invocation of the mapping f and the metric d, as described in Requirements 1 and 4, respectively.

The simulator can accept a network with any structure; however, the routing algorithm has to be provided as well. To get a good insight into the situation in the network, we limited our testing to just the rectangular, 4-connected grid networks. Each node is assigned a two-dimensional address, which is directly related to the node's location in the rectangular grid. In this way, each node can easily determine the direction in which a particular packet should be forwarded.

The simulator allows us to inject any type of network packet at any point in the network. It also allows the injection of a new image at any node in the network. Image processing, feature extraction, feature processing and feature comparison are integral parts of the simulation. For example, to test the object matching, we inject an image into a particular node and wait until the network activity ends. Then we read the result of the matching from the same node.

For the experiments, we used a network consisting of 99 nodes, arranged in a 11×9 rectangular grid. Each of the nodes, except those on the network boundaries, was connected to its four immediate neighbors; however, a similar experiment could be run with a different network topology.

B. Evaluation of the network performance

Several experiments were performed to evaluate the network performance of different feature-distribution methods. The first (learning) phase measured the performance of the network during learning. Twenty nodes (primary nodes), evenly distributed through the network, were injected with 100 images of the 100 different objects from the database. Those images corresponded to the zero orientation of objects in the COIL-100 database. In this instance, 20 nodes among a total of 99 nodes received images. The situation is shown in Fig. 3(a). Next, the simulation cycle was started, and, after the network traffic stopped, the statistics on the network load (the number of hops, the total network traffic per sample) and the statistics on processing load were examined.

The second (matching) phase measured the performance of the network during the matching. A pseudo-random sequence was used to choose an image from the database. Images used in the learning and in the building of the PCA subspace were excluded from the database for this step. A pseudo-random sequence was also used to choose any node from the network. The chosen image was injected into the chosen node, and the simulation cycle was started. After the network activity stopped, the result of the matching was read from the same node, and the statistics on the FPs, FNs and processing load were updated. The process of injecting a random image at a random node was repeated 5,000 times, and the overall statistics were recorded. Therefore, the object matching was tested with 5,000 images.

Both phases were performed 12 times, once for each combination of object-matching methods and feature-distribution methods. In each of the 12 trials, the same pseudo-random sequence was used in the matching to ensure that the results can be compared between the different combinations of methods.

The results for both phases are shown in Table I. We can see that in comparison to $\mathrm{M}_{\mathrm{flood-at-match}}$ and $\mathrm{M}_{\mathrm{flood-at-learn}}$, the proposed $\mathrm{M}_{\mathrm{hier}}$ results in a far lower network-traffic load and number of hops-per-sample. Moreover, $\mathrm{M}_{\mathrm{hier}}$ achieves equal matching rates to both flood methods, despite the data reduction that occurs in $\mathrm{M}_{\mathrm{hier}}$. The reason is that the fulfillment of Requirement 4 guarantees that the matching performance will not decrease with the application of the hierarchical feature-vector distribution scheme. In this respect all the feature-vector distribution methods are equivalent, even though our hierarchical method does not distribute full feature vectors.

During the learning $\mathrm{M}_{\mathrm{flood-at-match}}$ makes no use of the network (all the feature vectors are stored locally). Therefore, the results for $\mathrm{M}_{\mathrm{flood-at-match}}$ amount to zero hops and no traffic. On the other hand, the traffic drastically increases during the matching (matching phase), when this feature-distribution method is used. In the matching phase, $\mathrm{M}_{\mathrm{hier}}$ outperforms $\mathrm{M}_{\mathrm{flood-at-match}}$ in terms of both network-traffic load and number of hops-per-sample. On the other hand,

TABLE I
EXPERIMENTAL RESULTS FOR EACH COMBINATION OF MATCHING METHODS AND FEATURE-DISTRIBUTION METHODS

Matching method	Distribution method	Learning Hops $[\frac{1}{sample}]$	Learning Traffic $[\frac{kbytes}{sample}]$	Matching Hops $[\frac{1}{sample}]$	Matching Traffic $[\frac{kbytes}{sample}]$	Total traffic* $[Mbytes]$	Matching rate FPs**	Matching rate FNs**
Template	$\mathrm{M}_{\mathrm{flood-at-match}}$	0	0	398	4132	20660	22%	22%
	$\mathrm{M}_{\mathrm{flood-at-learn}}$	614	2855	0	0	285	22%	22%
	$\mathbf{M_{hier}}$	**614**	**41**	**279**	**2228**	**11144**	**22**%	**22**%
Histogram	$\mathrm{M}_{\mathrm{flood-at-match}}$	0	0	432	521	2605	27%	28%
	$\mathrm{M}_{\mathrm{flood-at-learn}}$	614	363	0	0	36	27%	28%
	$\mathbf{M_{hier}}$	**614**	**25**	**309**	**273**	**1367**	**27**%	**28**%
PCA	$\mathrm{M}_{\mathrm{flood-at-match}}$	0	0	381	1036	5180	19%	22%
	$\mathrm{M}_{\mathrm{flood-at-learn}}$	614	719	0	0	72	19%	22%
	$\mathbf{M_{hier}}$	**614**	**44**	**243**	**486**	**2434**	**19**%	**22**%
Haar 2D	$\mathrm{M}_{\mathrm{flood-at-match}}$	0	0	376	1036	5180	18%	21%
	$\mathrm{M}_{\mathrm{flood-at-learn}}$	614	719	0	0	72	18%	21%
	$\mathbf{M_{hier}}$	**614**	**44**	**251**	**536**	**2684**	**18**%	**21**%

* Note that total traffic depends on the number of samples used. It is based on 100 learning and 5,000 testing samples.
** FPs and FNs denote the percentage of false positives and false negatives, respectively.

