Distributed classification of multiple observation sets by consensus

Effrosyni Kokiopoulou and Pascal Frossard

***Abstract*—We consider the problem of distributed classification of multiple observations of the same object that are collected in an ad-hoc network of vision sensors. Assuming that each sensor captures a different observation of the same object, the problem is to classify this object by distributed processing in the network. We present a graph-based problem formulation whose objective function captures the smoothness of candidate labels on the data manifold formed by the observations of the object. We design a distributed average consensus algorithm for estimating the unknown object class by computing the value of the above smoothness objective function for different class hypotheses. It initially estimates the objective function locally based on the observation of each sensor. As the distributed consensus algorithm progresses, all observations are progressively taken into account in the estimation of the objective function. We illustrate the performance of the distributed classification algorithm for multi-view face recognition in an ad-hoc network of vision sensors. When the training set is sufficiently large, the simulation results show that the consensus classification decision is equivalent to the decision of a centralized system with access to all observations.**

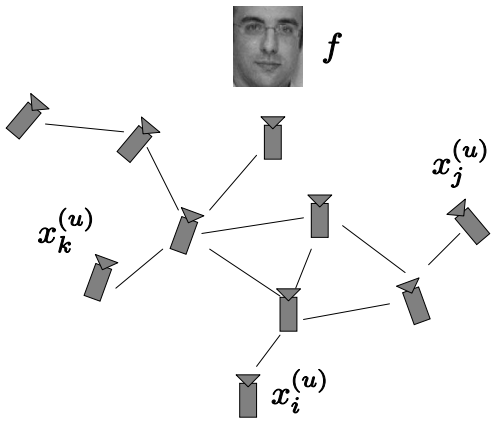


Fig. 1. Ad-hoc network of vision sensors.

I. Introduction

Over the past few years novel multimedia architectures such as vision sensor networks have rapidly emerged. Typically, these networks have an ad-hoc organization i.e., there is no central coordinator node and the topology can be arbitrary and dynamic (e.g., due to sensor motion). Moreover, the visual sensor nodes in such networks have limitations in their computation and communication capabilities. Rinner et al. [1], [2] and Akyildiz et al. [3] provide an overview of platforms that have been recently developed for visual sensor networks, which lend themselves as off-the-self computing infrastructures for conducting various scene analysis tasks in smart environments. The emergence of such distributed multimedia architectures poses new challenges to the analysis of multimedia information, which has to be done now distributively. We quote from [1]: "*Existing computer vision algorithms often are not designed with collaboration of distributed nodes in mind. For pervasive smart cameras, however, this aspect is highly important. Hence, ways have to be found how algorithms can be adopted for such environments.*" Therefore, the relevant algorithms have to be (re-)designed such that they accommodate collaborative processing, while at the same time respecting the computation and communication constraints of the underlying network (see e.g., [4]).

In this paper, we consider the problem of classifying an object, whose multiple observations are collected in a distributed fashion in a vision sensor network with ad-hoc topology (see, e.g., [5], [6], [7]). Fig. 1 illustrates the scenario of interest, where each vision sensor captures an observation of the *same* object in the context of (distributed) scene analysis, for example. The problem consists in the distributed classification of the observed object *at all sensors* such that a consensus decision is reached by aggregating partial information provided by each local observation. It is important to note that this problem is different from the well-studied problem of distributed classification in the presence of a fusion center (see, e.g., [8], [9], [10]), where the information from all sensors is gathered in order to reach the final classification decision. On the contrary, the ad-hoc sensor networks considered in this paper are purely distributed, and there is no possibility of transmitting directly information from the sensors to a central coordinator node.

We first present a graph-based problem formulation that defines a smoothness criterion of candidate labels on the data manifold. This criterion reflects the so-called smoothness assumption that is commonly used in semi-supervised learning [11]; namely two closeby data samples on the manifold are likely to share the same class label. It permits to define the objective function of the distributed classification problem, whose solution should satisfy the smoothness assumption.

E. Kokiopoulou is with the Seminar for Applied Mathematics, Department of Mathematics, ETH Zurich, CH-8092 Zurich. email: efrosyni.kokiopoulou@sam.math.ethz.ch. Part of this work has been conducted when E. Kokiopoulou was with LTS4, EPFL.

P. Frossard is with the Signal Processing Laboratory (LTS4), Institute of Electrical Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne e-mail: pascal.frossard@epfl.ch.

Our distributed classification algorithm further capitalizes on the fact that the multiple observations belong to the same class. In particular, each sensor captures an observation of the same object (see also Fig. 1) and computes its nearest neighbors among the labelled examples. Under a certain class hypothesis, those neighbors contribute to the local computation of a portion of the objective function value. Those portions are summed distributively by means of average consensus [12], [13], so that all observations are progressively taken into account and the total value of the objective function is computed at all sensors. This process is repeated for all class hypotheses. The sensors eventually reach a consensus classification decision, by picking the class resulting in the smoothest label assignment.

We illustrate the performance of the proposed distributed algorithm in multi-view face recognition in a simulated ad-hoc network of vision sensors. When the training set is sufficiently large, the simulation results show that the consensus classification decision is equivalent to the decision of a centralized system that would have access to all observations.

The rest of the paper is organized as follows. We formally define the problem of distributed classification in sensor networks with ad-hoc topology in Section II and then in Section III we present our graph-based problem formulation. In Section IV we introduce our distributed classification algorithm, which is solely based on consensus-based distributed averaging. In the sequel, in Section V, we show the feasibility of our algorithm in the context of distributed multi-view face recognition. Finally, we discuss the related work in Section VI.

II. Distributed classification of multiple observations

Let us formally define the problem of distributed classification of multiple observations in an ad-hoc sensor network. We consider a network of m sensors and we model the network topology as an undirected graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ with nodes $\mathcal{V}_s = \{1, \ldots, m\}$ corresponding to sensors. An edge $(i,j) \in \mathcal{E}_s$ is drawn if and only if the sensor i can communicate with the sensor j. Then, we associate a weight $W(i,j)$ with each edge $(i,j) \in \mathcal{E}_s$. We call *weight matrix* the matrix W that gathers the edge weights $W(i,j)$. Note that W is a sparse matrix whose sparsity pattern is driven by the network topology. We denote the set of neighbors for node i as $\mathcal{N}_i = \{j |\ (i,j) \in \mathcal{E}_s\}$.

We assume that each sensor j captures a single (unlabelled) observation $x_j^{(u)}$ of an object f. Each observation is different from its peers and has the following form,

$$x_j^{(u)} \triangleq U(\eta_j) f, \quad j = 1, \ldots, m. \tag{1}$$

In the above, $U(\eta_j)$ denotes a transformation applied on the object f with parameters η_j. For instance, the transformation could be a (in-plane or out-of-plane) rotation and η_j could denote the rotation angle. Hence, there are m observations of the object f that are recorded over the sensor network and there is one-to-one correspondence among sensors and observations.

