Technical Report FE-LSV-01/12
Efficient Feature Distribution in Visual-Sensor Networks - An Example

Vildana Sulić Kenk, Janez Perš, Matej Kristan, Stanislav Kovačič
Faculty of Electrical Engineering, University of Ljubljana
Tržaška 25, SI-1000 Ljubljana
E-mail: vildana.sulic@fe.uni-lj.si

Abstract

In the so-called visual-sensor networks (VSNs), the processing tasks are distributed across many spatially distributed smart cameras, and, in theory communication between them is expensive. Recently, we proposed a framework of hierarchical feature distribution (HFD) for object matching in a network of visual sensors, which utilizes the network resources in a more balanced way in comparison to naive distribution. In this report we summarize some essential elements of HFD and describe the use of such framework on an illustrative example.

1 Hierarchical feature-distribution scheme

Object matching (re-identification) in distributed camera network essentially requires obtaining feature correspondence between any pair of possibly distant camera nodes. It is obvious that the complexity of such task grows non-linearly with the network size. To keep the problem manageable, efficient hierarchical scheme for feature distribution may be used. Theoretical foundations for such scheme are described in [1]. In this report we provide only brief overview of its main concepts, along with the illustrative example, to familiarize the reader with the challenges and constraints that arise in distributed camera networks.

In any object recognition scheme we deal with the following two concepts. In *learning*, the compact representation (in the form of a feature vector) of the object is extracted from one or more images and either stored or used to train the appropriate classifier. In *matching*, the same compact representation of an object is extracted from the newly acquired image or image sequence. This vector is used to obtain a correspondence with one of the learned objects (or object classes).

In distributed camera network, two naive strategies to distribute features are possible. The entire network can be flooded with object features whenever a new image is acquired and as a result of this flooding, recognition can be done in any node; however, this essentially duplicates the role of a central processor on each node and as such defeats the purpose of distributed processing. On the other hand, if we do not flood the network with each and every acquired image and a particular node wants to recognize (re-identify) a previously seen object, all the features from the network have to be requested for a comparison, resulting in huge amount of

network traffic. Hierarchical feature-distribution (HFD) scheme [1] provides a more balanced approach by distributing a series of progressively coarse features, which are used for efficient routing of re-identification queries.

2 Formal definition

Hierarchical feature-distribution scheme is based on a hierarchical reduction of feature vectors. 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. For this, HFD requires *feature mapping function* (i.e., abstraction) $f : \mathbf{x}^{(n)} \mapsto \mathbf{x}^{(n+1)}, 0 < n \leq N$, 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. After applying f, feature vector requires less storage and transmission capacity. In this way, the amount of data transmitted across, or stored in the network, can be significantly reduced.

In HFD scheme, feature vectors can be only matched to other feature vectors of the same level n, however, if vector of a lower level $m; m < n$ is available, it can be always transformed to level n by applying mapping f several $(n-m)$ times. To compare two feature vectors, a distance measure $d^{(n)}(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)})$, 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 is needed. For classification, a simple threshold rule is used – if the distance d is smaller than the predefined threshold T, objects are declared to be of the same class. Finally, distance d has to decrease monotonically by successive application of f, which guarantees that any possible match with the less-descriptive feature vectors can be back-traced to the primary node, which has full feature vector at its disposal.

In other words, the use of such scheme 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 have proposed four requirements that have to be fulfilled by any object-matching method to be considered for use in the HFD scheme. The feature vectors x, which are passed across the network, should fulfill the four major requirements, which are mathematically defined as follows:

Requirement 1 (Abstraction): There exists a mapping $f : x^n \mapsto x^{n+1}$, which translates level n feature vector x^n into higher, more abstract level $(n+1)$ feature vector x^{n+1}, without access to the original visual data.

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. 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.

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

Requirement 3 (Existence of a 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 critical 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, $d^{n+1}(x_1^{n+1}, x_2^{n+1}) \leq d^n(x_1^n, x_2^n)$.