TABLE II
COMPUTATIONAL COST FOR EACH COMBINATION OF MATCHING METHODS AND FEATURE-DISTRIBUTION METHODS

Matching method	Distribution method	Feature extraction $[\frac{operations}{sample^*}]$ +	Feature extraction $[\frac{operations}{sample^*}]$ ×	Mapping f Learning $[\frac{10^6 operations}{sample}]$ +	Mapping f Learning ×	Mapping f Matching +	Mapping f Matching ×	Metric d Matching $[\frac{10^6 operations}{sample}]$ +	Metric d Matching ×	Metric d Matching $\sqrt{\cdot}$**
Template	$\mathrm{M}_{\mathrm{flood-at-match}}$	0	0	0	0	0	0	1.6384	1.6384	0.0001
	$\mathrm{M}_{\mathrm{flood-at-learn}}$			0	0	0	0	1.6384	1.6384	0.0001
	$\mathbf{M_{hier}}$			**0.0999**	**0.0333**	**102.50**	**34.167**	**3.7994**	**3.7994**	**0.0065**
Histogram	$\mathrm{M}_{\mathrm{flood-at-match}}$	16384	256	0	0	0	0	0.0256	0.0256	0.0257
	$\mathrm{M}_{\mathrm{flood-at-learn}}$			0	0	0	0	0.0256	0.0256	0.0257
	$\mathbf{M_{hier}}$			**0.0023**	**0**	**1.4795**	**0**	**0.1475**	**0.1475**	**0.1538**
PCA	$\mathrm{M}_{\mathrm{flood-at-match}}$	$8.4 \cdot 10^6$	$8.4 \cdot 10^6$	0	0	0	0	0.0512	0.0512	0.0001
	$\mathrm{M}_{\mathrm{flood-at-learn}}$			0	0	0	0	0.0512	0.0512	0.0001
	$\mathbf{M_{hier}}$			**0**	**0**	**0**	**0**	**0.2715**	**0.2715**	**0.0059**
Haar 2D	$\mathrm{M}_{\mathrm{flood-at-match}}$	$8.4 \cdot 10^6$	$8.4 \cdot 10^6$	0	0	0	0	0.0512	0.0512	0.0001
	$\mathrm{M}_{\mathrm{flood-at-learn}}$			0	0	0	0	0.0512	0.0512	0.0001
	$\mathbf{M_{hier}}$			**0**	**0**	**0**	**0**	**0.2911**	**0.2911**	**0.0063**

* Cost of extracting features from a single COIL-100 image, resolution 128×128 pixels. The cost of conversion from color to gray-scale image is not included.
** The number of square roots corresponds to the direct implementation of the proposed metrics d without any optimization. For actual implementation, optimized variants could be used.

$\mathrm{M}_{\mathrm{flood-at-learn}}$ makes no use of the network during the matching, while it makes the heaviest use in the learning phase (each and every node in the network has to receive a complete feature vector). Again, the $\mathrm{M}_{\mathrm{hier}}$ outperforms $\mathrm{M}_{\mathrm{flood-at-learn}}$ during the learning phase. Note that the total traffic depends on the number of samples used. It is provided for illustrative purposes only and it is based on 100 learning and 5,000 testing samples.

Table II shows the computational cost in the number of operations (additions, multiplications and square root calculations) per each phase of our experiment. The cost of feature extraction does not depend on the distribution method and is specified as the number of operations to extract a feature vector from a single COIL-100 database image. It is clear that in some cases the proposed hierarchical feature-distribution scheme requires an additional computation of the mapping $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}$ during the learning phase. The matching methods based on an orthogonal projection have an advantage in this respect, since the dropping of features requires no additional computation. The proposed hierarchical feature-distribution scheme also increases the number of operations per sample during the matching in comparison to the flood-based distribution methods, since calculations of the metric $d^{(n)}$ are needed along the route of the query packet.

Finally, the results depend on the maximum level of abstraction as well. To obtain the results presented in Tables I and II we did not limit the maximum level of abstraction, except for the template matching, as documented at the beginning of this section. To examine the influence of the maximum level of abstraction, the experiments were repeated by varying this parameter. The traffic during the learning phase increased when more descriptive features were used (less abstraction was allowed); however, the traffic and the number of hops during the matching phase decreased. This behavior is in line with expectations – more descriptive feature vectors cause more traffic when distributed, but enable greater accuracy of the

routing in the matching phase.

To summarize, a phase-by-phase (learning – matching) comparison of the network traffic shows that our proposed $\mathrm{M}_{\mathrm{hier}}$ significantly outperforms $\mathrm{M}_{\mathrm{flood-at-learn}}$ during the learning and $\mathrm{M}_{\mathrm{flood-at-match}}$ during the matching. Essentially, while those two methods represent the extreme ends of the network-load spectrum (one with high traffic during the learning, and other with high traffic during the matching), our method can utilize the network resources in a balanced way.

