Sensors **2013**, *13*, 8750-8770; doi:10.3390/s130708750

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

Multi-View Human Activity Recognition in Distributed Camera Sensor Networks

Ehsan Adeli Mosabbeb [1], Kaamran Raahemifar [2,*] and Mahmood Fathy [1,*]

[1] Computer Engineering Department, Iran University of Science and Technology, Narmak, Tehran 16846-13114, Iran; E-Mail: eadeli@iust.ac.ir

[2] Department of Electrical and Computer Engineering, Ryerson University, 350 Victoria Street, Toronto, ON M5B 2K3, Canada

* Authors to whom correspondence should be addressed; E-Mails: kraahemi@ee.ryerson.ca (K.R.); mahfathy@iust.ac.ir (M.F.); Tel.: +1-416-979-5000 (ext. 6097) (K.R.); Fax: +1-416-979-5280 (K.R.).

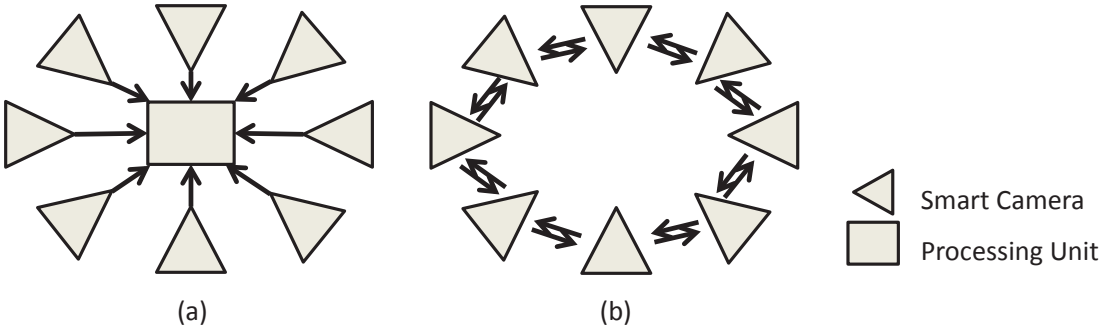
Abstract: With the increasing demand on the usage of smart and networked cameras in intelligent and ambient technology environments, development of algorithms for such resource-distributed networks are of great interest. Multi-view action recognition addresses many challenges dealing with view-invariance and occlusion, and due to the huge amount of processing and communicating data in real life applications, it is not easy to adapt these methods for use in smart camera networks. In this paper, we propose a distributed activity classification framework, in which we assume that several camera sensors are observing the scene. Each camera processes its own observations, and while communicating with other cameras, they come to an agreement about the activity class. Our method is based on recovering a low-rank matrix over consensus to perform a distributed matrix completion via convex optimization. Then, it is applied to the problem of human activity classification. We test our approach on IXMAS and MuHAVi datasets to show the performance and the feasibility of the method.

Keywords: human activity recognition; camera sensor networks; consensus; convex optimization; matrix completion; nuclear norm

1. Introduction

A camera sensor network (CSN) is defined as a set of vision sensors, which can communicate through a network. Each of these smart camera nodes also has its own processing element and memory. With such settings, many applications could be addressed, due to the ease of deployment and their robustness. For instance, creating smart homes, intelligent environments and robot coordination are some great potential applications, which can lead us to a better quality of life. Traditional systems make each camera transmit its own image data or low-level features over the network to a centralized processing unit, which analyzes everything in a centralized fashion (Figure 1(a)). However, this needs a huge amount of processing and communication and requires dealing with a large amount of data. To address this problem, we can develop distributed algorithms, in which each camera deals with its own image (data) and communicates with other cameras in the network. To analyze the whole scene, the cameras collaborate and come to a decision together via fusing their own local analysis (Figure 1(b)) [1,2].

Figure 1. (**a**) Centralized camera network setup; (**b**) distributed camera network setup.



Human action recognition has been proven to have many applications, including vision-based surveillance [3,4], human-computer interaction [5], patient and healthcare monitoring systems [6], smart homes and environments [7] and a lot more [8,9]. This makes it a very important field in computer vision studies. With the development of smart camera technology and networks, the huge amount of processing for such high level applications could be performed in a more robust and scalable way. Several previous works have developed many computer vision applications in such distributed environments [1]. Some also have targeted the activity recognition problem [10–12].