3 Illustrative example: Learning and object matching in HFD scheme

Learning: Let us assume that node 7 (Figure 1 a) has acquired an image, located an object in it (using some kind of motion detection scheme, for example) and extracted object feature vector $\mathbf{x}_?$. The node generates a unique identification number – ID (in this case 'A'), which is then attached to the feature vector. Feature vector is stored in the node's local storage and is marked as being level 0 ($\mathbf{x}_A^0$). Now the knowledge about object 'A' is local. Other nodes have no idea that node 7 has seen object A. Therefore, node 7 has to advertise/broadcast the fact that it has seen object 'A'. This is done in the following way. Using the mapping, the next level (level 1) feature vector $\mathbf{x}_A^1$ is prepared and propagated to the direct neighbors (to the nodes 3, 6, 8 and 11) with the ID 'A' still attached. (Note that level 1 feature vector $\mathbf{x}_A^1$ requires less storage space and is more abstract representation of an object than level 0 feature vector $\mathbf{x}_A^0$.) Nodes 3, 6, 8 and 11 store the received level 1 feature vector $\mathbf{x}_A^1$ and remember the direction (the neighbor) where it came from (in our case node 7). Next, these nodes prepare next level (level 2) feature vector $\mathbf{x}_A^2$ and broadcast this feature vector to theirs direct neighbors (to the nodes 2, 4, 5, 10, 12, 15). These nodes again remember the origin of this feature vector $\mathbf{x}_A^2$ (e.g., origin of the $\mathbf{x}_A^2$ in the node 15 is node 11; this means that node 11 has more descriptive feature vector on level 1, $\mathbf{x}_A^1$). This procedure of applying the mapping, storing the origin and broadcasting the less descriptive feature vector to the direct neighbors is repeated on each node, until every node in the network has at least some information about the observed object (situation is shown in Figure 1 b).

A note on maximum level of abstraction N: Let us assume that N is set to 5. That means that in our example, $\mathbf{x}_A^6$ is never generated and $\mathbf{x}_A^5$ is forwarded unchanged from that point on. Communications between the nodes and the unique IDs ensure that the nodes refuse to accept any duplicated feature vectors or more abstract versions of the feature vectors they already have in local storage.

Now, let us say that node 10 has acquired image B, node 13 image C, node 8 image D and node 1 image E. The following Table 1 shows which feature vectors can be found in particular node's local storage.

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*.