The three described feature-distribution schemes allow different trade-offs between several variables of interest, including, but not limited to, the amount of data stored on the nodes, the computational cost, and the network load. These variables depend on multiple factors, e.g., the size of the network, the number of learning samples, the number of primary nodes, the discriminatory power of the chosen matching method and the maximum level of abstraction in the case of $\mathrm{M}_{\mathrm{hier}}$. For example, in addition to the network load, the $\mathrm{M}_{\mathrm{hier}}$ also reduces the storage cost in comparison to $\mathrm{M}_{\mathrm{flood-at-learn}}$. On the other hand, $\mathrm{M}_{\mathrm{hier}}$ may not be the best choice if we know that the number of learning samples will be low. Such an example would be a surveillance task in an office area with limited access and permanent staff. In this case, $\mathrm{M}_{\mathrm{flood-at-learn}}$ would outperform the proposed distribution scheme in terms of overall performance. With a larger number of learning samples the performance of $\mathrm{M}_{\mathrm{flood-at-learn}}$ will quickly deteriorate in terms of total traffic and the required storage space (because with $\mathrm{M}_{\mathrm{flood-at-learn}}$ every node needs to store all the information about every observed object). It is up to the engineer to pick the most appropriate approach for the particular application.

C. Effect of the number of primary nodes

To evaluate how the number and distribution of nodes that receive original images in the learning phase (primary nodes) affect the network efficiency, the experiment from the previous section was repeated. The number of injected images was the same as in the previous tests, while the number and distribution of the primary nodes was varied (as shown in Fig. 3):

- 10 nodes are injected with 10 images each in the learning phase (ratio of 10:99, Fig. 3(b)),
- 4 nodes are injected with 25 images each in the learning phase (ratio of 4:99, Fig. 3(c)),
- 2 nodes are injected with 50 images each in the learning phase (ratio of 2:99, Fig. 3(d)),
- 1 node is injected with 100 images in the learning phase (ratio of 1:99, Fig. 3(e)).

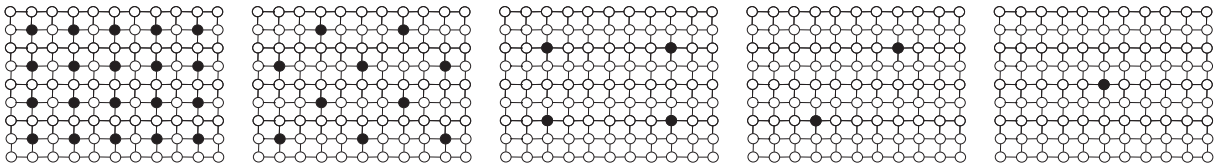
The results are shown in Table III. We see that during the learning phase the number of hops does not change with the different ratios of the primary nodes. On the other hand, the network traffic increases slightly with a lower density of primary nodes, regardless of which feature-distribution method is used. Nevertheless, the network traffic is significantly lower when $\mathrm{M}_{\mathrm{hier}}$ is used in comparison to $\mathrm{M}_{\mathrm{flood-at-learn}}$. During the matching phase we see that the number of hops and the amount of network traffic remain almost constant when $\mathrm{M}_{\mathrm{flood-at-match}}$ is used. On the other hand, with a decreasing density of primary nodes, both the number of hops and the amount of network traffic decreases significantly, when $\mathrm{M}_{\mathrm{hier}}$ is used. The reason for such behavior is that queries are forwarded only in directions in which possible matches can be found. This means that no queries will be sent to the nodes that did not observe any of the new objects and therefore did not originate any feature vectors. For example, consider the situation shown in Fig. 3(e), where all the new objects have been seen by a single primary node. Every feature vector distributed by our method points to that single node, and each query will be routed to that node only or dropped along the way. If there were two such primary nodes (Fig. 3(d)), every query would have been routed, at most, in two directions. Since the placement of primary nodes influences the average network distance over which the data is transmitted, it is clear that it increases the total amount of traffic. However, the influence of the placement of the primary nodes is less significant than the effect of reducing the actual number of primary nodes.

Note that one could modify the generic $\mathrm{M}_{\mathrm{flood-at-match}}$ to perform better in this test. Primary nodes could broadcast a small amount of data which would indicate that they are in possession of learning samples. This would improve the efficiency of the routing in $\mathrm{M}_{\mathrm{flood-at-match}}$ by restricting the possible directions of the flooding. However, the purpose of this experiment is also to demonstrate that such a mechanism is implicitly provided by the proposed $\mathrm{M}_{\mathrm{hier}}$, illustrating its self-organizing capabilities.

The increasing efficiency of our method when the number of primary nodes is reduced has an important practical consequence. In real-life scenarios we can expect that the proportion of *primary nodes* in comparison to the total number of nodes in the network will be relatively small. Those nodes consist of cameras that are expected to encounter new objects. In many cases, where the network of cameras covers a large area, there are only a few *entry points* where unknown objects may enter the observed area. For example, if a large, multi-storey building is covered by such a network, only cameras that cover the entrances to the building would encounter new people entering the building. All the other cameras would discover that the people they observe have already been seen by the network, and would not perform learning. In this situation we expect that a hierarchical feature distribution in such a network would be even more efficient. For this reason, we performed a preliminary test in a basic surveillance scenario as well.

