Technical Report FE-LSV-01/09
Efficient Feature Distribution in Visual Sensor Networks

Vildana Sulić, Janez Perš, Matej Kristan and Stanislav Kovačič

Faculty of Electrical Engineering, University of Ljubljana
Trzaska 25, SI-1000 Slovenia
e-mail: vildana.sulic@fe.uni-lj.si

Note: Extended version of this report submitted to journal.

Abstract

In this paper, we propose a framework for hierarchical feature distribution scheme for object recognition in a distributed environment, e.g. in a network of visual sensors. The key of our approach is the principle that individual nodes in the network hold only a small amount of information about objects seen by the network, however, this small amount is sufficient to efficiently route queries through the network. We outline a set of requirements, which have to be fulfilled by the object recognition method to be used in our framework. We provide examples of three simple object recognition methods, which can be adapted for use in such feature distribution scheme, and we also present their implementations which fulfill those requirements. The proposed approach has been tested using our own distributed network simulator on a standard COIL-20 database. Preliminary results suggest that amount of data transmitted through the network can be significantly reduced in comparison to naive feature distribution schemes.

1 Introduction

One of the main characteristics of computer vision algorithms is a large amount of data that has to be processed, stored or transmitted. Usually a state-of-the-art hardware is employed to deal with the requirements that go along with processing of digital images and digital video streams. In terms of sensor network topology, such approach usually leads to a *star* network structure – a single powerful processing unit and one or more sensors. In the majority of applications, these sensors are simply ordinary digital or analog cameras.

Many applications, such as video surveillance, traffic and environment monitoring, need a large number of visual sensors. In some cases the number of cameras is large, where in extreme cases it may go into hundreds (e.g. 230 for London Congestion Charge monitoring), and the requirements for transmitting and processing such amounts of data are correspondingly large. For such systems, a decentralized, distributed network structure would be a natural choice.

Transition to a distributed network topology of visual sensors is not straightforward. Due to a high processing, transmission and storage requirements of many computer vision algorithms, the distributed network would be put under significant strain, if such algorithms are directly mapped to the network. Let us, for example consider a frequent task in computer vision – detection and recognition of objects that have been previously seen by any of the sensors (cameras). In a distributed system, one of the following two scenarios would appear:

- Captured visual information (images or extracted object features) from each sensor are distributed to all nodes for local storage. Future detection of similar objects is then performed locally by each sensor as new images are acquired.

- Captured visual information is stored locally. However, each task of finding an object is broadcasted across the network, for comparison with the locally stored information on each node, for each new image acquired.

In the first scenario, large amount of data would have to be transmitted across the network for each object learned. Storage requirements for the individual nodes are similar to the central unit in a star network structure. In the second scenario, large amount of data would have to be transmitted across the network for each new object encountered, as every node has to be supplied with newly acquired image or object features to find a possible match. In this case, each node would have to handle the similar amount of traffic than the central node in the star network structure.

This problem may be tackled by encoding information in a hierarchical way, where each node stores only part of information about objects previously seen by the network. This information can be low in volume and serves only to efficiently route network queries, when a new object is encountered by any of the nodes (e.g. when new image is acquired).

Not all recognition methods can be used in that way. In this paper we outline a set of criteria, which have to be fulfilled by object recognition method, to be considered for use in such framework.

In the remainder of the paper we describe the concept of such hierarchical feature distribution scheme. Algorithms for efficient feature distribution and distributed object recognition are proposed. Next, three very simple object recognition methods are analyzed and shown to fulfill the stated requirements. Finally, results of tests in application-layer distributed network simulator are presented and obtained results are discussed.

1.1 Related work

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

This issue is not unique to typical VSNs. With widespread use of digital cameras, prevalence of wireless connectivity and an integration of visual sensors in objects of personal use, like mobile phones or PDAs, the technology of VSNs has become widespread as well [1]. In addition to this, the technology of VSNs is applicable to other fields of science and engineering, such as mobile robotics, where many robots now include low cost imaging sensors and wireless communication solutions. Therefore, the popularity of the VSN research is not surprising.

In [2], an object recognition system with real time capabilities for deployment on a DSP-based embedded platform is proposed. In [23], authors proposed a system for target detection, classification and tracking. Those tasks are performed in each sensor node. In [18], authors applied a multi-agent framework to control the capture parameters of a surveillance system using a VSN. In [10], authors proposed a technique for tracking objects across spatially separated, uncalibrated, non-overlapping cameras. In [11] a new technique for tracking objects across uncalibrated, non-overlapping cameras is described as well. An autonomous multicamera tracking method on distributed, embedded smart cameras is presented in [19]. In [3] an implementation of a smart camera for traffic surveillance is described. In [9], authors present a distributed network of smart cameras for real-time tracking and the corresponding visualization in 3D. Person detection system based on distributed smart cameras is presented in [14], where authors show that using on-line learning classifiers on smart cameras can reduce memory and bandwidth requirements and therefore can be appropriate for embedded systems.

One of the major issues in VSNs, is dealing with efficient data transmission between the nodes. Transmission of visual information usually requires a large bandwidth and therefore, specifically tailored optimization to the distributed sensor topologies is highly desirable. Data transmission techniques in VSNs can be roughly categorized into three categories [5]. In the first category, there are efficient image and video transmission methods for transmission over a single hop, such as [13]. In the second category, there are techniques that consider multihop transmission strategy such as [24]. The third category includes end-to-end multi-path transmission techniques. One of many is [6].