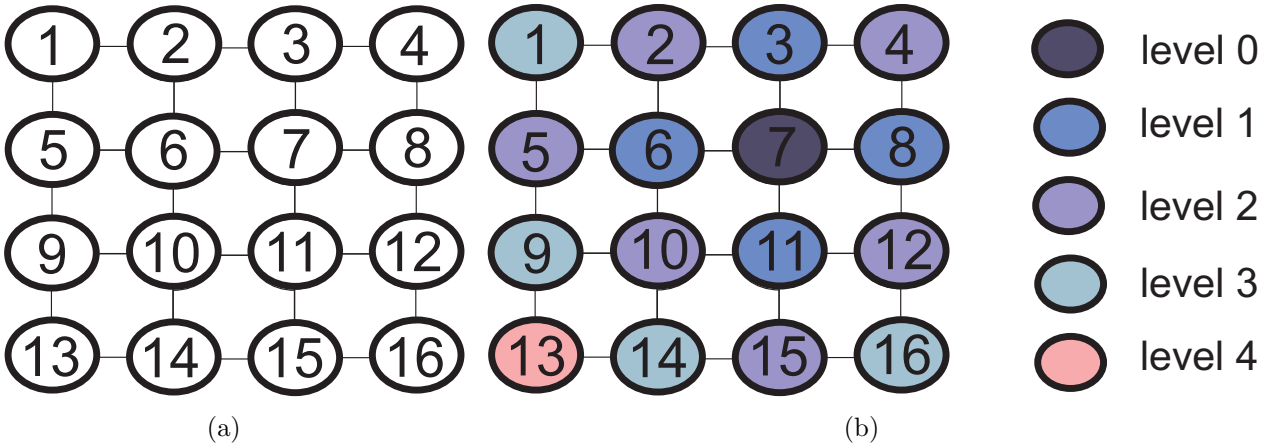

Figure 1: (a) an example of the network and its nodes; (b) situation, when feature vector $\mathbf{x}_A$ is broadcasted across the whole network (learning phase); Each color represents different level of abstraction of feature vector $\mathbf{x}_A$.

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**

Let us assume that node 8 has acquired new image, located an object in it and extracted object feature vector $\mathbf{x}_?$. Node starts with the local search. In its own storage it can find following feature vectors: $\mathbf{x}_A^1, \mathbf{x}_B^3, \mathbf{x}_C^5, \mathbf{x}_D^0, \mathbf{x}_E^4$. Note: node can generate any of more abstract

Table 1: Feature vectors stored in the node's local storage

Node	Local storage
1	$\mathbf{x}_A^3, \mathbf{x}_B^3, \mathbf{x}_C^3, \mathbf{x}_D^4, \mathbf{x}_E^0$
2	$\mathbf{x}_A^2, \mathbf{x}_B^2, \mathbf{x}_C^4, \mathbf{x}_D^3, \mathbf{x}_E^1$
3	$\mathbf{x}_A^1, \mathbf{x}_B^3, \mathbf{x}_C^5, \mathbf{x}_D^2, \mathbf{x}_E^2$
4	$\mathbf{x}_A^2, \mathbf{x}_B^4, \mathbf{x}_C^5, \mathbf{x}_D^1, \mathbf{x}_E^3$
5	$\mathbf{x}_A^2, \mathbf{x}_B^2, \mathbf{x}_C^2, \mathbf{x}_D^3, \mathbf{x}_E^1$
6	$\mathbf{x}_A^1, \mathbf{x}_B^1, \mathbf{x}_C^3, \mathbf{x}_D^2, \mathbf{x}_E^2$
7	$\mathbf{x}_A^0, \mathbf{x}_B^2, \mathbf{x}_C^4, \mathbf{x}_D^1, \mathbf{x}_E^3$
8	$\mathbf{x}_A^1, \mathbf{x}_B^3, \mathbf{x}_C^5, \mathbf{x}_D^0, \mathbf{x}_E^4$
9	$\mathbf{x}_A^3, \mathbf{x}_B^1, \mathbf{x}_C^1, \mathbf{x}_D^4, \mathbf{x}_E^2$
10	$\mathbf{x}_A^2, \mathbf{x}_B^0, \mathbf{x}_C^2, \mathbf{x}_D^3, \mathbf{x}_E^3$
11	$\mathbf{x}_A^1, \mathbf{x}_B^1, \mathbf{x}_C^3, \mathbf{x}_D^2, \mathbf{x}_E^4$
12	$\mathbf{x}_A^2, \mathbf{x}_B^2, \mathbf{x}_C^4, \mathbf{x}_D^1, \mathbf{x}_E^5$
13	$\mathbf{x}_A^4, \mathbf{x}_B^2, \mathbf{x}_C^0, \mathbf{x}_D^5, \mathbf{x}_E^3$
14	$\mathbf{x}_A^3, \mathbf{x}_B^1, \mathbf{x}_C^1, \mathbf{x}_D^4, \mathbf{x}_E^4$
15	$\mathbf{x}_A^2, \mathbf{x}_B^2, \mathbf{x}_C^2, \mathbf{x}_D^3, \mathbf{x}_E^5$
16	$\mathbf{x}_A^3, \mathbf{x}_B^3, \mathbf{x}_C^3, \mathbf{x}_D^2, \mathbf{x}_E^5$