D. Test in a basic surveillance scenario

A preliminary test was performed to evaluate the performance of the proposed framework in a slightly more realistic scenario. A dataset, consisting of 226 different images of four persons and two cars, was acquired. A few images from the dataset are shown in Fig. 4. The images were taken outdoors, from different viewpoints. They were cropped manually and therefore include only the objects of interest and relatively small patches of the background. A synthetic scenario was devised to test the framework, as follows. The scenario assumes that there exists a network of 99 4-connected cameras



(a) Ratio 20:99 (b) Ratio 10:99 (c) Ratio 4:99 (d) Ratio 2:99 (e) Ratio 1:99

Fig. 3. Different distributions of primary nodes. Black discs correspond to nodes that have received images during learning, primary nodes, and white discs correspond to all the others. The network consists of 99 nodes. Figs. (a)–(e) show the same network with a decreasing number of primary nodes.

TABLE III
EFFECT OF THE NUMBER OF THE PRIMARY NODES

Matching method	Ratio	Learning phase				Matching phase			
		Hops [$\frac{1}{sample}$]		Traffic [$\frac{kbytes}{sample}$]		Hops [$\frac{1}{sample}$]		Traffic [$\frac{kbytes}{sample}$]	
		$M_{flood-at-learn}$	M_{hier}	$M_{flood-at-learn}$	M_{hier}	$M_{flood-at-match}$	M_{hier}	$M_{flood-at-match}$	M_{hier}
Template	20:99	614	614	2855	41	398	279	4132	2228
	10:99	614	614	2855	41	398	258	4132	1894
	4:99	614	614	2855	42	397	209	4132	1118
	2:99	614	614	2855	43	387	156	4132	425
	1:99	614	614	2855	43	369	116	4132	81
Histogram	20:99	614	614	363	25	432	309	521	273
	10:99	614	614	363	25	436	292	521	231
	4:99	614	614	363	26	432	244	521	140
	2:99	614	614	363	27	420	189	521	54
	1:99	614	614	363	29	398	145	520	11
PCA	20:99	614	614	719	44	381	243	1036	486
	10:99	614	614	719	44	381	225	1036	412
	4:99	614	614	719	45	380	189	1036	267
	2:99	614	614	719	47	371	139	1036	104
	1:99	614	614	719	51	355	102	1036	21
Haar 2D	20:99	614	614	719	44	376	251	1036	536
	10:99	614	614	719	44	377	232	1036	455
	4:99	614	614	719	45	375	187	1036	280
	2:99	614	614	719	47	367	135	1036	107
	1:99	614	614	719	51	352	98	1036	21

in the same arrangement as in the previous experiments, now covering the building and the neighboring area, including a parking lot. There are six cameras covering the three entrances: two are covering the entrance to the parking lot, two are covering the front entrance, and two are covering the back entrance to the building. The experiment followed the basic structure described in Section V-B. In the learning phase, two persons appear at the front entrance, two at the back entrance and two cars appear at the parking-lot entrance. Six images of those six objects were injected at the corresponding nodes according to this scenario. In the matching phase, all the objects appear randomly throughout the building and its surroundings 1,500 times. Histogram matching was used as the matching method. The results are presented in Table IV and are consistent with experiments on the COIL-100 database.

Fig. 4. Several representative images from the surveillance dataset, one per object.

VI. CONCLUSION

This paper focuses on an important conceptual problem of mapping object-matching methods to VSNs, specifically the issue of knowledge storage and propagation. We propose a hierarchical, feature-distribution scheme that guarantees the *visibility* of any feature vector from any node in the network with only a fraction of the network load that the full distribution of feature vectors would cause. To achieve the same matching performance as with the full distribution of feature vectors we propose a set of requirements regarding abstraction, storage space, similarity metric and convergence. This set of requirements has to be fulfilled by the object-matching method in order for it to be used in our framework. Our framework is based on a hierarchical encoding of the visual knowledge, where the node that originally captures the visual knowledge retains complete information about the object. All the other nodes in the network receive less-detailed (more abstract) visual knowledge: in our case, shorter feature vectors. These feature vectors are then used for the routing of queries only to those nodes that have complete feature vectors available in their local storage. In this way, a final decision on the identity of the known objects is made using complete feature vectors. On the other hand, queries travelling in the directions where there are no matching feature vectors are dropped along the

TABLE IV
PRELIMINARY EXPERIMENTAL RESULTS IN A BASIC SURVEILLANCE SCENARIO

Matching method	Distribution method	Learning Hops $[\frac{1}{sample}]$	Learning Traffic $[\frac{kbytes}{sample}]$	Matching Hops $[\frac{1}{sample}]$	Matching Traffic $[\frac{kbytes}{sample}]$	Total traffic* $[Mbytes]$	Matching rate FPs**	Matching rate FNs**
Histogram	$\mathrm{M}_{\mathrm{flood-at-match}}$	0	0	273	519	778	25%	22%
	$\mathrm{M}_{\mathrm{flood-at-learn}}$	614	363	0	0	2	25%	22%
	$\mathbf{M_{hier}}$	**614**	**25**	**54**	**80**	**120**	**25**%	**22**%

* Note that total traffic depends on the number of samples used. It is based on 6 learning and 1,500 testing samples.
** FPs and FNs denote the percentage of false positives and false negatives, respectively.