Understanding the events and activities of humans in video sequences is a challenging task, due to several different issues, including: (1) the large variability in the imaging conditions, as well as the way different people perform a particular action; (2) the background clutter and motion; (3) the high dimensionality of such data is another significant challenge for recognition problems; and (4) a huge amount of occlusion in real-world environments. Many previous works have targeted these challenges by introducing different sets of features [13,14] and classifiers and have achieved good results. One of the best methods to overcome many of these challenges is to analyze the activities from multiple views and, therefore, acquire more information about the activity for better understanding. However, this makes it even harder, since there will be more amounts of data to be processed, and on the other hand, the fusion of the information across the views is a hard task. Therefore, camera sensor networks could create a great

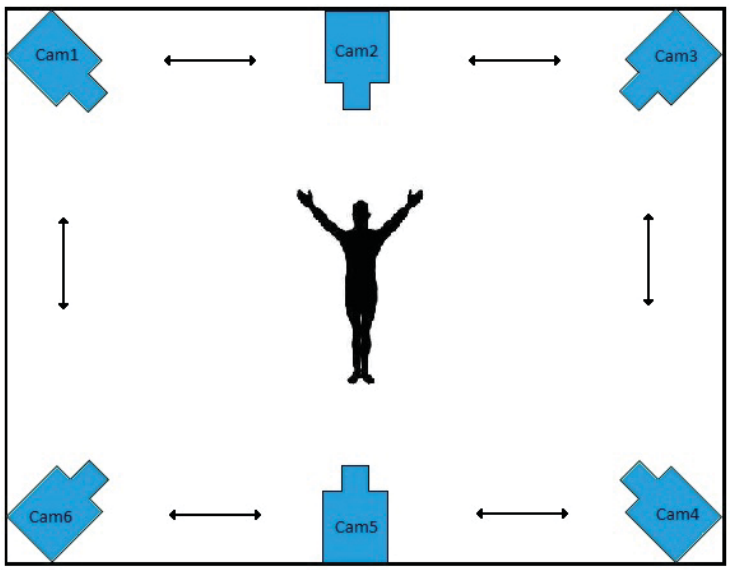
bed for such applications, where the processing could be distributed among the cameras and the decision about the scene could be made in a distributed manner via communication and fusion of features.

Rank Minimization has recently gained a lot of attention, due to the simple, effective success in solving many problems. As noted by [15], the minimization of the rank function can be achieved using the minimizer obtained by the nuclear norm, which is calculated as the sum of singular values. In the field of computer vision, nuclear norm minimization has been applied to many problems, such as camera calibration [16], structure from motion [17], image segmentation [18] and image categorization [19].

In this paper, we develop a method for the recognition of human activities portrayed in multi-view video sequences. Our method is based on matrix completion, which finds the best action label(s) for each test scene. Each view is composed of a single smart camera, which locally processes its video sequence and decides about the activity being performed in the scene via communication. A sample configuration of the smart cameras for activity recognition is depicted in Figure 2. Each scene is represented with a number of fixed length histograms of densely sampled features, which captures both the visual content and the temporal changes in the scene. This makes the method independent from the video content, view point and imaging conditions. In real applications, there is a lot of clutter and noise present in the scene, from the background and/or the imaging conditions, besides the variability in performing the actions by the subjects. Our low-rank matrix recovery framework is capable of taking out the noise and the outliers, efficiently. In this paper, a consensus-based distributed algorithm for matrix completion is presented and is applied for activity recognition in camera sensor networks. The algorithm is based on singular value thresholding to minimize the nuclear norm and enjoys a convex formulation. The minimization problem is solved via the Alternating Direction method of Multipliers (ADM) [20].

In the rest of the paper, the next section reviews the previous work, Section 3 explains our distributed matrix completion technique and the proceeding section explains the proposed activity recognition approach in detail. Section 5 outlines a set of experiments for distributed activity recognition. Finally, Section 6 concludes the paper.