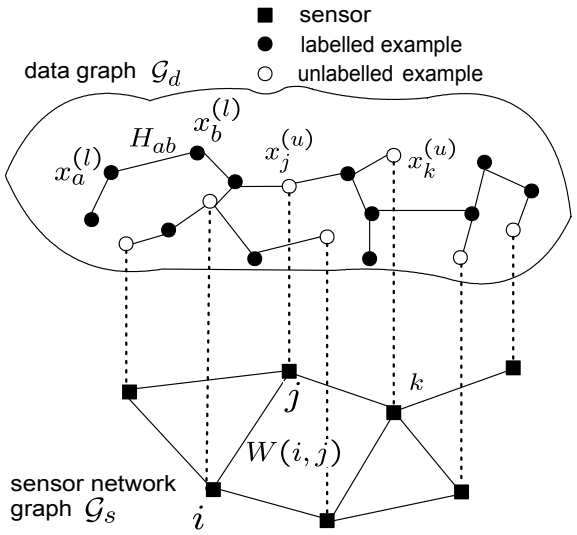


Fig. 2. Conceptual distinction between the two graphs of the problem. $\mathcal{G}_s$ (resp. $\mathcal{G}_d$) denotes the graph of the sensor network topology (resp. the data graph). In $\mathcal{G}_d$, the filled (resp. empty) circles correspond to labelled (resp. unlabelled) examples.

Assume further that the data set is organized in two parts $X = \{X^{(l)}, X^{(u)}\}$, where $X^{(l)} = \{x_1, x_2, \ldots, x_l\} = \{x_1^{(l)}, x_2^{(l)}, \ldots, x_l^{(l)}\} \subset \mathbb{R}^d$ and $X^{(u)} = \{x_{l+1}, \ldots, x_n\} = \{x_1^{(u)}, \ldots, x_m^{(u)}\} \subset \mathbb{R}^d$, where $n = l + m$. Let also $\mathcal{L} = \{1, \ldots, c\}$ denote the label set. The l examples in $X^{(l)}$ are labelled $\mathcal{Y}^{(l)} := \{y_1, y_2, \ldots, y_l\}$, $y_i \in \mathcal{L}$ and common to all sensors, and the m examples in $X^{(u)}$ are unlabelled and distributed. Each of these examples corresponds to an observation made at a sensor, which is not available to the other sensors. The problem of distributed classification can be formally defined as follows.

Problem 1. *Assume that each sensor j has a copy of the labelled set $\{X^{(l)}, \mathcal{Y}^{(l)}\}$ in addition to its single observation $x_j^{(u)}$ defined in (1). Assume also that each sensor knows its neighbors and the weights of its links to them. The problem is to reach a consensus classification decision where each sensor predicts the correct class c^* of the object of interest f, by aggregating via local communication information from all available observations over the network.*

III. Graph-based problem formulation

We present a graph-based formulation of Problem 1, which is inspired by Label Propagation [14]. The latter is a very popular method for semi-supervised classification [11], which refers to the problem of assigning (possibly different) class labels to a set of given test data samples. It can be seen as a generalization of the problem of assigning a set of multiple test observations to a single class, which is the focus of the present work. Label Propagation is a well known method for semi-supervised classification that takes into account the manifold structure of the data by means of a graph.

We make use of a *smoothness assumption*, which states that if data samples x_1 and x_2 are similar, then their corresponding labels y_1 and y_2 should be close. We represent the data labels

with a 1-of-c encoding, which permits to form a binary label matrix of size $n \times c$, whose ith row encodes the class label of the ith example. The class label is basically encoded in the position of the nonzero element. Denote by $\mathcal{M}$ the set of matrices with nonnegative entries of size $n \times c$. Notice that any matrix $M \in \mathcal{M}$ provides a labelling of the data set by applying the following rule: $y_i = \max_{j=1,\ldots,c} M_{ij}$. We denote the initial label matrix as $Y \in \mathcal{M}$ where $Y_{ij} = 1$ if x_i belongs to class j and 0 otherwise.

We further form the k nearest neighbor (k-NN) graph denoted as $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$, where the vertices $\mathcal{V}_d$ correspond to the data samples X. Typically, an edge $e_{ij} \in \mathcal{E}_d$ is drawn if and only if x_j is among the k nearest neighbors of x_i. Hence, the k-NN graph captures the affinity of the data samples in the ambient space. It is common practice to assign weights on the edge set of $\mathcal{G}_d$, gathered in a weight matrix $H \in \mathbb{R}^{n\times n}$. The (normalized) similarity matrix $S \in \mathbb{R}^{n\times n}$ is further defined as

$$S = D^{-1/2} H D^{-1/2}, \tag{2}$$

where D is a diagonal matrix with entries $D_{ii} = \sum_{j=1}^{n} H_{ij}$. It is important to distinguish between the two graph models involved in our problem: the sensor graph and the data graph. Figure 2 illustrates the conceptual distinction between the two.

In the sequel, we first review briefly the basics of Label Propagation. Then we present our problem formulation first in centralized settings, which serve as performance benchmark, and then in distributed settings.

A. Label Propagation.

The algorithm computes a real valued $M^* \in \mathcal{M}$ based on which the final classification is performed using the rule $y_i = \max_{j=1,\ldots,c} M^*_{ij}$. This is done via a regularization framework with a cost function defined as

$$\begin{aligned}\Phi(M) \quad = \quad & \frac{1}{2}\Big(\sum_{i,j=1}^{n} H_{ij} \| \frac{1}{\sqrt{D_{ii}}} M_i - \frac{1}{\sqrt{D_{jj}}} M_j \|^2 \\ & + \mu \sum_{i=1}^{n} \|M_i - Y_i\|^2 \Big),\end{aligned} \tag{3}$$

where M_i denotes the ith row of M. The computation of M^* is done by solving the quadratic optimization problem $M^* = \arg\min_{M\in\mathcal{M}} \Phi(M)$. Intuitively, we are seeking for an M^* that is smooth along the edges of similar pairs (x_i, x_j) and at the same time close to Y when evaluated on the labelled data $X^{(l)}$. The first term in the definition of $\Phi(M)$ is the *smoothness* term and the second is the *fitness* term.

It can be shown [14] that the solution to the minimization of $\Phi(M)$ is given by

$$M^* = \beta(I - \alpha S)^{-1} Y, \tag{4}$$

where $\alpha = \frac{1}{1+\mu}$ and $\beta = \frac{\mu}{1+\mu}$.

Since the algorithm has been designed for semi-supervised learning, where the unlabeled data samples may have different class labels, the estimated class of Label Propagation in Problem 1 is finally obtained by majority voting on M^*.

B. Problem formulation in centralized settings

We now exploit the special structure of the problem, namely that the multiple observations belong to the same class. If we define a binary class label vector $\lambda = [\lambda_1, \ldots, \lambda_c] \in \mathbb{R}^c$, the optimal classification of Problem 1 should have only one non-zero entry, with the form $\lambda = [0, \ldots, \underbrace{1}_{c^*}, \ldots, 0]$. Intuitively, we seek for one of the c vectors λ with only one non-zero entry, which best reflects the manifold smoothness assumption. This optimal vector results in similar class label assignments for pairs that are similar.