way. To show the performance of our approach, which aims to reduce the amount of traffic transmitted, while still preserving the matching performance, four object-matching methods were selected. For these methods we prove that they satisfy the four requirements of our feature-distribution method. The proposed distribution was compared with two flood-feature distributions using our network simulator. The proposed hierarchical feature distribution outperformed both flood-based feature distributions, without any degradation in the matching performance. Nevertheless, both the flood-based distribution methods can be interpreted as two extreme operating points of the hierarchical feature-distribution method. $\mathrm{M}_{\mathrm{flood-at-learn}}$ can be seen as a variant of $\mathrm{M}_{\mathrm{hier}}$ with a maximum abstraction level of $n = 0$ (no abstraction allowed), and $\mathrm{M}_{\mathrm{flood-at-match}}$ can be interpreted as a case of $\mathrm{M}_{\mathrm{hier}}$ with a mapping $f : \mathbf{x}^{(n)} \mapsto 0$, and a metric $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}) = 0, n > 0$. $\mathrm{M}_{\mathrm{hier}}$ can be positioned between these two extrema by varying its parameters, such as the maximum level of abstraction and the amount of discarded information at each hop. In future, we plan to study different operating points of the proposed hierarchical feature-distribution scheme.

Four object-matching methods that have been used to illustrate the performance of a hierarchical feature-distribution scheme might not work on real-life object images. Nevertheless, those methods also form the foundation of many state-of-the-art algorithms. For example, PCA is widely used and can be applied to various kinds of features. Histograms are widely used for color matching and also form the basis of other advanced image features, such as SIFT [38] and HOG [39] descriptors. Therefore, our hierarchical feature-distribution scheme can be directly applied to several state-of-the-art matching methods with little or no adaptation.

Our experimental setup assumed a single view of an object. Nevertheless, the proposed framework could be extended to a multi-view case as well. Assuming spatially calibrated and temporally synchronized cameras, additional spatio-temporal information about the observed object could be distributed along with the feature vectors. Using this additional information, each individual node could establish a spatio-temporal correspondence between a locally observed object and the features distributed by other node(s). Based on this correspondence, a common identity could be assigned to multiple views of a same object. In this way the network would build a more complex, distributed, multi-view representation of an object, while still preserving the advantages of the proposed, hierarchical, feature-distribution scheme. An alternative approach would be the introduction of *supernodes* [23] or *fusion sensors* [24] into the network structure. The task of such nodes would be to aggregate the information from spatially related nodes. In addition to those issues, the future challenge lies in mapping state-of-the-art matching and recognition methods to such a distributed framework. When using the proposed scheme with state-of-the-art matching methods it may also happen that the requirements of our scheme would not be fulfilled *in general*. However, even if those requirements are satisfied *most of the time*, we expect that the use of such methods in our hierarchical scheme would result in only a minor decrease in the matching performance. This will be the focus of our future research.

APPENDIX A
PROOF OF THE REQUIREMENT 4 FOR TEMPLATE MATCHING WHEN USING EUCLIDEAN DISTANCE

Let us assume that we have two images $A^{(n)}$ and $B^{(n)}$ with dimensions $k \times k$. Requirement 4 is fulfilled if the next inequality holds,

$$\begin{aligned} & d^{(n)}(\mathbf{x}_A^{(n)}, \mathbf{x}_B^{(n)}) \geq d^{(n+1)}(\mathbf{x}_A^{(n+1)}, \mathbf{x}_B^{(n+1)}), \\ & \sqrt{\sum_{i=1}^{k}\sum_{j=1}^{k}(A^{(n)}(i,j) - B^{(n)}(i,j))^2} \\ & \qquad \geq \sqrt{\sum_{i=1}^{k/2}\sum_{j=1}^{k/2}(A^{(n+1)}(i,j) - B^{(n+1)}(i,j))^2}, \\ & \sum_{i=1}^{k}\sum_{j=1}^{k}(A^{(n)}(i,j) - B^{(n)}(i,j))^2 \\ & \qquad \geq \sum_{i=1}^{k/2}\sum_{j=1}^{k/2}(A^{(n+1)}(i,j) - B^{(n+1)}(i,j))^2, \end{aligned} \tag{5}$$

where $\mathbf{x}_A^{(n)}$, $\mathbf{x}_B^{(n)}$, $\mathbf{x}_A^{(n+1)}$, $\mathbf{x}_B^{(n+1)}$ are the level n and $n+1$ feature vectors, respectively.

The level $n+1$ feature vectors are obtained using the mapping f, which in our case is a simple subsampling operation that reduces the image dimensions by calculating 2×2 pixels averages. The inequality (5) with the superscript n (on both

sides) omitted can be rewritten as

$$\begin{aligned}
&\sum_{i=1}^{k/2}\sum_{j=1}^{k/2}\big[(A(2i-1,2j-1)-B(2i-1,2j-1))^2\\
&\quad+(A(2i-1,2j)-B(2i-1,2j))^2\\
&\quad+(A(2i,2j-1)-B(2i,2j-1))^2\\
&\quad+(A(2i,2j)-B(2i,2j))^2\big]\\
&\geq\sum_{i=1}^{k/2}\sum_{j=1}^{k/2}\Bigg[\Bigg(\frac{A(2i-1,2j-1)+A(2i-1,2j)}{4}\\
&\quad+\frac{A(2i,2j-1)+A(2i,2j)}{4}\Bigg)\\
&\quad-\Bigg(\frac{B(2i-1,2j-1)+B(2i-1,2j)}{4}\\
&\quad+\frac{B(2i,2j-1)+B(2i,2j)}{4}\Bigg)\Bigg]^2. \quad (6)
\end{aligned}$$