The next major issue in VSNs is their distributed nature. Computer vision algorithms have to be either adapted or totally rebuilt to deal with the specifics of distributed networks. For example in [7], authors propose decentralized methods for obtaining the vision graph for a distributed camera network. The algorithm for estimating camera relations in vision system built of distributed, uncalibrated, smart cameras is presented by [20]. A geometry-free method that allows camera networks systems to estimate their topology and auto-organize their own activities according to the content of the scene and the task to be achieved is introduced in [4]. In [8], a fully distributed calibration approach for 3D camera networks, using belief propagation is proposed. Calibration and synchronization of a network of cameras from a set of unsynchronized silhouettes sequences is proposed in [21]. Using distributed look-up tables for selecting a subset of cameras, that is likely to carry out the current requested task most effectively, is solved by [17].

Our approach aims to facilitate an efficient distribution of features to a large number of visual sensors and an efficient routing of queries to those nodes, which have enough information to perform the object recognition. Differently from many other studies, we do not deal with the implementation details or low level issues in a data transmission. Our approach aims to reduce the total amount of data that has to be transmitted across the network or stored in individual nodes, *while retaining exactly the same recognition performance.* We achieve this by encoding features in a hierarchical way, where less descriptive features are derived from the original object features. Those less descriptive features are used for efficient routing, while consuming significantly less network resources than full distribution of original object features.

2 Hierarchical feature distribution scheme for object recognition

Despite discrepancy between computationally intensive image processing in computer vision algorithms and relatively low processing capabilities of nodes in a typical VSN, properly designed VSN may provide significant storage and computing capabilities, but in a highly distributed manner. The algorithms that run on such network have to be aware of this distributed nature to take advantage of these capabilities.

The basic approach to object recognition is as follows:

- Learning phase: compact representation (model) of object is extracted from one or more images and stored. In our case, we assume that such compact representation is in the form of a feature vector.

- Recognition phase: the same compact representation of an object (feature vector, in our case) is extracted from the newly acquired image. This vector is compared to the feature vectors stored during learning phase to obtain a correspondence with one of the learned objects.

In general, a network of visual sensors is expected to encounter a large number of objects. The purpose of learning in such distributed environment would for example be to provide the capability to recognize objects that have been already seen by any of the nodes. Therefore, after a new image is acquired, a node would proceed as follows:

1. Extraction of relevant feature vector from the visual representation of object.

2. Distribution of feature vector to other nodes that comprise VSN.

As soon as sufficient amount of knowledge (e.g. sufficient number of feature vectors) has been obtained by the network, object recognition can be performed simultaneously with learning. In our case, we do not explicitly deal with the concept of incremental learning; as new images are acquired, network *knowledge* increases, but only due to increase in number of stored feature vectors. Already acquired feature vectors remain unchanged.

It is evident that an efficient mechanism to disseminate knowledge (e.g. mechanism for distribution of feature vectors) across the network is needed, as the complexity of this problem increases exponentially with the number of nodes.

2.1 Hierarchical encoding structure

Hierarchy is fundamental principle on which our feature encoding is based. We require that the *primary node* (the visual sensor that has originally seen the unknown object) retains complete information about the object (e.g. unmodified feature vector). Its neighbors receive less detailed, more abstract information, which, in general, requires less storage space and less transmission capacity. This way, the amount of data transmitted across or stored in the network can be significantly reduced.

Specifically, for an object recognition task such structure requires that the feature vectors x, which are passed across the network, fulfill the four major requirements.

Requirement 1 (Abstraction): There exists a mapping $f : x^n \mapsto x^{n+1}$, $0 < n \leq M$, which translates level n feature vector x^n into higher, more abstract level $(n + 1)$ feature vector x^{n+1}, without access to the acquired image. M denotes maximum level of abstraction.

This requirement assumes that the primary node extracts level 0 feature vectors, x^0, directly from the acquired image. Its direct neighbors receive level 1 feature vectors, x^1, their neighbors receive level 2 feature vectors, x^2, and so on. The mapping $f : x^n \mapsto x^{n+1}$ is done on each of the nodes before transmitting the feature vectors x^{n+1} to its neighbors, until the maximum level of abstraction is reached. From this point on, feature vectors are forwarded unchanged. Maximum level of abstraction can be recognition method dependent (e.g. at some point it may become impossible to further reduce the dimensionality of the feature vector). Other important consideration in selecting M is network transmission overhead – from certain point on reduction in features does not meaningfully reduce the length of the network packets.

Requirement 2 (Storage): If $I(x)$ is the storage space required for the feature vector x in bits, then it should hold that: $I(x^n) \geq I(x^{n+1})$. If this requirement is not fulfilled (in a significant way, e.g. the required storage space on the level $n+1$ has to be *significantly* smaller), the hierarchical encoding scheme does not provide any improvements in the terms of network traffic and storage requirements.

Requirement 3 (Existence of metric): There exists a metric $d^n(x_1^n, x_2^n)$, which provides a measure of *similarity* between two feature vectors x_1^n and x_2^n of the same level n.

The existence of the metric is essential both for the object recognition itself and for the hierarchical feature encoding scheme. The distance $d^n(x_1^n, x_2^n)$, when compared to the threshold T, determines if the objects are *similar*, $d^n(x_1^n, x_2^n) \leq T$, or not, $d^n(x_1^n, x_2^n) > T$.