The label smoothness criterion is alternatively captured by the following objective function

$$\mathcal{Q}_c(M) = \sum_{i,j=1}^{n} S_{ij} \|M_i - M_j\|^2, \tag{5}$$

where M_i (resp. M_j) denotes the ith (resp. jth) row of M. The objective function above becomes equivalent to the smoothness term of eq. (3) when S is row-stochastic i.e., the sum of each row is equal to one.

Since all multiple observations belong to the same class, M can be defined as

$$M = \sum_{p=1}^{c} \lambda_p Z_p, \tag{6}$$

where $\lambda_p \in \{0, 1\}$, $\sum_{p=1}^{c} \lambda_p = 1$ and Z_p is defined as

$$Z_p = \begin{bmatrix} Y^{(l)} \in \mathbb{R}^{l\times c} \\ \hline \mathbf{1} e_p^\top \in \mathbb{R}^{m\times c} \end{bmatrix} \in \mathbb{R}^{n\times c}. \tag{7}$$

In the above, $Y^{(l)}$ denotes the submatrix of Y associated with the labeled data $X^{(l)}$, and e_p is the canonical basis vector whose pth element is one and the rest is zero.

With the above definition of M, it can be shown [15] that the objective function (5) can be written in the following form,

$$\mathcal{Q}_c(\lambda) = C + \sum_{i\le l, j>l} S_{ij} \|Y_i - \lambda\|^2 + \sum_{i>l, j\le l} S_{ij} \|Y_j - \lambda\|^2,$$

where $C = \sum_{i\le l, j\le l} S_{ij} \|Y_i - Y_j\|^2$ is a constant term that does not depend on λ.

C. Problem formulation in distributed settings

Observe that the evaluation of the cost function $\mathcal{Q}_c(\lambda)$ defined above is not feasible in distributed settings. In this case, the nearest neighbors of each example can be chosen only among the labelled ones, as each sensor does not have access to the unlabelled examples apart from its own observation. For this reason, we adopt a slightly modified cost function in distributed settings, which is discussed below.

For each candidate vector λ, each sensor j locally computes a smoothness criterion as a weighted summation over the labelled examples that reads

$$r(j) = \sum_{i=1}^{l} S_{ji} \|Y_i - \lambda\|^2 \tag{8}$$

where Y_i denotes the ith row of the label matrix Y. The weight S_{ji} denotes the similarity of the unlabeled observation

x_j (collected at sensor j) with the labeled data sample x_i. The global smoothness function Q_d then aggregates the local criteria as

$$Q_d(\lambda) \quad = \quad \sum_{j=l+1}^{n} r(j) \tag{9}$$

where the index j runs over the unlabelled examples (observations). Notice that when an unlabelled example x_j ($j > l$) is similar to a labelled example x_i (i.e., the weight S_{ji} is large), then minimizing the above objective function will result in labels that are smooth across similar examples. Hence, we need to solve the following optimization problem.

Optimization problem: **OPT**
$\min_{[\lambda_1,\ldots,\lambda_c]} Q_d([\lambda_1,\ldots,\lambda_c])$
subject to
$\lambda_p \in \{0,1\}$, $p = 1,\ldots,c$,
$\sum_{p=1}^{c} \lambda_p = 1$.

IV. The distributed classification algorithm

In what follows, we discuss first how one can compute distributively the sum of local functions with consensus algorithms. Then we introduce our proposed distributed algorithm for solving the classification problem OPT.

A. Distributed consensus

Distributed consensus [12], [13] has recently become an important computational tool for various aggregation tasks in ad-hoc sensor networks. We consider distributed linear iterations of the following form

$$z_{t+1}(i) = W(i,i)z_t(i) + \sum_{j\in\mathcal{N}_i} W(i,j)z_t(j), \tag{10}$$

for $i = 1,\ldots,m$, where $z_t(j)$ represents the value computed by sensor j at iteration t. The above iteration can be compactly written in the following form

$$z_{t+1} = Wz_t. \tag{11}$$

Consensus can be employed for the problem of distributed averaging, as we explain below. Assume that initially each sensor i reports a scalar value $z_0(i) \in \mathbb{R}$. We denote by $z_0 = [z_0(1),\ldots,z_0(m)]^\top \in \mathbb{R}^n$ the vector of initial values on the network. Denote by

$$\bar{z}_0 = \frac{1}{m}\sum_{i=1}^{m} z_0(i) \tag{12}$$

the average of the initial values of the sensors. The problem of distributed averaging therefore becomes typically to compute $\bar{z}_0$ *at each sensor* by distributed linear iterations of the form of (11). Iteration (11) converges to the average for every z_0 if and only if

$$\lim_{t\to\infty} W^t = \frac{\mathbf{1}\mathbf{1}^\top}{m}, \tag{13}$$

where $\mathbf{1}$ is the vector of ones [13]. Indeed, notice that in this case

$$z^* = \lim_{t\to\infty} z_t = \lim_{t\to\infty} W^t z_0 = \frac{\mathbf{1}\mathbf{1}^\top}{m} z_0 = \bar{z}_0\mathbf{1}.$$

Algorithm 1 The distributed MASC algorithm

1: **Input to each sensor**:
   l: number of labelled data.
   $X^{(l)} \in \mathbb{R}^{d\times l}, Y^{(l)}$: labelled examples.
   $x^{(u)} \in \mathbb{R}^{d\times 1}$: unlabelled example (observation).
2: **Output at each sensor**:
   $\hat{p}$: estimated unknown class.
3: **Initialization at each sensor**:
4: Form the k-NN graph $\tilde{\mathcal{G}}_d$ of the data set $\{X^{(l)}, x^{(u)}\}$.
5: Compute the weight matrix $\tilde{H} \in \mathbb{R}^{(l+1)\times(l+1)}$ of $\tilde{\mathcal{G}}_d$.
6: Compute the diagonal matrix $\tilde{D}$, where $\tilde{D}_{i,i} = \sum_{j=1}^{l+1} \tilde{H}_{ij}$.
7: Compute $\tilde{S} = \tilde{D}^{-1/2}\tilde{H}\tilde{D}^{-1/2}$.
8: **for** $p = 1 : c$ **do**
9: Each sensor sets $\lambda = [0,\ldots,\underbrace{1}_{p},\ldots,0]$.
10: Each sensor j computes $r(j) = \sum_{i=1}^{l} \tilde{S}_{l+1,i}\|Y_i - \lambda\|^2$.
11: $q(p) = \sum_{j=1}^{m} r(j)$:=`average_consensus`(r).
12: **end for**
13: $\hat{p} = \arg\min_p q(p)$

B. Distributed classification

We are ready now to describe the distributed algorithm. First, each sensor j computes the nearest neighbors of its observation $x_j^{(u)}$ among the labelled examples and further computes the associated similarity weights. Next, it computes the value of the objective function $Q_d(\lambda)$ (see eq. (9)) for each candidate class p. The aforementioned computation involves first a local computation step and then a distributed computation step. In particular, for a certain class p, the neighbors of $x_j^{(u)}$ contribute to the calculation of a portion $r(j)$ of the objective function value, which involves only local computation (see eq. (8)). Next, those portions are averaged distributively, by means of average consensus, so that all observations are taken into account and the total value of the objective function is computed *at all sensors*, according to eq. (9). The evaluation of $Q_d(\lambda)$ is repeated for all candidate classes and eventually the sensors reach a consensus classification decision, by picking the class that results in the minimum value of the objective function.