Figure 2. Sample camera network setup for human activity recognition.



2. Related Works

Action and activity recognition methods from single-view video sequences could be categorized into three classes: (1) models that directly utilize bag-of-words (BoWs) representations [21,22]; (2) approaches that decompose an action into smaller parts for capturing the local spatial or temporal structure of the activity and to better model the interaction between parts [23]; (3) approaches that use the global spatio-temporal templates, such as motion history, spatio-temporal shapes, the human model changing in time or other templates [24]. These approaches try to retain the visual shape and structure of the activity. As shown by Laptev *et al.* [13], compared to simple bag-of-words [21], approaches encoding the spatio-temporal layout of a video using a fixed space-time grid enhance the recognition rates.

Several multi-camera and distributed action recognition approaches have also been proposed in the literature [10–12,25–28], which aimed at extending single-view techniques for the multi-view case. Sirvastava *et al.* [10] use spatio-temporal interest points from each single view. This method is specifically designed for a network of low-powered camera sensors. Song *et al.* [11] use a Markov chain with a known transition matrix to model the actions. There are also several papers proposing fusion strategies for multi-view action recognition [29]. Wu *et al.* [25] use the best view as a simple strategy for fusion, whereas [12] uses data from all views for the classification task. In [11], the authors use a probabilistic consensus method for fusing the similarity scores of neighboring cameras.

Matrix completion is a great tool for classification purposes, where the instances are classified through convex optimization for best labels and, simultaneously, finding the error and outliers present in the data. The problem of matrix completion and rank minimization is initially a non-convex optimization problem [30,31], which is simply based on factorizing the matrix into two matrices of a rank of at most r. However, recently, rank minimization has gained attention and is achieved by using the minimizer obtained with the nuclear norm [15]. In order to solve this convex rank minimization problem, many approaches are developed, such as Iterative Thresholding [15,32], Fixed Point Continuation [33], the Augmented Lagrangian Multipliers method [32] and the Alternating Direction method [34].

Distributed algorithms for matrix factorization and low rank recovery mostly include using parallel or distributed programming models, such as MapReduce and Hadoop. For instance, [35,36] are designed for MapReduce and [37] for the second version of Hadoop. The drawbacks of these models are that they are limited to the restrictive programming models and mostly suffer from run-time overheads. Furthermore, the cluster management is hard, and optimal configuration of the nodes is not obvious. Other approaches in this area include introducing a separable regularization for the nuclear norm, which makes the process distribution much easier [38,39]. These approaches use the Alternating Direction method or Stochastic Gradient Descent approaches for the optimization process. However, these regularizations or approaches that factorize the main matrix into two lower-rank matrices suggest non-convex objectives.

In this paper, a distributed algorithm is proposed, which uses a convex formulation of matrix completion and is applied to the problem of multi-view activity recognition in a network of smart cameras.

3. Distributed Matrix Completion

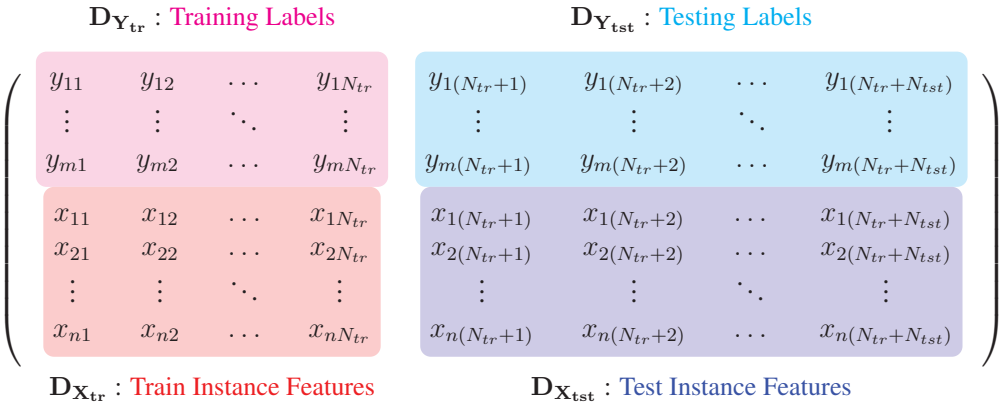
3.1. Network Setup

Let's assume that the network of the processing nodes or the smart cameras is modeled with a connected undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \{1, \ldots, N_p\}$ as the set of camera nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ representing the nodes that can communicate with each other. With this definition, each node, i, can have some neighbors denoted by $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$ and the degree, $d_i = |\mathcal{N}_i|$.