The inequality (6) is always satisfied if for $(i,j=1,\ldots,\frac{k}{2})$

$$\begin{aligned}
&(A(2i-1,2j-1)-B(2i-1,2j-1))^2\\
&\quad+(A(2i-1,2j)-B(2i-1,2j))^2\\
&\quad+(A(2i,2j-1)-B(2i,2j-1))^2\\
&\quad+(A(2i,2j)-B(2i,2j))^2\\
&\geq[\frac{A(2i-1,2j-1)+A(2i-1,2j)+A(2i,2j-1)+A(2i,2j)}{4}\\
&-\frac{B(2i-1,2j-1)+B(2i-1,2j)+B(2i,2j-1)+B(2i,2j)}{4}]^2.
\end{aligned}$$

Introducing $x_1=A(2i-1,2j-1)-B(2i-1,2j-1)$, $x_2=A(2i-1,2j)-B(2i-1,2j)$, $x_3=A(2i,2j-1)-B(2i,2j-1)$, $x_4=A(2i,2j)-B(2i,2j)$ the inequality for an arbitrary term (i,j) can be rewritten as follows

$$x_1^2+x_2^2+x_3^2+x_4^2\geq(\frac{x_1+x_2+x_3+x_4}{4})^2. \quad (7)$$

To prove that inequality (7) is satisfied, we may use Jensen's inequality for the mean, which states that $E(f(x))\geq f(E(x))$, where f is any convex function – a square function in our case. Using Jensen's inequality it is obvious that $4\cdot E(f(x))=4\cdot\frac{x_1^2+x_2^2+x_3^2+x_4^2}{4}$ and $f(E(x))=(\frac{x_1+x_2+x_3+x_4}{4})^2$, which implies that Requirement 4 is fulfilled.

APPENDIX B
PROOF OF THE REQUIREMENT 4 FOR HISTOGRAM MATCHING WHEN USING HELLINGER DISTANCE

Let us assume that we have two images $A^{(n)}$ and $B^{(n)}$ with the corresponding histograms $h_A^{(n)}$ and $h_B^{(n)}$. Requirement 4 is fulfilled if the next inequality holds:

$$\begin{aligned}
d^{(n)}(\mathbf{x}_A^{(n)},\mathbf{x}_B^{(n)}) &\geq d^{(n+1)}(\mathbf{x}_A^{(n+1)},\mathbf{x}_B^{(n+1)}), \quad (8)\\
\sqrt{1-\rho(h_A^{(n)},h_B^{(n)})} &\geq \sqrt{1-\rho(h_A^{(n+1)},h_B^{(n+1)})},\\
\rho(h_A^{(n)},h_B^{(n)}) &\leq \rho(h_A^{(n+1)},h_B^{(n+1)}),
\end{aligned}$$

where $\mathbf{x}_A^{(n)}$, $\mathbf{x}_B^{(n)}$, $\mathbf{x}_A^{(n+1)}$, $\mathbf{x}_B^{(n+1)}$ are the level n and $n+1$ feature vectors, respectively, and $h_A^{(n)}$, $h_B^{(n)}$, $h_A^{(n+1)}$, $h_B^{(n+1)}$ are the level n an $n+1$ histograms, respectively.

Feature vectors and level $n+1$ histograms are obtained using the mapping f, which combines adjoining bins. Substituting $\rho(h_A^{(n)},h_B^{(n)})$ with $\sum_{i=1}^{P}\sqrt{h_{iA}^{(n)}h_{iB}^{(n)}}$, with the superscripts n and $n+1$ omitted and introducing $K=\frac{P}{2}$ (we assume that there is always an even number of bins), the inequality can be rewritten as

$$\begin{aligned}
&\sum_{i=1}^{K}(\sqrt{h_{(2i-1)A}h_{(2i-1)B}}+\sqrt{h_{2iA}h_{2iB}})\\
&\quad\leq\sum_{i=1}^{K}\sqrt{(h_{(2i-1)A}+h_{2iA})(h_{(2i-1)B}+h_{2iB})}. \quad (9)
\end{aligned}$$

The inequality (9) holds, if

$$\begin{aligned}
&\sqrt{h_{(2i-1)A}h_{(2i-1)B}}+\sqrt{h_{2iA}h_{2iB}}\\
&\quad\leq\sqrt{(h_{(2i-1)A}+h_{2iA})(h_{(2i-1)B}+h_{2iB})}, \quad (10)
\end{aligned}$$

where $(i=1,\ldots,K)$. For $i=1$

$$\begin{aligned}
&\sqrt{h_{1A}h_{1B}}+\sqrt{h_{2A}h_{2B}}\\
&\quad\leq\sqrt{(h_{1A}+h_{2A})(h_{1B}+h_{2B})},\\
&h_{1A}h_{1B}+2\sqrt{h_{1A}h_{1B}}\sqrt{h_{2A}h_{2B}}+h_{2A}h_{2B}\\
&\quad\leq(h_{1A}+h_{2A})(h_{1B}+h_{2B}),\\
&h_{1A}h_{1B}+2\sqrt{h_{1A}h_{1B}}\sqrt{h_{2A}h_{2B}}+h_{2A}h_{2B}\\
&\quad\leq h_{1A}h_{1B}+h_{1A}h_{2B}+h_{2A}h_{1B}+h_{2A}h_{2B},\\
&2\sqrt{h_{1A}h_{1B}}\sqrt{h_{2A}h_{2B}}\leq h_{1A}h_{2B}+h_{2A}h_{1B},\\
&0\leq(\sqrt{h_{1A}h_{2B}}-\sqrt{h_{2A}h_{1B}})^2, \quad (11)
\end{aligned}$$