versions of the feature vector $\mathbf{x}_?$ by applying the mapping f multiple times, and therefore, other levels of the newly obtained feature vector are calculated: $\mathbf{x}_?^1, \mathbf{x}_?^3, \mathbf{x}_?^5, \mathbf{x}_?^0, \mathbf{x}_?^4$. Each of these feature vectors has to be compared to the corresponding feature vectors previously stored in the local storage. Comparison is done using the metric of similarity, which is matching-method dependent: $d^1(\mathbf{x}_?^1, \mathbf{x}_A^1)$, $d^3(\mathbf{x}_?^3, \mathbf{x}_B^3)$, $d^5(\mathbf{x}_?^5, \mathbf{x}_C^5)$, $d^0(\mathbf{x}_?^0, \mathbf{x}_D^0)$, $d^4(\mathbf{x}_?^4, \mathbf{x}_E^4)$.

1. If each of the comparison resulted above the predefined threshold T, we can declare that newly acquired image has not been seen before in the network.

2. If $d^0(\mathbf{x}_?^0, \mathbf{x}_D^0) \leq T$, we can declare that this image has been seen locally (by node 8) on some previous occasion.

3. If, for example $d^4(\mathbf{x}_?^4, \mathbf{x}_E^4) \leq T$ and $d^1(\mathbf{x}_?^1, \mathbf{x}_A^1) \leq T$, some other node in the network might have seen the object. Since the match is on more abstract level, the quality of this decision is questionable, and we have to confirm the match by comparing two feature vectors on level 0. But the node does not have $\mathbf{x}_E^0$ and $\mathbf{x}_A^0$ and it has to proceed with the network search, which is essentially forwarding the unknown feature vector $\mathbf{x}_?^0$ in the directions where matching level 0 feature vectors could be found. To do this, it looks up the origin of the feature vectors that resulted in a match (origin is stored in the local storage), in our case $\mathbf{x}_E^4$ and $\mathbf{x}_A^1$. Node forwards $\mathbf{x}_?^0$ to the nodes 4 (origin of the $\mathbf{x}_E^4$) and 7 (origin of the $\mathbf{x}_A^1$). Nodes 4 and 7 start with the local search as described above.

   (a) $d^0(\mathbf{x}_?^0, \mathbf{x}_A^0) \leq T$, node can declare that it found a match. Since this comparison was on level 0 (original feature vectors), the identity of the object is confirmed - judging

from the feature comparison, this object is the same as A and has been seen before in the node 7 ($\mathbf{x}_?^0 = \mathbf{x}_A^0$).

(b) If $d^3(\mathbf{x}_?^3, \mathbf{x}_E^3) \leq T$, node 4 proceeds with the network search, forwarding $\mathbf{x}_?^0$ to the node 3 (origin of the $\mathbf{x}_E^3$).

(c) In case that both $d^0(\mathbf{x}_?^0, \mathbf{x}_A^0)$ and $d^3(\mathbf{x}_?^3, \mathbf{x}_E^3)$ would be above the threshold, we can terminate the search. Object is unknown and the rationale is the same than in point 1 above.

4 Conclusion

HFD scheme utilizes network in a more balanced way than trivial network flooding. The scheme is based on hierarchical distribution of the information, where each individual node retains only a small amount of information about the objects seen by the network. However, this amount is sufficient to efficiently route queries through the network without any degradation in the recognition performance. As it can be seen above, the efficiency of the described approach stems from the fact that after applying feature mapping f, feature vector requires less storage and transmission capacity. In this way, the amount of data transmitted across, or stored in the network, can be significantly reduced. Furthermore, the feature vectors are only forwarded in a direction where there is a possibility that the object has been seen.

References

[1] V. Sulić, J. Perš, M. Kristan, and S. Kovačič. Efficient feature distribution for object matching in visual-sensor networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(7):903–916, 2011.