Requirement 4 (Convergence): Given two vectors x_1^n and x_2^n which are similar, $d^n(x_1^n, x_2^n) \leq T$, the corresponding vectors on the next level $n+1$ should be at least as similar as the vectors on the previous level, $d^{n+1}(x_1^{n+1}, x_2^{n+1}) \leq d^n(x_1^n, x_2^n)$. If this requirement is violated, then the higher level features cannot be used to route the network queries, since it is possible, that at some point, the higher level feature x_{n+1} is above the threshold, while the lower level feature x_n is below the threshold. Given the object recognition algorithm in Section 2.3, that would cause a node with feature x_{n+1} blocking the access to the node which may have matching features x_n.

2.2 Propagation of feature vectors

For a moment, 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, feature vector is extracted and random identification number (ID) is generated and attached to the feature vector. Assuming proper length (e.g. 32 bits), an ID can safely be assumed to be unique for reasonably-sized network. The actual procedure of a feature extraction depends on the recognition method used. The extracted vector is then stored locally and marked as being level 0 vector, x^0. Then, using the mapping $f : x^n \mapsto x^{n+1}$, the next level feature vector x^1 is prepared, assigned the same ID and broadcasted to all 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 x^1 require less storage space than the vectors x^0.

The process of applying the mapping $f : x^n \mapsto x^{n+1}$ is repeated on each node, until every node has at least *some* information about the object seen by the primary node. Communication between nodes and unique IDs ensure that the nodes refuse to accept any duplicated feature vectors.

2.3 Object recognition

The task of object recognition may be performed concurrently with learning (e.g. the network tries to recognize the object before proceeding with learning). For now, let us assume that the network is in recognition-only mode. The algorithm for object recognition is presented as Algorithm 1. Again, we call the node that has acquired the image the *primary node*.

As it can be seen, the Algorithm 1 is recursive in its nature. In the recognition phase, the primary node generates network *query* packet, if the object has not been seen locally, but it has been seen by any of the other nodes. Query packet contains unmodified level 0 features of an unknown object, and is transmitted to those primary node's neighbors, which provided matching features. Upon receiving the query packet, every node runs the Algorithm 1 from the Line 2 on.