We call the proposed algorithm distMASC i.e., distributed MAnifold Smoothing under Constraints. For notational ease, we drop the subscript j from $x_j^{(u)}$ when it is clear from the context that we refer to sensor j. The main steps are shown in Algorithm 1, where we have used a slightly different notation: we have attached a tilde to those quantities that are different from Section III-C due to the partial information of each sensor. For example, the local similarity matrix $\tilde{S} \in \mathbb{R}^{(l+1)\times(l+1)}$, which gathers the similarity weights of the local data set $\{X^{(l)}, x^{(u)}\}$ at each sensor, is not to be confused with the global similarity matrix $S \in \mathbb{R}^{n\times n}$ associated with the whole dataset $\{X^{(l)}, X^{(u)}\}$. We discuss below the proposed distributed algorithm in details.

First, each sensor computes the k-NN graph of its own data set $\{X^{(l)}, x^{(u)}\}$ and forms the corresponding $\tilde{S}$ matrix of size $(l+1)\times(l+1)$ (Lines 4-7). Next, each class hypothesis is tested (loop 8-12). For each class hypothesis p, each sensor j first

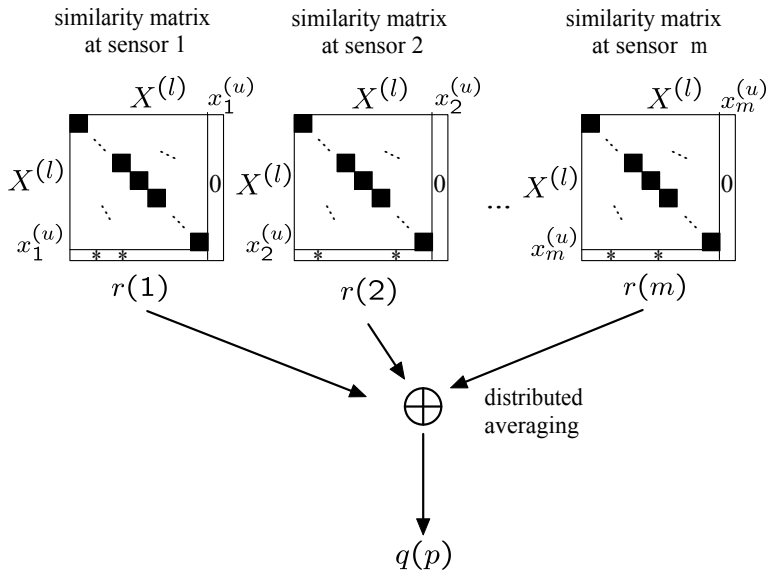


Fig. 3. Flow of computation, which is repeated for each hypothesis $p,\ p = 1, \ldots, c$. The stars in the last row of each similarity matrix correspond to the nearest neighbors of the observation $x^{(u)}$ among the labelled examples. The computation of $r(j)$ in the first row is local, i.e., no communication among the sensors is required.

computes a scalar number $r(j)$ that involves local computation only; namely a weighted sum of the nonzero entries of the last row of $\tilde{S}$ (i.e., $(l+1)$th row). This corresponds to a portion of the value of the objective function, which captures the smoothness of the label assignment under the current class hypothesis. In order to compute the value of the objective function $q(p)$, the partial sums $r(j)$ need to be summed together and this involves distributed computation. This step is performed by *distributed average consensus* (Line 11), where the summation of all r's is computed *at each sensor*. Note that this will result in a scaled version of $q(p)$, due to presence of $1/m$ in the average. However, this has no influence on the classification decision, which is taken in Line 13 by all sensors, after all hypotheses have been tested. At the end of the algorithm, all sensors reach a consensus decision.

Figure 3 shows schematically the flow of the distributed computation in Line 11 of Algorithm 1 for a single hypothesis p. We show the general structure of the similarity matrix $\tilde{S}$ formed at each sensor $j,\ j = 1, \ldots, m$ (assuming that the labelled data samples are ordered according to their class labels). Observe that the upper left block of S corresponding to the labelled set is common to all similarity matrices of the sensors, as they all have a copy of $X^{(l)}$. The only difference is in their last row, whose non-zero entries correspond to the nearest neighbors of their own observation $x^{(u)}$ among the labelled examples (indicated by asterisks in Figure 3). Notice that those entries contribute to the computation of the partial sums $r(j)$ in Line 10, which involves only local computation. Then, the sum of all values $r(j),\ j = 1, \ldots, m$ is computed distributively by average consensus, which yields the value of the objective function $q(p)$ for the current class hypothesis p. All observations contribute to the final classification decision, thanks to the employment of average consensus.

a) Computational cost analysis: Let us discuss the computational cost of distributed MASC. In what follows, denote by T the number of required consensus iterations and $\bar{k} = E\{|\mathcal{N}_j|\}$ the average number of neighbors of a node in the sensor network. The main computational steps that *each* sensor has to perform consists of (see also Algorithm 1):

- The construction of k-NN graph among the labelled examples that scales as O(l^2), where l denotes the number of labelled examples. However, this can be performed off-line (e.g., before the deployment of the sensor network).
- Local computation of the nearest neighbors of $x_j^{(u)}$ among the labeled data $X^{(l)}$. This requires computing the distance of $x_j^{(u)}$ to all labelled examples and scales as $O(l)$.
- Local computation of $r(\cdot)$ in Line 10. It scales as O(kc), because it involves only the last row of $\tilde{S}$ that contains only k non-zero entries (see also Fig. 3), where k is the set of nearest neighbors of each data sample in the data graph.
- Distributed computation of the objective function via distributed averaging in Line 11. This scales as O($\bar{k}Tc$), which corresponds to the cost of linear iteration (10), repeated T times until convergence, for each class hypothesis.

If we omit the off-line cost of forming the graph among the labelled samples, we conclude that the total average computational cost *per sensor* is O($l + (k + \bar{k}T)c$).

Given the fact the number of consensus iterations T increases when more sensors are added to the network, one would expect that the cost per sensor will also increase with the network size. However, one can practically overcome this problem by resorting to accelerated consensus methods, such as polynomial filtering [16], which admit an almost negligible increase of T with respect to the network size, by means

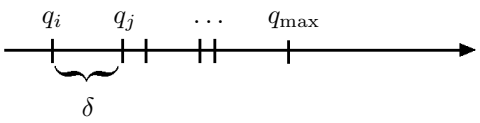


Fig. 4. The objective function values $q(p),\ p = 1, \ldots, c$, sorted in ascending order.

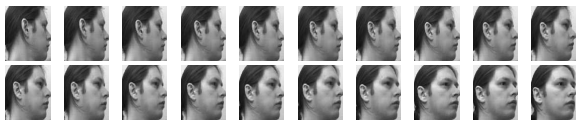


Fig. 5. Sample face images from the UMIST database. The number of different poses for each subject is varying.