where $h_{1A},h_{2A},h_{1B},h_{2B}$ are histogram bins and are always $h_{1A},h_{2A},h_{1B},h_{2B}\geq 0$. The inequality holds since $0\leq(\sqrt{h_{1A}h_{2B}}-\sqrt{h_{2A}h_{1B}})^2$ always holds. By applying a similar argument for all i in (10) it is obvious that Requirement 4 is fulfilled.

REFERENCES

[1] S. Misra, M. Reisslein, and G. Xue, "A survey of multimedia streaming in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 18–39, 2008.

[2] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Comput. Networks*, vol. 51, no. 4, pp. 921–960, 2007.

[3] H. Aghajan and A. Cavallaro, *Multi-Camera Networks, Principles and Applications*. Elsevier, 2009.

[4] B. Rinner and W. Wolf, "A bright future for distributed smart cameras," *Proc. IEEE*, vol. 96, no. 10, pp. 1562–1564, 2008.

[5] Y. Charfi, N. Wakamiya, and M. Murata, "Challenging issues in visual sensor networks," *IEEE Wirel. Commun.*, vol. 16, no. 2, pp. 44–49, 2009.

[6] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Adv. Multimedia*, 2009, article ID 640386, doi:10.1155/2009/640386.

[7] C. Arth, "Visual surveillance on DSP-based embedded platforms," Ph.D. dissertation, Institute for Computer Graphics and Vision, Graz University of Technology, 2008.

[8] L. Tessens, M. Morbee, W. Philips, R. Kleihorst, and H. Aghajan, "Efficient approximate foreground detection for low-resource devices," in *ACM/IEEE Int. Conf. Distr. Smart Cameras (ICDSC)*, 2009.

[9] M. Bramberger, R. P. Pflugfelder, A. Maier, B. Rinner, B. Strobl, and H. Schwabach, "A smart camera for traffic surveillance," in *Proc. Works. Intell. Solutions in Embedded Syst. (WISES)*, 2003, pp. 153–164.

[10] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl, "Autonomous multicamera tracking on embedded smart cameras," *EURASIP J. Embedded Syst.*, 2007, article ID 92827, doi:10.1155/2007/92827.

[11] X. Wang, S. Wang, D.-W. Bi, and J.-J. Ma, "Distributed peer-to-peer target tracking in wireless sensor networks," *Sensors*, vol. 7, no. 6, pp. 1001–1027, 2007.

[12] S. Fleck, F. Busch, P. Biber, and W. Strasser, "3D surveillance – a distributed network of smart cameras for real-time tracking and its visualization in 3D," in *Comput. Vision and Patt. Rec. Workshop (CVPRW)*, 2006.

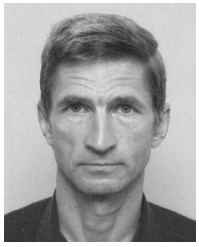
[13] C. Arth and H. Bischof, "Real-time object recognition using local features on a DSP-based embedded system," *J. Real-Time Image Process.*, vol. 3, pp. 233–253, 2008.