Algorithm 1 : Object recognition

```
Input: Image
Output: Object correspondence
 1: Extract object features.
 2:   // Local search
 3: for All levels in the local storage do
 4:   Apply the mapping $f : x^n \mapsto x^{n+1}$ and calculate the next level feature $x^{n+1}$ from $x^n$.
 5:   Compare $x^{n+1}$ 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 forwarded features 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 primary node.
- The object is known (match on level 0 is found). This result is reported to the primary node.
- The object might have been seen by the network (match found on one of the higher levels).

The effect of this algorithm is that, during the recognition phase, features of an unknown object in the unmodified form (level 0 features) are forwarded from node to node along the trail, left by the propagated feature vectors in the learning phase. If the primary node does not receive any replies from the other nodes, the object is unknown to the network. The efficiency of this approach stems from the fact that features are forwarded only in the direction, where there is a possibility that the object has been seen.

3 Examples

Our hierarchical feature encoding framework does not rely on particular object recognition method. It only provides the requirements, which enable any recognition method to be implemented in a distributed way. To be able to illustrate the performance of our approach, we selected three extremely simple recognition methods. The first two methods, template matching and histogram matching, are trivial and provided for illustrative purposes only. The third one, principal component analysis (PCA), is widely used in many object recognition tasks. Nevertheless, we use PCA in its simplest form. In the following section we illustrate the appropriate mappings f^n and the metrics d^n, for which these three simple methods fulfill the stated requirements and can be therefore used in our framework. Therefore, the *recognition* performance of our hierarchical feature distribution scheme is expected to be exactly the same than with simpler feature distribution schemes.

3.1 Template matching

Let us limit the discussion to the simplest form of template matching – direct comparison of two images of the same dimensions. In this case, the feature vector, representing the object, simply contains pixels values of the original image. Following the Requirement 1, we can define the mapping $f : x^n \mapsto x^{n+1}$ as simple subsampling operation, which reduces image dimensions by calculating 2×2 pixels averages. It is obvious that the Requirement 2 is fulfilled, as the resulting image dimensions are halved and both the image and the corresponding feature vector require only a quarter of the original storage space.

The metric, $d^n(x_1^n, x_2^n)$, could be $(1 - \frac{\gamma^n+1}{2})$, where γ^n is normalized cross-correlation between the subsampled images [15]. It is easy to show that the Requirement 4 holds. The situation is presented in Figure 1.

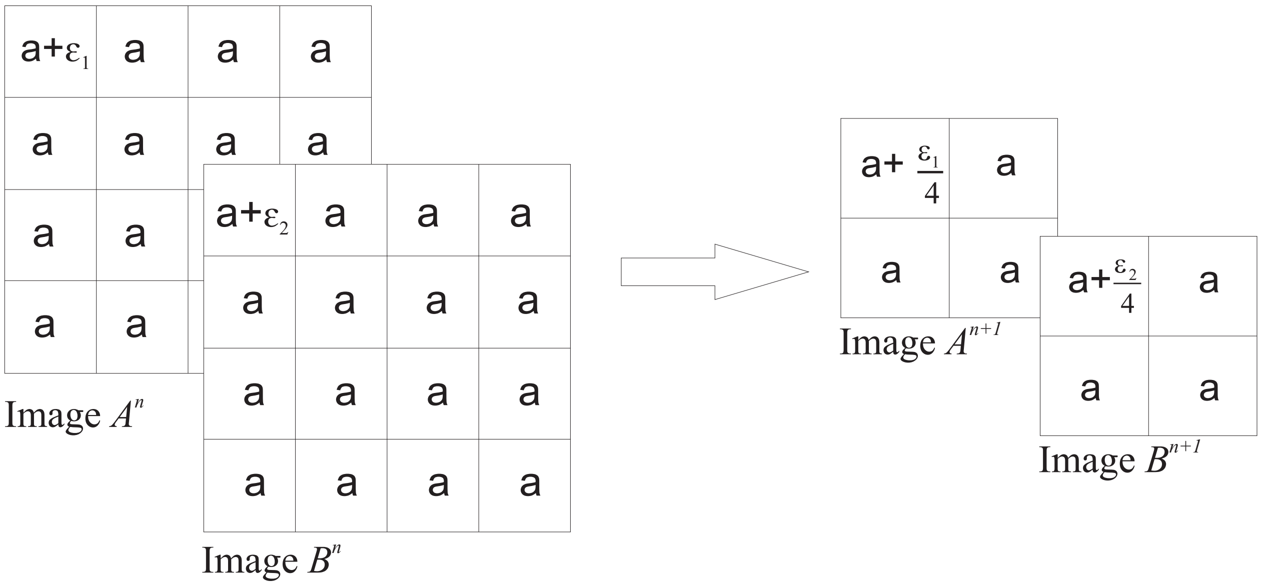

Figure 1: Mapping $f : x^n \mapsto x^{n+1}$ in the case of template matching.

Let us use metric d^n to compare two images that are nearly identical (for the sake of simplicity, let us assume that every pixel has value of a), except for the one pixel in image A^n, which has value $a + \epsilon_1$ and one pixel in image B^n, which has value $a + \epsilon_2$. The original images A^n and B^n have dimensions $m \times m$ and the resized images A^{n+1} and B^{n+1} have the dimensions $m/2 \times m/2$. The metric d^n is then

$$d^n(x_A^n, x_B^n) = d^n(A^n, B^n) = 1 - \tfrac{\gamma^n+1}{2} \tag{1}$$

$$\gamma^n = \frac{\sum_{i,j}^{m}[A^n(i,j)-\overline{A^n}(i,j)]\cdot[B^n(i,j)-\overline{B^n}(i,j)]}{\sqrt{\sum_{i,j}^{m}[A^n(i,j)-\overline{A^n}(i,j)]^2\cdot\sum_{i,j}^{m}[B^n(i,j)-\overline{B^n}(i,j)]^2}},$$

where $\overline{A^n}$ and $\overline{B^n}$ are mean values of the A^n and B^n, respectively. The distances d^n, $d^{n+1}, \ldots, d^M$ are equal to 0, until both images are subsampled to a single pixel, where the metric d^n does not make sense. Since the metric d^n cannot increase with the increasing n, the Requirement 4 is fulfilled. We did not make any assumptions about the value of ϵ_1 and ϵ_2. Similar derivation could be done for different spatial arrangements of ϵ_1 and ϵ_2. Therefore, we can expect that the Requirement 4 is fulfilled regardless of the contents of images A^n and B^n.

3.2 Histogram matching

Intensity and color histograms are a compact representation of an object. Although in practice [12] high dimensional histograms are used, we will limit ourselves to usage of one-dimensional intensity histograms.