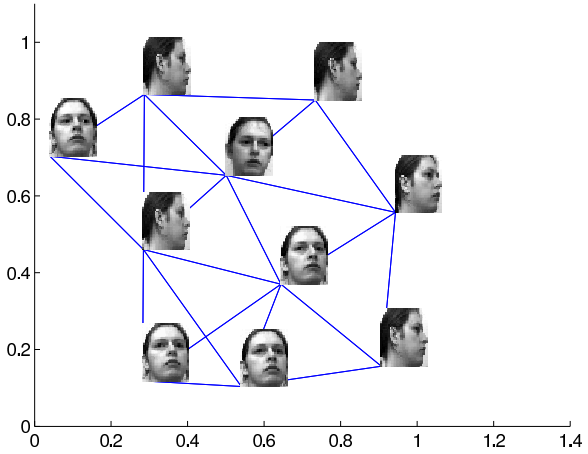


Fig. 6. Distributed multi-view face recognition in a vision sensor network. Each facial image corresponds to the observation of a sensor. The problem is to estimate the unknown class in a distributed fashion.

of increased convergence rate (see [16, Sec. V-B] for more details). Hence, distributed MASC is of very low complexity and thus appropriate for sensor networks.

Furthermore, the costs of communication stay similar to those of distributed average consensus solutions, which are very low. In particular, the number of messages *per sensor* scales as $\mathrm{O}(\bar{k}Tc)$, see also eq. (10).

C. Further remarks

Each sensor is able to provide an estimate of the unknown class even before the consensus process starts. This is possible by using its local r value as a (crude) approximation to the objective function value and looping over all class hypotheses. Then, while distributed consensus progresses, information from all observations is propagated over the network, the approximations to the objective function are refined and the partial classification decisions are updated. Eventually, the approximations of the objective function values converge and the sensors reach a consensus classification decision. The latter may even occur long before the function values stabilize. In what follows, we analyze why this is the case.

Observe that the consensus decision is reached when the approximation error of consensus at each sensor becomes smaller than half of the gap between the smallest q_i and second smallest q_j value of the objective function. Denote the gap between them by $\delta = q_j - q_i > 0$ as shown in Fig. 4. The marks on the horizontal axis represent the sorted list (in ascending order) of the objective function values $q(p)$ for $p = 1, \ldots, c$. Therefore, as long as the approximation error of consensus at each sensor is smaller than $\delta/2$, the order between the estimates $\tilde{q}_i$ and $\tilde{q}_j$ cannot change, and the consensus decision has been reached. From this point on, further consensus iterations will decrease the approximation error, but they will have no influence on the consensus decision.

V. Simulation results

A. Setup

We compare our distributed algorithm with a distributed baseline scheme for the classification of multiple observations consisting of k-NN followed by majority voting. Each sensor computes a local classification decision using k-NN classification on the labeled set $X^{(l)}$, and the final decision is obtained by majority voting across sensors. We also compare with two centralized algorithms: Label Propagation (see Sec. III-A) and centralized MASC (see Sec. III-B). In the centralized scenario, each algorithm has access to all observations $X^{(u)}$ and can further form a full similarity matrix $S \in \mathbb{R}^{n \times n}$. We illustrate the performance of all methods in distributed face recognition. Note that our goal is not to present a new method for multi-view face recognition, but rather to use this application as a showcase in order to illustrate the feasibility and the behavior of our distributed classification algorithm.

In the construction of the sensor networks, we use the random geographic graph model [17]. According to this model, we randomly distribute m sensor nodes on a 2-dimensional unit area. Two nodes are adjacent if their Euclidean distance is smaller than $\epsilon = \sqrt{\frac{\log m}{m}}$, which ensures connectedness with high probability. We also assign weights on the edges of the sensor network graph. We provide more information about the weights in the sequel in Section V-C.

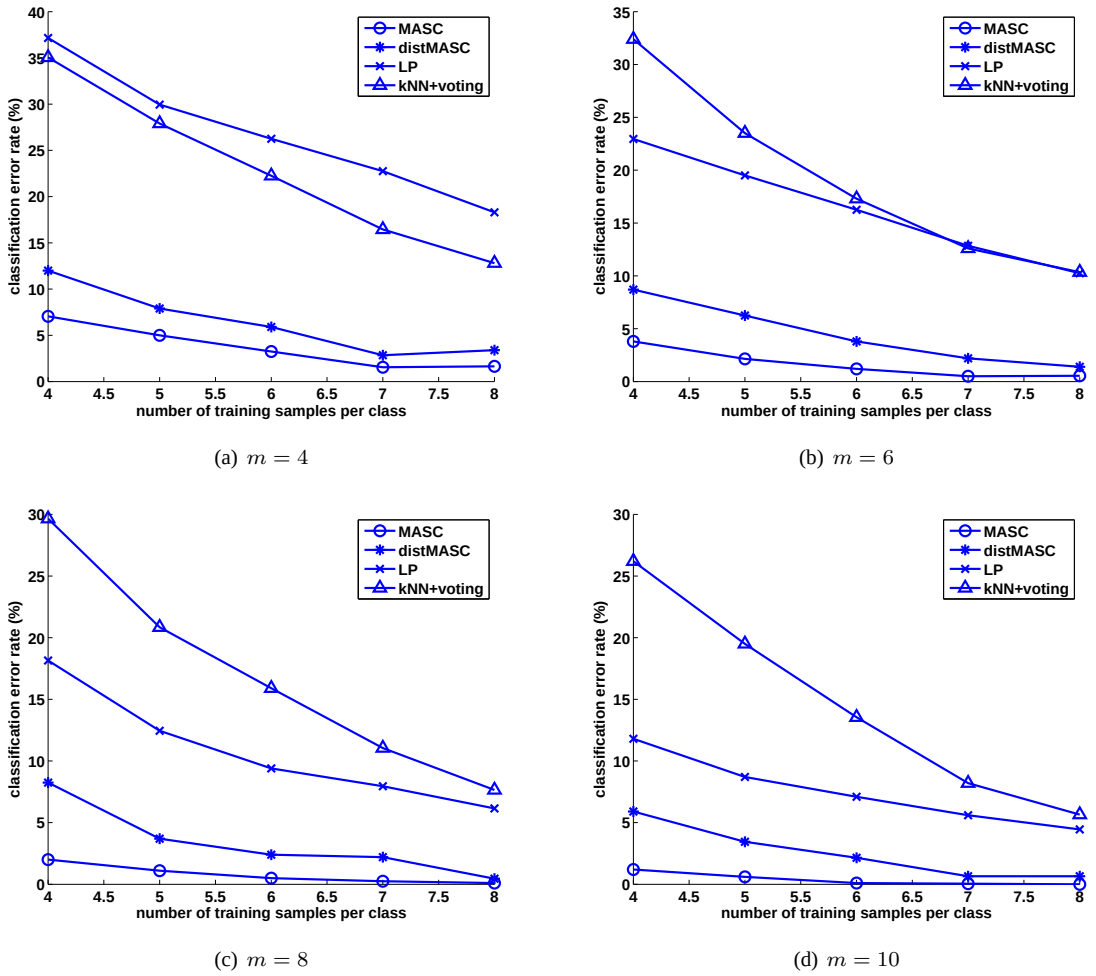
In all algorithms we use Gaussian weights defined as

$$H_{ij} = \begin{cases} \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}) & \text{when } (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \qquad (14)$$

where each x_i corresponds to a raw facial image represented as a high-dimensional vector in $\mathbb{R}^d$. The parameter σ in the above equation is set equal to half of the median of pairwise distances obtained from a large (random) sample of points. Finally, we set the number of nearest neighbors k to 3 in all methods.