3.2. Matrix Completion for Classification

Matrix Completion is the process of recovering a matrix from a sampling of its entries. We want to recover a data matrix, $\mathbf{D}$, from a matrix, $\mathbf{D_0}$, in which we only get to observe a number of its entries, which is comparably much smaller than the total number of elements in the matrix. Let Ω denote the set of known entries. With sufficiently large measurements and uniformly distributed entries in the matrix, we can assume that there is only one low-rank matrix with these entries [15]. As denoted by [15,30], if a matrix has rank r, it should have exactly r nonzero singular values. Thus, the rank function could be simply defined as the number of non-vanishing singular values (σ_k). Therefore, a simple estimate of the rank function can be defined as $\|\mathbf{D}\|_* = \sum_{k=1}^{d} \sigma_k(\mathbf{D})$, which is called the nuclear or trace norm. Recently, this formulation has been used for classification tasks. The task is to learn the connection between the space of features, X, and the space of labels, Y, from N_{tr} training instances. Let m be the number of different classes, n the dimensionality of the feature space, N the number of total instances and N_{tr} and N_{tst} the number of training and testing instances, respectively.

Figure 3. Data matrix, $\mathbf{D_0}$, which contains training and testing instances, each as a single column.



As noted by Goldberg *et al.* [33] the problem of classifying N_{tst} test entries can be cast as a matrix completion task. To this end, we can concatenate all labels and features into a single matrix (as illustrated in Figure 3). If a linear classification model holds, this matrix should be rank-deficient. In this formulation, the classification process would be defined as filling the unknown entries in Y_{tst}, such that the nuclear norm of $\mathbf{D_0}$ is minimized. This could be done via a convex minimization

process [33,34,40]. In practice, we have errors and incomplete data in the training features and labels. Therefore, we define the set of known entries in $\mathbf{D_0}$ as Ω_X and Ω_Y and zero out unknown entries:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D_Y} \\ \mathbf{D_X} \\ \mathbf{D_1} \end{bmatrix} = \begin{bmatrix} \mathbf{Y_{tr}} & \mathbf{Y_{tst}} \\ \mathbf{X_{tr}} & \mathbf{X_{tst}} \\ \mathbf{1}^\top & \end{bmatrix} + \begin{bmatrix} \mathbf{E_{Y_{tr}}} & \mathbf{0} \\ \mathbf{E_{X_{tr}}} & \mathbf{E_{X_{tst}}} \\ \mathbf{0}^\top & \end{bmatrix} \tag{1}$$

where $\mathbf{Y_{tr}} \in \mathbb{R}^{m \times N_{tr}}$ and $\mathbf{Y_{tst}} \in \mathbb{R}^{m \times N_{tst}}$ are the training and testing labels and $\mathbf{X_{tr}} \in \mathbb{R}^{n \times N_{tr}}$ and $\mathbf{X_{tst}} \in \mathbb{R}^{n \times N_{tst}}$ are the training and testing feature vectors, respectively. Therefore, the classification process would be posed as finding the best $\mathbf{Y_{tst}}$ and the error matrix, $\mathbf{E}$, such that the rank of $\mathbf{D} = \mathbf{D_0} + \mathbf{E}$ is minimized [33]. This would be equivalent to [40]:

$$\begin{aligned} \min_{\mathbf{D}} \quad & \gamma \|\mathbf{D}\|_* + \frac{1}{|\Omega_X|} \sum_{ij \in \Omega_X} c_x(\mathbf{E_{X_{ij}}}) + \frac{\lambda_1}{|\Omega_Y|} \sum_{ij \in \Omega_Y} c_y(\mathbf{E_{Y_{ij}}}) \\ \text{subject to} \quad & \mathbf{D} = \mathbf{D_0} + \mathbf{E}, \\ & \mathbf{D_1} = \mathbf{1}^\top \end{aligned} \tag{2}$$

where $c_y(.)$ is a log loss function and $c_x(.)$ is a least squares error. These two terms are to avoid trivial solutions and to penalize large distortions of $\mathbf{D}$. The parameters, γ and λ_1, are positive trade-off weights [33,40]. This minimization problem can be solved using a Fixed Point Continuation (FPC) method [33] or an Alternating Direction method (ADM) [34].