Intensity histogram for an 8-bit image may contain up to 256 bins. Each bin contains a normalized count of image pixels within certain range of grey levels. Accordingly, the elements of feature vectors are simply normalized histogram bin counts. Following the Requirement 1, we can define the mapping $f : x^n \mapsto x^{n+1}$ as an operation that combines adjoining bins, giving the coarser representation of original image. Again, it is obvious that the Requirement 2 is fulfilled, since lower number of bins requires less storage space.

The metric, $d^n(x_1^n, x_2^n)$, could be simply the Hellinger distance [12] between the histograms:

$$d^n(x_A^n, x_B^n) = \sqrt{1 - \rho(h_A^n, h_B^n)}, \tag{2}$$

where $d^n(x_A^n, x_B^n)$ is the distance between the feature vectors, x_A^n and x_B^n, $\rho(h_A^n, h_B^n)$ is Bhattacharyya coefficient, $\rho(h_A^n, h_B^n) = \sum_{i=1}^{p} \sqrt{h_{iA}^n h_{iB}^n}$, p is number of bins, which is the same for both histograms, and h_{iA}^n and h_{iB}^n are normalized bin values for i-th bin in histograms h_A^n and h_B^n, respectively.

Given the similar example than in previous section, let us assume that image B^n contains only pixels with value of a, and image A^n has one pixel with value of $a + \epsilon$, with the rest of pixels having value of a. The corresponding histograms of images A^n and B^n are shown in the first column of Figure 2. Histogram for image A^n contains two non-zero bins, a and $a + \epsilon$, whereas histogram for image B^n contains only one, a.

The distance $d^n(x_A^n, x_B^n)$ for each level n can be calculated as follows. For level 0, where 256-bin histogram is used, the distance is:

$$d^0(x_A^0, x_B^0) = \sqrt{1 - \rho(h_A^0, h_B^0)} = \tag{3}$$

$$= \sqrt{1 - \left(\sqrt{\tfrac{m\cdot(m-1)}{m\cdot m}\cdot\tfrac{m\cdot m}{m\cdot m}} + \sqrt{\tfrac{1}{m\cdot m}\cdot\tfrac{0}{m\cdot m}} + \cdots\right)}$$

$$d^0(x_A^0, x_B^0) = \sqrt{1 - \sqrt{\tfrac{m^2-1}{m^2}}},$$

where m is the dimension of the original (square) images, x_A^0, x_B^0 and h_A^0, h_B^0 are level 0 feature vectors and level 0 histograms, respectively. It can be seen that the distance d^0 is greater than 0. After performing mapping f, we have two possible scenarios. If the bins a and $a + \epsilon$ are adjoining

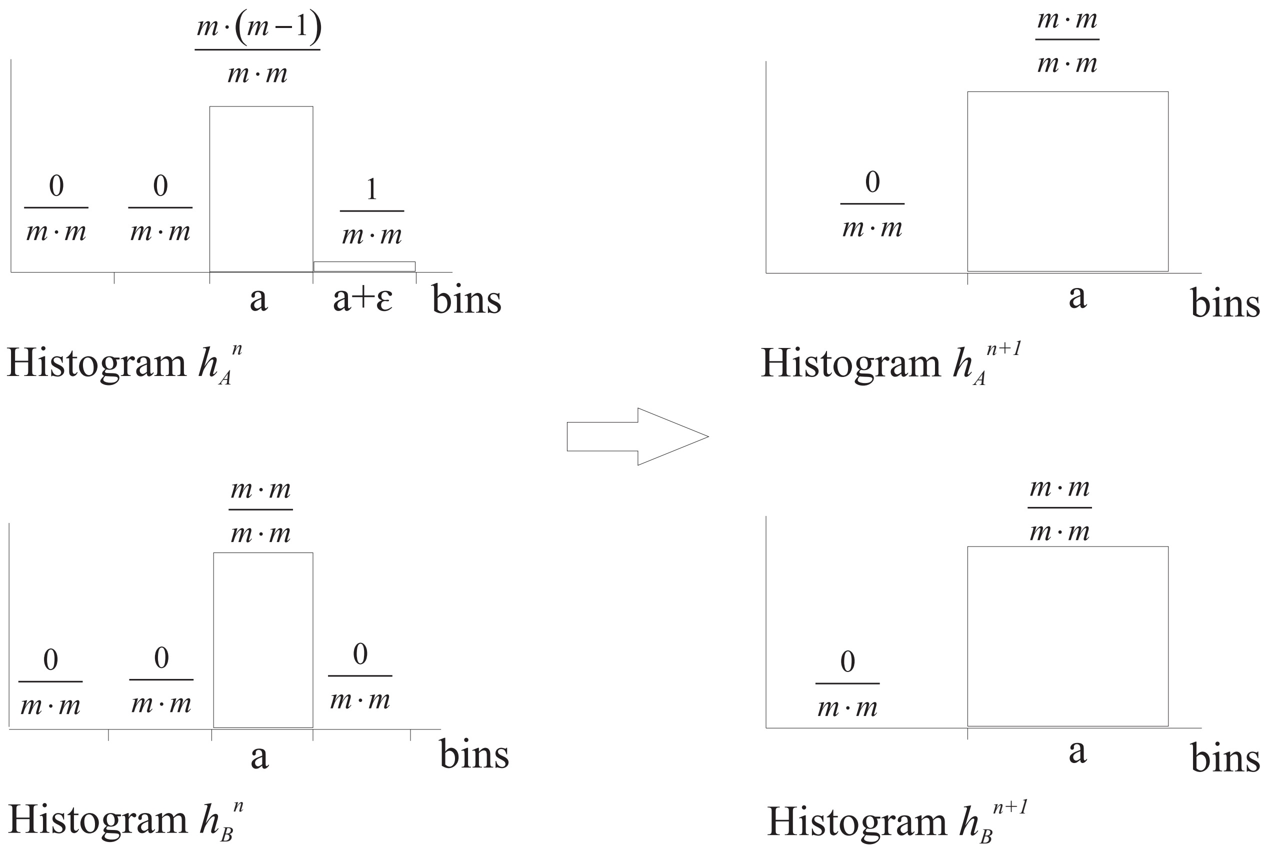

Figure 2: Mapping $f : x^n \mapsto x^{n+1}$ in the case of histogram matching.

bins, the newly calculated histogram h_A^{n+1} will contain only one non-zero bin with the value 1 and it will be the same as histogram h_B^{n+1}, so the distance $\sqrt{1-\rho(h_A^{n+1}, h_B^{n+1})}$ will be 0. In the other scenario, the bins are not adjoining and the mapping f will has no effect on the distance $\sqrt{1-\rho(h_A^{n+1}, h_B^{n+1})}$, which remains greater than 0. Therefore, the Requirement 4 is fulfilled.

3.3 Principal component analysis

Principal component analysis (PCA) is a vector space transform, which reduces the multidimensional data sets to lower dimensions without significant loss of information. In our case, we deal with the simplest variant of the PCA, which means that the eigenvectors are obtained in advance, and remain fixed through the learning and recognition phase.

PCA transforms the data to a new coordinate system in which basis vectors follow modes of greatest variance in data [22]. In essence, properly constructed feature vectors contain feature values, which are already ordered by decreasing importance in terms of reconstruction of the original data.

This opens up a possibility of the mapping function $f : x^n \mapsto x^{n+1}$, which can be defined as dropping the certain number of features of lowest importance from the feature vector. Again, it is obvious that the Requirement 2 holds.

Considering the metric $d^n(x_1^n, x_2^n)$, the Euclidean distance is commonly used when comparing PCA-based feature vectors. It is easy to show that the Requirement 4 holds as well, if the Euclidean distance is used.

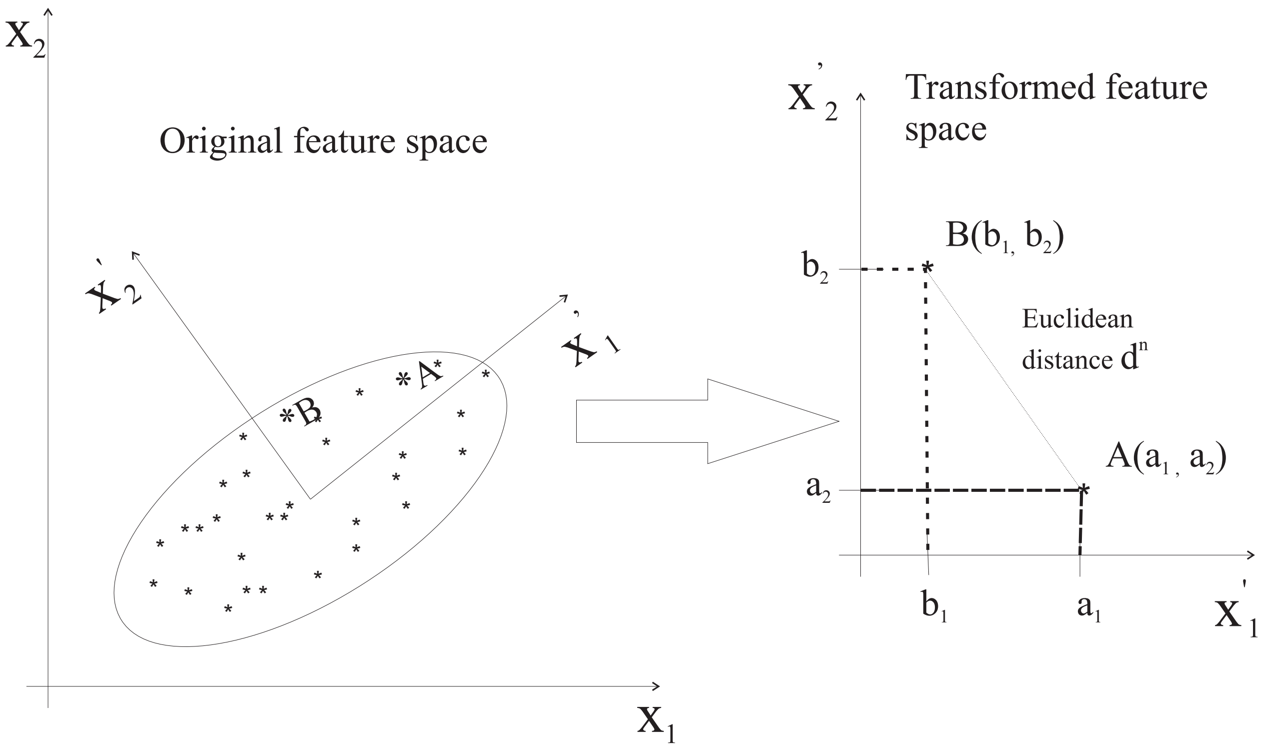

Figure 3: Mapping $f : x^n \mapsto x^{n+1}$ in the case of PCA-based feature vectors. Image A and B in original feature space (left) and image A and B in transformed feature space (right).

Ignoring one of the dimensions from the Euclidean space never increases the distance between the two points, as illustrated in Figure 3. At most, the distance remains the same. The distance d^n is always equal to or larger than any of the distances $|b_1 - a_1|$ or $|b_2 - a_2|$. This holds also for high-dimensional cases. That means, regardless of the order of features, the distance will always decrease with decreased dimensionality of the feature vectors, and the Requirement 4 is fulfilled.

4 Experiments and results

To test our concept of efficient 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 some network-related phenomena, such as network delays and limitations in network bandwidth.