We consider the case of a vision sensor network, such as the one shown in Fig. 1, where the face of a subject is captured by different cameras organized in an ad-hoc network. Each observation in this case represents a facial image captured under different viewing angles. Observe again that all observations belong to the same class and the problem resides in estimating the unknown class i.e., recognizing the subject.

We used the UMIST database [18] in our simulations. The UMIST database contains 20 people under different poses. The



(a) $m = 4$ (b) $m = 6$

(c) $m = 8$ (d) $m = 10$

Fig. 7. Difference in performance between MASC and its distributed version versus the number of training samples (per class).

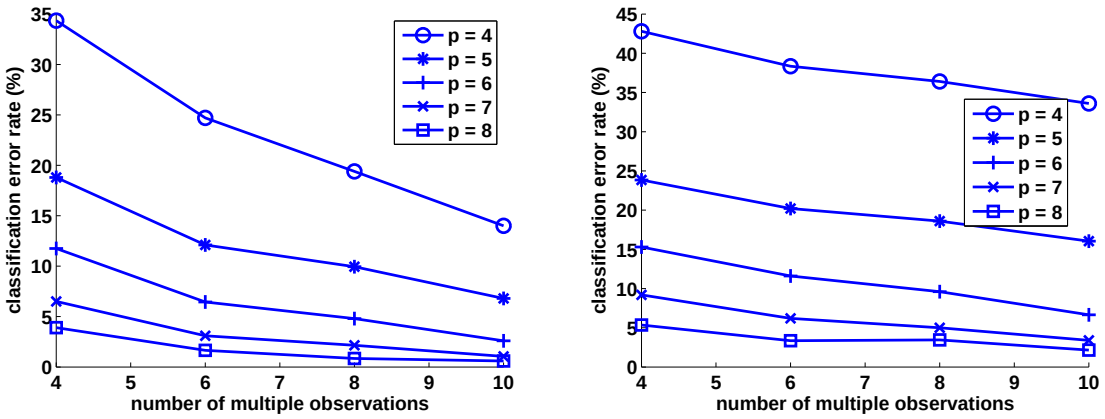
number of different views per subject varies from 19 to 48. Fig. 5 illustrates a sample subject from the UMIST database along with its first 20 views. Fig. 6 illustrates a snapshot of the simulated network. The facial image next to each sensor corresponds to its own observation. In order to simulate a generic scenario, we assign randomly the different face poses among the sensors.

B. Classification Performance

In the first experiment we will investigate the classification performances of all methods: distributed MASC, distributed k-NN + majority voting, centralized MASC and centralized Label Propagation (LP). We assume that the distributed average consensus in Line 11 of Algorithm 1 has converged to the asymptotic solution. In other words, we assume that the distributed summation is exact. The purpose of this experiment is to investigate whether the distributed algorithm suffers any loss in performance due to the partial information and what are the factors that influence this phenomenon. We set $\mu = 0.1$ in LP that worked best in this data set. We investigate the behavior of all methods, when the number of multiple observations m varies from 4 to 10 with step 2. For each particular value of m, we measure the classification error rate for different sizes of training set. In particular, we increase gradually the number of training examples per class and measure the average classification error rate over 100 random experiments. Each random experiment corresponds to a random split of the data set into training (labelled) and test (unlabelled) sets. We do many random experiments in order to avoid any bias in the measured classification performances, due to a particular realization of the labeled and unlabeled data sets.

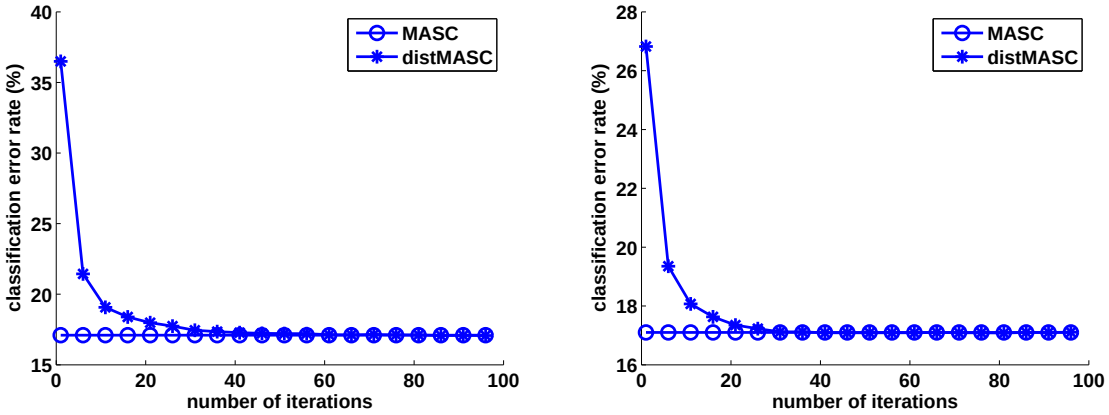
Figs 7(a)-7(d) show the obtained results for different number m of multiple observations, when the number of training examples per class increases from 4 to 8 with step 1. First, we see that distributed MASC outperforms the distributed baseline scheme of k-NN followed by majority voting as well as the (centralized) LP, which does not exploit the fact that all observations belong to the same class.

Second, we observe that there is a small loss in performance of distributed MASC with respect to its centralized counterpart. To see why this happens, it is important to realize that the k-NN graph in the distributed case is different than the graph in the centralized case. This is due to the fact that the multiple observations are collected distributively. Hence, the neighbors of an observation $x^{(u)}$ can be selected only among the labelled



(a) MASC (b) distMASC

Fig. 8. Classification performance versus number of multiple observations, for both methods. Each curve corresponds to different number of training samples per class.



(a) Maximum degree weights (b) Metropolis weights

Fig. 9. Average classification error rate vs consensus iterations, for different weight matrices.

examples, whereas in the centralized case they may be selected among all (labelled and unlabelled) examples. This is the main reason for the difference in performance in Fig. 7, which is more pronounced when the training set is small. However, it is exactly this difference in the construction of the k-NN graph that allows for the distributed MASC algorithm to have much lower computational cost than that of centralized MASC. Essentially, this is the main characteristic that makes it efficient and feasible in distributed settings. However, this comes at the cost of a small performance loss, which however reduces when the training set is sufficiently large.

Fig. 8 illustrates the same results as Fig. 7 in a different way. In particular, it illustrates the behavior of classification performances of both MASC methods with respect to the number of multiple observations, when the size of the training set is fixed. The number of multiple observations m varies from 4 to 10 with step 2. Each curve corresponds to a fixed number of training samples per class, denoted by p. Unsurprisingly, we observe that an increase in the number of observations tends to improve the classification performance in both algorithms.