[14] W. Yu, Z. Sahinoglu, and A. Vetro, "Energy efficient jpeg 2000 image transmission over wireless sensor networks," in *IEEE Global Telecomm. Conf. (GLOBECOM)*, vol. 5, 2004, pp. 2738–2743.
[15] D. U. Lee, H. Kim, M. Rahimi, D. Estrin, and J. D. Villasenor, "Energy-efficient image compression for resource-constrained platforms," *IEEE Trans. Image Process.*, vol. 18, no. 9, pp. 2100–2113, 2009.
[16] V. Lecuire, C. Duran-Faundez, and N. Krommenacker, "Energy-efficient image transmission in sensor networks," *Int. J. Sensor Networks*, vol. 4, no. 1/2, pp. 37–47, 2008.
[17] H. Wu and A. A. Abouzeid, "Error resilient image transport in wireless sensor networks," *Comput. Networks*, vol. 50, no. 15, pp. 2873–2887, 2006.
[18] Y. Charfi, N. Wakamiya, and M. Murata, "Trade-off between reliability and energy cost for content-rich data transmission in wireless sensor networks," in *Int. Conf. Broadband Comm. Networks and Syst. (BROADNETS)*, 2006, pp. 1–8.
[19] E. Simmons, E. Ljung, and R. Kleihorst, "Distributed vision with multiple uncalibrated smart cameras," in *Worksh. on Distr. Smart Cameras (in conj. with ACM SenSys)*, 2006, pp. 67–72.
[20] D. Devarajan and R. J. Radke, "Calibrating distributed camera networks using belief propagation," *EURASIP Journal on Applied Signal Processing*, 2007, article ID 60696, doi:10.1155/2007/60696.
[21] S. N. Sinha and M. Pollefeys, "Synchronization and calibration of camera networks from silhouettes," in *Int. Conf. Patt. Recog. (ICPR)*, 2004, pp. 116–119.
[22] Z. Cheng, D. Devarajan, and R. J. Radke, "Determining vision graph for distributed camera networks using feature digests," *EURASIP J. Adv. Sig. Pr.*, 2007, article ID 57034, doi:10.1155/2007/57034.
[23] T. Kirishima, Y. Manabe, K. Sato, and K. Chihara, "Real-time multiview recognition of human gestures by distributed image processing," *EURASIP J. Image Video Process.*, 2010, article ID 517861, doi:10.1155/2010/517861.
[24] Z. Zhang, A. Scanlon, W. Yin, L. Yu, and P. L. Venetianer, *Multi-Camera Networks: Principles and Applications*. Elsevier, 2009, ch. Video Surveillance Using a Multi-Camera Tracking and Fusion System, pp. 435–456.
[25] A. T. Ihler, "Inference in sensor networks: Graphical models and particle methods," Ph.D. dissertation, Massachusetts Institute of technology, 2005.
[26] V. Sulić, J. Perš, M. Kristan, and S. Kovačič, "Phd forum: Hierarchical feature scheme for object recognition in visual sensor networks," in *ACM/IEEE Int. Conf. Distr. Smart Cameras (ICDSC)*, 2009.
[27] M. A. Patricio, J. Carbo, O. Perez, J. Garcia, and J. M. Molina, "Multi-agent framework in visual sensor networks," *EURASIP J. Adv. Sig. Pr.*, 2007, article ID 98639, doi:10.1155/2007/98639.
[28] A. Gilbert and R. Bowden, "Incremental, scalable tracking of objects inter camera," *Comput. Vis. Image Underst.*, vol. 111, no. 1, pp. 43–58, 2008.
[29] R. Chang, S.-H. Ieng, and R. Benosman, "Auto-organized visual perception using distributed camera network," *Robot. Auton. Syst.*, vol. 57, no. 11, pp. 1075–1082, 2009.
[30] C. Leistner, P. Roth, H. Grabner, H. Bischof, A. Starzacher, and B. Rinner, "Visual on-line learning in distributed camera networks," in *ACM/IEEE Int. Conf. Distr. Smart Cameras (ICDSC)*, 2008.
[31] M. Karakaya and H. Qi, "Target detection and counting using a progressive certainty map in distributed visual sensor networks," in *ACM/IEEE Int. Conf. Distr. Smart Cameras (ICDSC)*, 2009.
[32] J. Park, P. C. Bhat, and A. C. Kak, "A look-up table based approach for solving the camera selection problem in large camera networks," in *Worksh. on Distr. Smart Cameras (in conj. with ACM SenSys)*, 2006, pp. 72–77.
[33] A. Y. Yang, S. Maji, C. M. Christoudias, T. Darell, J. Malik, and S. S. Sastry, "Multiple-view object recognition in band-limited distributed camera networks," in *ACM/IEEE Int. Conf. Distr. Smart Cameras (ICDSC)*, 2009.
[34] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," Tilburg University, Tech. Rep. TiCC-TR 2009-005, 2009.
[35] M. Kristan, J. Perš, M. Perše, and S. Kovačič, "Closed-world tracking of multiple interacting targets for indoor-sports applications," *Comput. vis. Image Underst.*, no. 113, pp. 598–611, 2009.
[36] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vis.*, vol. 7, pp. 11–32, 1991.
[37] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-100)," Department of Computer Science, Columbia University, Tech. Rep. CUCS-006-96, 1996.
[38] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[39] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Conf. Comput. Vision Patt. Recog. (CVPR)*, vol. 1, 2005, pp. 886–893.

Vildana Sulić received her B.Sc. degree from the Faculty of Electrical Engineering of the University of Ljubljana, Slovenia in 2006, in the field of biomedical engineering. Currently she is a Ph.D. student and junior researcher at the Machine Vision Laboratory at the Faculty of Electrical Engineering, with her interests in computer vision, image processing, human-motion analysis and visual-sensor networks.

Janez Perš received B.Sc., M.Sc. and Ph.D. degrees in Electrical engineering at the Faculty of Electrical Engineering, University of Ljubljana, in 1998, 2001 and 2004, respectively. He is currently an assistant professor at the Machine Vision Laboratory at the Faculty of Electrical Engineering, University of Ljubljana. His research interests lie in image-sequence processing, object tracking, human-motion analysis, dynamic-motion-based biometry, and in autonomous and distributed systems.

Matej Kristan received the Dipl.ing., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2003, 2005, and 2008, respectively. He is currently an Assistant Professor at the Machine Vision Laboratory, Faculty of Electrical Engineering, University of Ljubljana and a Researcher with the Visual Cognitive Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana. His research interests include probabilistic methods for computer vision and pattern recognition with focus on tracking, probabilistic dynamic models, online learning and mobile robotics. Dr. Kristan has received several awards for his research in the field of computer vision and pattern recognition.



Stanislav Kovačič is a professor, a vice dean and the head of the Laboratory for Machine Vision at the Faculty of Electrical Engineering, University of Ljubljana. He was a visiting researcher in the GRASP Laboratory at the University of Pennsylvania, the Technische Fakultät der Friedrich-Alexander-Universität in Erlangen, and at the Faculty of Electrical Engineering and Computing, University of Zagreb. His research is focused on various aspects of image and video analysis with applications in medicine, industry and sports.