A VSN node in the simulator is represented as an object with feature vector storage, inbound and outbound network packet buffers, image buffer and temporary processing storage. We do not impose any restrictions on node processing capabilities and we do not limit the amount of network traffic in any way. However, the simulator measures both the amount of traffic transmitted between the nodes and the number of nodes (hops) over which the traffic is transmitted. The size of packets can be defined by user and does not necessary correspond to the actual size of Matlab data structures, which are used to store the data in the simulator. This allows for accurate measurement of network load, which would otherwise be distorted due to redundant nature of some Matlab data structures.

The simulator can accept any structure of the simulated network, however, the routing algorithm has to be supplied as well. To get good insight into conditions in the network, we limited our testing only to the rectangular, 4-connected grid networks. In such case, a simple and deterministic routing algorithm can be used. Our simulator supports several types of network packets. Some packet types are transmitted only node to node, and processed before the data is passed along (for example, feature packets during the learning and query packets during the recognition), while others are routed through multiple nodes with the help of the routing algorithm (for example, query answer packets, which are sent by node which finds level 0 match to the primary node).

The simulator allows us to inject any type of network packet to any point in the network. It also allows injection of new image to 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 recognition, we inject an image into particular node, wait until the network activity subsides, and read the result of recognition from the same node.

4.1 Feature distribution methods

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

- Method, denoted "naive 1" corresponds to the scenario, where captured visual information is stored locally and each task of finding an object has to be broadcasted across the network, for each new image acquired. Such method of distribution requires little or no network traffic during the learning phase, however, it produces large transmissions of data in the recognition phase.

- Method, denoted "naive 2" corresponds to the scenario, where captured visual information from each sensor is distributed to all nodes for local storage. Detection of similar objects is then performed locally by each sensor as new images are acquired. Opposite from the first method, this feature distribution method produces large transmissions of data during the learning phase and requires very little or no traffic during the recognition phase.

- Method, denoted "hierarchical", which is essentially a feature distribution method proposed in this paper.

4.2 Database

We used standard COIL-20 database, which consists of images of 20 different objects; each one is rotated with 5 degree angle interval, corresponding to 72 images per object. That sums up to 1,440 images for the whole database [16]. The limitations of this database are *numerous* and *obvious*. However, we are not interested in object recognition performance of the three very simple object recognition methods we used (template matching, histogram matching and PCA) on a very limited database. Instead, the simplicity of both the chosen methods and the database allowed us to focus on the comparison between the three feature distribution methods, outlined above, and spend less time dealing with the recognition-related issues that are not the focus of our current work.

4.3 Our implementation of object recognition methods

In our implementation of template matching, the maximum number of levels M was set to 5, which resulted in minimum image dimensions of 4×4 pixels at 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).

In our implementation of histogram matching, first 10 bins (out of 256) were set to zero immediately after level 0 features were extracted from image. This was done to decrease the effect of the black background in the images from COIL-20 database, which dominated the histogram comparison if unmodified histograms were used.

During the testing of the PCA-based recognition, only odd-numbered object orientations were used. The even-numbered orientations were used to build a PCA subspace before the recognition phase. This subspace was used both for learning and recognition.

4.4 Network setup

For experiments, we used a network, consisting of 99 nodes, arranged to a 11×9 rectangular grid. Each of the nodes, except those on the network boundaries, was connected to its four immediate neighbors. Same experiment could be run with different network topology, however, rectangular grid allows simple routing algorithms to be used and provides best insight into network operation.

4.5 Experimental setup