C. Consensus Performance

In the previous experiment, we assumed that the distributed summation in Line 11 of Algorithm 1 is exact. In this experiment we drop this assumption and we investigate the effect of employing distributed consensus for the computation of this sum. Note that our goal in this particular experiment is to study the effect of consensus on the classification performances. For this reason, we use the same k-NN graph of distributed MASC in its centralized counterpart. This way, the performance difference of the two algorithms is only due to the summation part.

First, we split randomly the data set into training and test sets, by including two examples per class in the labelled set $X^{(l)}$ and the rest is assigned to the test set. We form $m = 10$ multiple observations, which are drawn randomly from the test set, and we use $k = 1$ in the construction of the k-NN graph.

Fig. 9 shows the average classification error rate (over 500 random experiments) measured on a certain sensor, say the first one, when the number of iterations in distributed consensus varies from 1 to 100 with step 5. Each random experiment

in this case corresponds to a random realization of the labelled and unlabelled data sets, as well as random generation of the underlying sensor network. We use two different weights from the literature[13], namely the Maximum-degree weights:

$$W(i,j) = \begin{cases} \frac{1}{n}, & (i,j) \in \mathcal{E}_s \\ 1 - \frac{d(i)}{n}, & i = j \\ 0 & \text{otherwise,} \end{cases} \quad (15)$$

and the Metropolis weights:

$$W(i,j) = \begin{cases} \frac{1}{1+\max\{d(i),d(j)\}}, & (i,j) \in \mathcal{E}_s \\ 1 - \sum_{(i,j)\in\mathcal{E}} W(i,k), & i = j \\ 0 & \text{otherwise,} \end{cases} \quad (16)$$

where $d(i)$ denotes the degree of the ith node. The weights above are known to satisfy condition (13) and therefore lead the iteration $z_{t+1} = Wz_t$ to asymptotic convergence to the average $\bar{z}_0 = \frac{1}{m}\sum_{i=1}^{m} z_0(i)$. Observe that fairly few iterations, namely between 30 and 40, provide sufficient accuracy in the computation of the distributed sum, in order to offer similar performance as the centralized MASC algorithm.

VI. Related work

In this section, we provide a more detailed exposition of the related work in the field. We start with consensus algorithms for various distributed problems in vision sensor networks and then we discuss distributed classification, first in general settings and then in relation to distributed consensus.

A. Consensus algorithms for vision sensor network problems

The methods that we are going to discuss below are not directly related to the algorithm proposed in this paper as they address different problems. However, we believe that it is advantageous to mention them as they are all based on distributed consensus, which further emphasizes the importance of the latter as a powerful tool for distributed information processing in vision sensor networks.

Distributed consensus [12], [13], [19], [20], [21] has recently become an important computational tool for multimedia data analysis and various aggregation tasks in ad-hoc sensor networks. In general, the main goal of distributed consensus is to reach a global solution iteratively in ad-hoc networks using only local computation and communication, while staying robust to changes in the network topology.

The authors in [22] propose a message-passing version of the Kalman-Consensus Filter (KCF) [23] for target tracking in sensor networks with a limited sensing range. The proposed algorithm reaches a consensus on estimates obtained by local Kalman filters in a hybrid architecture formed by a fusion center and a peer-to-peer network. Recently, this distributed tracking algorithm has been applied in [24] for tracking multiple targets in a self-configuring camera network.

The authors in [25], [6] have generalized the Euclidean distributed consensus algorithm to non-Euclidean manifolds. In particular, they have considered SE(3), which is the group of rigid-body transformations consisting of rotations in SO(3) and translations. They have applied their algorithm to distributed object pose estimation [25] as well as distributed face pose estimation [6]. A different approach is proposed in [26] for object pose averaging in distributed camera networks. It mainly differs from the approach above in that it includes a rigidity penalty term to distributed consensus, which penalizes the estimates that deviate from the model. Therefore, it bypasses the need for special handling of rotations.

B. Distributed classification

The authors in [9] propose a distributed multi-target classification algorithm for sensor networks. The authors formulate the classification problem as a multiple hypothesis testing problem and propose a decision fusion methodology by aggregating local classifier decisions to a fusion center. Since the number of hypothesis grows exponentially with the number of targets, the authors propose a sub-optimal approach of partitioning the hypothesis space.

A parallel active-set algorithm was proposed in [27] for distributed Support Vector Machines (SVM) training. The authors propose a relaxation to the dual of the SVM training optimization problem, which further permits the partition of the (relaxed) problem into subproblems that can be solved by Lagrangian decomposition and gradient projection. Despite the general scope of the proposed algorithm, the main focus has been on its computational efficiency, rather on its feasibility and implementation aspects in the context of wireless sensor networks.

The overview article [28] discusses the problem of distributed classification with non-parametric kernel methods [29], where the goal is to learn a global classification function from distributed data in wireless sensor networks. The method proposed in this work is fundamentally different from the methods discussed in [28] in that it tries to predict directly the single unknown class label based on the multiple observations, rather than trying to learn the classification function itself. The reader is referred to [28] and references therein, for more details on the related methods for nonparametric distributed learning.

Finally, we mention that there are approaches that address the problem by distributed feature extraction followed by (centralized) classification at the fusion center. For instance, Yang et. al. in [30] propose a distributed scheme for segmentation and classification of human actions using a network of wearable motion sensors. It is assumed that sensors are able to transmit local feature vectors to a central computer, where the global classification is performed.

C. Consensus-based distributed classification

Consensus-based methods for distributed classification in ad-hoc sensor networks have recently started to emerge. The authors in [5] propose two consensus algorithms for distributed SVM training for binary classification. The main idea of the first algorithm is to exchange support vectors between adjacent sensor nodes until consensus on the separating hyperplane has been reached. However, it was shown that it results in a sub-optimal solution. The second proposed algorithm computes

the optimal solution, at the price of increased communication though.

Another distributed SVM algorithm has been recently proposed in [31] that avoids the communication of support vectors between adjacent sensor nodes. The main idea is to cast the SVM optimization problem as the solution of several local convex optimization subproblems solved at each sensor, which are coupled by consensus constraints imposed on the classifier parameters (i.e., hyperplane and bias). The resulting problem is solved using the alternating direction method of multipliers [12] involving only node-to-node message exchanges. The generalization of the distributed algorithm to nonlinear SVMs is discussed in [32].

The above approaches are conceptually the closest to the method proposed in this work under the same perspective of being consensus-based. However, a few things should be kept in mind. First, SVMs are binary classifiers and, to the best of our knowledge, their multi-class extension to distributed settings has not been studied yet. On the contrary, our method inherently operates on multi-class problems. Second, the above methods, unlike our algorithm, have not been explicitly designed for the problem of multiple observations classification considered in this paper. Applying such methods directly on multiple observations will most likely result into several different estimated class labels available at each sensor and one is confronted then with the problem of fusing them in order to reach a single consensus decision. This is due to the fact that consensus is imposed on the classifier parameters and *not* on the estimated class label, as done by our method.

VII. Conclusions

We studied the problem of classification of multiple observations in the scenario where the observations are collected distributively. We showed that distributed classification in ad-hoc sensor networks can be effectively performed using distributed consensus. In particular, we proposed a distributed graph-based algorithm that aggregates information from all observations across the network and leads to a consensus classification decision among the sensors. We have illustrated its performance in the context of distributed multi-view face recognition. The simulation results have shown that, when the training set is sufficiently large, the classification decision of the distributed algorithm is equivalent to that of the centralized algorithm. Furthermore, the convergence of the distributed classification algorithm is very fast thanks to the effective consensus strategy.