The experiment was divided into two separate phases. In the first phase, only learning was performed. Twenty nodes, evenly distributed through the network, were injected with images of the twenty different objects from the database. Those images corresponded to the zero orientation in the COIL-20 database. Next, the simulation cycle was started, and, after the network traffic stopped, the statistics on network load (number of hops and the total network traffic) was examined.

In the second phase, only recognition (testing) was performed. A pseudo-random sequence was used to choose any image from the database and any node from the network. The image was injected into the chosen node, and the simulation cycle was started. After the network activity stopped, the result of the recognition was read from the same node, and the statistics on false positives (FP) and false negatives (FN) was updated. The process of injecting the random image to a random node was repeated 5,000 times, and the statistics on the network load (number of hops and the total network traffic) was examined again.

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

5 Results

The results are shown in Table 1. It can be seen that recognition rates (number of false positives and false negatives) are exactly the same for all feature distribution methods. This is not surprising, since the fulfillment of the Requirements 1–4 guarantees that the recognition performance will not decrease if the hierarchical feature vector distribution scheme is used. Therefore, all feature vector distribution methods are equivalent in that respect, which we expected.

Recognition	Distribution	Learning		Recognition		Total		Recognition rate	
method	method	Hops	Traffic	Hops	Traffic	Hops	Traffic	FP	FN
Template	Naive 1	0	0	1,398,789	20,172 M	1,398,789	20,172 M	14%	23%
	Naive 2	12,280	56 M	0	0	12,280	56 M	14%	23%
	Hierarchical	12,280	816 K	534,524	6,658 M	546,804	6,659 M	14%	23%
Histogram	Naive 1	0	0	1,472,740	2,536 M	1,472,740	2,536 M	26%	29%
	Naive 2	12,280	7 M	0	0	12,280	7 M	26%	29%
	Hierarchical	12,280	505 K	723,381	1,0634 M	735,661	1,064 M	26%	29%
PCA	Naive 1	0	0	1,404,649	5,055 M	1,404,649	5,055 M	15%	22%
	Naive 2	12,280	14 M	0	0	12,280	14 M	15%	22%
	Hierarchical	12,280	870 K	554,256	1,676 M	554,415	1,676 M	15%	22%

Table 1: Experimental results for each combination of recognition methods and feature distribution methods. In the Traffic columns, plain numbers represent the number of bytes transferred, while K denotes kilobytes, and M denotes megabytes. FP and FN denote the percentage of false positives and false negatives, respectively.

It can also be seen that, regardless of the recognition method used, the proposed hierarchical feature distribution scheme results in far lower total network traffic load than the "naive 1" method, while the "naive 2" method shows the lowest total numbers. The totals for "naive 2" method are somewhat non-representative, since that method makes the heaviest use of the network during the training, and we trained the network with 20 images, while testing it with 5,000 images.

Phase-by-phase (learning-recognition) comparison of the network traffic shows that our proposed hierarchical method significantly outperforms "naive 2" method during learning and "naive 1" method during recognition. 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 recognition), our methods utilizes the network resources in a more balanced way. If the learning *and* recognition would be performed on each object, our method would significantly outperform both naive feature distribution methods.

The number of hops for hierarchical feature vector distribution scheme for different recognition methods shows an interesting phenomenon. The recognition methods with lower number of false positives (PCA and template matching) generate lower number of packets (which is reflected in lower number of hops). This is easily explained by the nature of the hierarchical feature distribution scheme and verified by visual inspection of network traffic animation. Large number of false positives at higher levels of abstraction n causes larger number of forwarded queries during the recognition phase – essentially, larger number of *stray query packets*, which diverge from optimum query path, appears. These packets are forwarded for a hop or two, and disappear when they reach nodes with sufficiently detailed features, which show the nodes that the packets do not match the stored features and should be dropped.

6 Conclusion

The paper focuses on important conceptual problem of mapping object recognition methods to VSNs, specifically on the issue of knowledge propagation. We propose hierarchical feature distribution scheme, which guarantees *visibility* of any feature vector from any node of the network with only a fraction of the network load the full distribution of feature vectors would cause.

In this study, extremely simple object recognition methods were used. For these methods we can be reasonably sure that they satisfy the full set of requirements of our feature distribution method, which results in unchanged performance when they are used in our hierarchical framework. However, the real challenge lies in adaptation of state-of-the art recognition methods to such distributed framework. It is unlikely that the requirements of our scheme would be fulfilled *in general* for those methods. However, even if those requirements are satisfied *most of the time* and only for *reasonable input data* (e.g. natural images), we expect that use of such methods in our hierarchical scheme would result in only a minor decrease in recognition performance. This is the future focus of our research.

References

[1] C. Arth. *Visual Surveillance on DSP-based Embedded Platforms.* PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, 2008.

[2] C. Arth and H. Bischof. Real–time object recognition using local features on a dsp–based embedded system. *Journal of Real–time Image Processing*, pages 233–253, 2008.

[3] M. Bramberger, R. P. Pflugfelder, A. Maier, B. Rinner, B. Strobl, and H. Schwabach. A smart camera for traffic surveillance. In *Proceedings of the first Workshop on Intelligent Solutions in Embedded Systems (WISES)*, pages 153–164, 2003.

[4] R. Chang, S.-H. Ieng, R. Benosman, L. Lacheze, and T. Debaecker. Auto-organized visual perception using distributed camera network, In *OMNIVIS 2008, 8th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras, in conjunction with ECCV 2008*, 2008.

[5] Y. Charfi, N. Wakamiya, and M. Murata. Challenging issues in visual sensor networks. `http://www.nal.ics.es.osaka-u.ac.jp/~charfi/`.

[6] Y. Charfi, N. Wakamiya, and M. Murata. Trade-off between reliabilty and energy cost for content–rich data transmission in wireless sensor networks. In *Proceedings of the Broadnets*, pages 1–8, 2006.

[7] Z. Cheng, D. Devarajan, and R.J. Radke. Determining vision graph for distributed camera networks using feature digests. *Journal on Advances in Signal Processing*, 2007. doi:10.1155/2007/57034.

[8] D. Devarajan and R. J. Radke. Calibrating distributed camera networks using belief propagation. *EURASIP Journal on Applied Signal Processing*, 2007. doi:10.1155/2007/60696.

[9] 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 *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW06)*, 2006. doi:10.1109/CVPRW.2006.6.

[10] A. Gilbert and R. Bowden. Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity. In *Proceedings ECCV 2006*, pages 125–136, 2006.

[11] A. Gilbert and R. Bowden. Incremental, scalable tracking of objects inter camera. *Computer Vision and Image Understanding*, (111):43–58, 2008.

[12] M. Kristan, J. Perš, M. Perše, and S. Kovačič. Closed-world tracking of multiple interacting targets for indoor-sports applications. *Computer Vision and Image Understanding*, 2008. In press.

[13] V. Lecuire, C. Duran-Faundez, and N. Krommenacker. Energy–efficient transmission of wavelet-based images in wireless sensor networks. *EURASIP Journal on Image and Video Processing*, 2007. doi:10.1155/2007/47345.

[14] C. Leistner, P. M. Roth, H. Grabner, H. Bischof, A. Starzacher, and B. Rinner. Visual on-line learning in distributed camera networks. In *Second ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC 2008*, pages 1–10. IEEE, 2008.

[15] J. P. Lewis. Fast template matching. *Vision Interface*, pages 120–123, 1995.

[16] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20), cucs-005-96. Technical report, Department of Computer Science, Columbia University, 1996.

[17] 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 *Workshop on Distributed Smart Cameras, in conjunction with ACM SenSys'06*, pages 72–77, 2006.

[18] M. A. Patricio, J. Carbo, O. Perez, J. Garcia, and J. M. Molina. Multi–agent framework in visual sensor networks. *EURASIP Journal on Advances in Signal Processing*, 2007. doi:10.1155/2007/98639.

[19] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl. Autonomous multicamera tracking on embedded smart cameras. *EURASIP Journal on Embedded Systems*, 2007. doi:10.1155/2007/92827.

[20] E. Simmons, E. Ljung, and R. Kleihorst. Distributed vision with multiple uncalibrated smart cameras. In *Workshop on Distributed Smart Cameras, in conjunction with ACM SenSys'06*, pages 67–72, 2006.

[21] S.N. Sinha and M. Pollefeys. Synchronization and calibration of camera networks from silhouettes. *Proceedings of the 17th International Conference on Pattern Recognition, 2004 ICPR 2004*, 1:116–119, Aug 2004.

[22] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision.* Thomson Learning, third edition, 2008.

[23] X. Wang, S. Wang, D.-W. Bi, and J.-J. Ma. Distributed peer-to-peer target tracking in wireless sensor networks. *Sensors*, pages 1001–1027, 2007.

[24] H. Wu and A.A. Abouzeid. Error resilient image transport in wireless sensor networks. *Computer Networks*, 50(15):2873–2887, 2005.