References

[1] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf. The evolution from single to pervasive smart cameras. *2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, 2008.

[2] B. Rinner and W. Wolf. An introduction to distributed smart cameras. *Proceedings of the IEEE*, 96(10):1565–1575, October 2008.

[3] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury. Wireless Multimedia Sensor Networks: Applications and Testbeds. *Proceedings of the IEEE*, 2008.

[4] Akio Kosaka Johnny Park Gaurav Srivastava, Hidekazu Iwaki and Avinash Kak. Distributed and lightweight multi-camera human activity classification. *ACM/IEEE International Conference on Distributed Smart Cameras (ICDCS-09)*, September 2009.

[5] K. Flouri, B. Beferull-Lozano, and P. Tsakalides. Distributed consensus algorithms for SVM training in wireless sensor networks. *16th European Signal Processing Conference (EUSIPCO)*, 2008.

[6] R. Tron and R. Vidal. Distributed face recognition via consensus on SE(3). *8th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*, 2008.

[7] W. Schriebl, T. Winkler, A. Starzacher, and B. Rinner. A pervasive smart camera network architecture applied for multi-camera object classification. *ACM/IEEE International Conference on Distributed Smart Cameras (ICDCS-09)*, September 2009.

[8] J. B. Predd, S. R. Kulkarni, and H. V. Poor. Distributed learning in wireless sensor networks. In *Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing, Monticello*, IL, Sept 29-Oct 1 2004.

[9] J. H. Kotecha, V. Ramachandran, and A. M. Sayeed. Distributed multi-target classification in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):703–713, April 2005.

[10] D. Li, K. Wong, Y. Hen Hu, and A. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19(2), March 2002.

[11] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised learning*. MIT Press, 2006.

[12] D. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[13] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, (53):65–78, February 2004.

[14] D. Zhou, O. Bousquet, T. Navin Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[15] E. Kokiopoulou and P. Frossard. Graph-based classification of multiple observation sets. November 2009. http://arxiv.org/pdf/0810.4617, submitted.

[16] E. Kokiopoulou and P. Frossard. Polynomial filtering for fast convergence in distributed consensus. *IEEE Transactions on Signal Processing*, 57(1):342–354, 2009.

[17] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. on Information Theory*, 46(2):388–404, March 2000.

[18] D. B Graham and N. M Allinson. Characterizing virtual eigensignatures for general purpose face recognition. *Face Recognition: From Theory to Applications*, 163:446–456, 1998.

[19] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. *Int. Conf. on Information Processing in Sensor Networks*, pages 63–70, April 2005. Los Angeles.

[20] L. Xiao, S. Boyd, and S. Lall. Distributed average consensus with time-varying metropolis weights. *Automatica*, June 2006. submitted.

[21] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, January 2007.

[22] R. Olfati-Saber and N. F. Sandell. Distributed tracking in sensor networks with limited sensing range. *Proceedings of the American Control Conference*, June 2008.

[23] R. Olfati-Saber. Distributed kalman filtering algorithms for sensor networks. *IEEE Conference on Decision and Control*, December 2007.

[24] C. Soto, B. Song, and A. K. Roy-Chowdhury. Distributed multi-target tracking in a self-configuring camera network. *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[25] R. Tron, R. Vidal, and A. Terzis. Distributed pose averaging in camera networks via consensus on SE(3). *ACM/IEEE International Conference on Distributed Smart Cameras (ICDCS-08)*, September 2008.

[26] A. Jorstad, D. DeMenthon, I-J. Wang, and P. Burlina. Distributed consensus on camera pose. *IEEE Transactions on Image Processing*, 19(9):2396–2407, September 2010.

[27] T. Alpcan and C. Bauckhage. A discrete-time parallel update algorithm for distributed learning. *IEEE Int. Conf. on Pattern Recognition (ICPR)*, December 2008.

[28] J. B. Predd, S. R. Kulkarni, and H. V. Poor. Distributed learning in wireless sensor networks. *IEEE Signal Processing Magazine*, pages 56–69, July 2006.

[29] X. L. Nguyen. *Learning in decentralized systems: A nonparametric approach*. PhD thesis, University of California, Berkeley, 2007.

[30] A.Y. Yang, S. Iyengar, S. Sastry, R. Bajcsy, P. Kuryloski, and R. Jafari. Distributed segmentation and classification of human actions using a wearable motion sensor network. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, June 2008.

[31] P. Forero, A. Cano, and G. B. Giannakis. Consensus-based distributed linear support vector machines. In *9th ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, Stockholm, Sweden, April 2010.

[32] P. Forero, A. Cano, and G. B. Giannakis. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11:16631707, May 2010.

Effrosyni Kokiopoulou (S05,M09) received her Diploma in Engineering in June 2002, from the Computer Engineering and Informatics Department of the University of Patras, Greece. In June 2005, she received a M.Sc. degree in Computer Science from the Computer Science and Engineering Department of the University of Minnesota, USA, under the supervision of Prof. Yousef Saad. In September 2005, she joined as a PhD student the Signal Processing Laboratory (LTS4) at EPFL, Lausanne, Switzerland and completed her PhD studies in December 2008. Since 2009, she has been a postdoctoral researcher with the Seminar for Applied Mathematics, ETH, Zurich, Switzerland. Her research interests include multimedia data mining, pattern recognition, computer vision and numerical linear algebra.

Dr. Kokiopoulou is the 2010 winner of the ACM Special Interest Group on Multimedia (SIGMM) award for Outstanding PhD Thesis in Multimedia Computing, Communications and Applications. She has been elected to receive the EPFL doctorate award in 2010.

Pascal Frossard (S96,M01,SM04) received the M.S. and Ph.D. degrees, both in electrical engineering, from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1997 and 2000, respectively. Between 2001 and 2003, he was a member of the research staff at the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he worked on media coding and streaming technologies. Since 2003, he has been a professor at EPFL, where he heads the Signal Processing Laboratory (LTS4). His research interests include image representation and coding, visual information analysis, distributed image processing and communications, and media streaming systems.

Dr. Frossard has been the General Chair of IEEE ICME 2002 and Packet Video 2007. He has been the Technical Program Chair of EUSIPCO 2008, and a member of the organizing or technical program committees of numerous conferences. He has been an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA (2004-2010), the IEEE TRANSACTIONS ON IMAGE PROCESSING (2010-) and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2006-). He is an elected member of the IEEE Image and Multidimensional Signal Processing Technical Committee (2007-), the IEEE Visual Signal Processing and Communications Technical Committee (2006-), and the IEEE Multimedia Systems and Applications Technical Committee (2005-). He has served as Vice-Chair of the IEEE Multimedia Communications Technical Committee (2004-2006) and as a member of the IEEE Multimedia Signal Processing Technical Committee (2004-2007). He received the Swiss NSF Professorship Award in 2003, the IBM Faculty Award in 2005 and the IBM Exploratory Stream Analytics Innovation Award in 2008.