3.3. Distributed Nuclear Norm Minimization for Matrix Completion

As shown by [33,41], as long as the error matrix, $\mathbf{E}$, is sufficiently sparse, we can exactly recover the low-rank matrix, $\mathbf{D}$, from $\mathbf{D_0} = \mathbf{D} + \mathbf{E}$ by solving the convex optimization problem, Equation (2). Let us, for simplicity, replace the second and the third terms in the objective function in Equation (2) with $f(\mathbf{E_X})$ and $g(\mathbf{E_Y})$, respectively. By introducing a Lagrangian multiplier, the Lagrangian function would be:

$$\mathcal{L}(\mathbf{D}, \mathbf{E}, \mathscr{L}) = \gamma \|\mathbf{D}\|_* + f(\mathbf{E_X}) + g(\mathbf{E_Y}) + + \langle \mathscr{L}, \mathbf{D_0} - \mathbf{D} - \mathbf{E} \rangle + \frac{\mu}{2} \|\mathbf{D_0} - \mathbf{D} - \mathbf{E}\|_F^2 \tag{3}$$

Using the iterative thresholding or the singular value thresholding (SVT) algorithm [41,42] and the Alternating Direction method, Problem (2) could be solved by updating each variable, while keeping the others fixed. $\mathbf{D}$ and $\mathbf{E}$ are calculated by minimizing $\mathcal{L}(\mathbf{D}, \mathbf{E}, \mathscr{L})$, and then, the amount of violation, $\mathbf{D_0} - \mathbf{D} - \mathbf{E}$, is used to update $\mathscr{L}$. A shrinkage operator as a proximal operator for the nuclear norm could be defined as:

$$\mathcal{S}_\epsilon[x] = \begin{cases} x - \epsilon & \text{if } x > \epsilon \\ x + \epsilon & \text{if } x < -\epsilon \\ 0 & otherwise. \end{cases} \tag{4}$$

With the singular value decomposition of a matrix, $\mathbf{USV}^\top$, we can apply an Alternating Direction method (ADM) for recovering the low-rank matrix, $\mathbf{D}$, via an iterative optimization procedure, as

proposed by [32,41]. For this purpose, we need to iterate to optimize the above Lagrangian function for the $\mathbf{E}$ and $\mathbf{D}$ matrices. The error matrices, $\mathbf{E_X}$ and $\mathbf{E_Y}$, would have a closed form solution, by solving the following two subproblems in each k^{th} iteration:

$$\begin{aligned}\mathbf{E}_{\mathbf{X_{k+1}}} &= \underset{\mathbf{E_X}}{\operatorname{argmin}} \frac{1}{\mu_k} f(\mathbf{E}_{\mathbf{X_k}}) + \frac{1}{2}\|\mathbf{E}_{\mathbf{X_k}} - (\mathbf{D_{0X}} - \mathbf{D}_{\mathbf{X_{k+1}}} + \frac{\mathscr{L}_k}{\mu_k})\|_F^2 \\ \mathbf{E}_{\mathbf{Y_{k+1}}} &= \underset{\mathbf{E_Y}}{\operatorname{argmin}} \frac{1}{\mu_k} g(\mathbf{E}_{\mathbf{Y_k}}) + \frac{1}{2}\|\mathbf{E}_{\mathbf{Y_k}} - (\mathbf{D_{0Y}} - \mathbf{D}_{\mathbf{Y_{k+1}}} + \frac{\mathscr{L}_k}{\mu_k})\|_F^2\end{aligned} \tag{5}$$

where μ_k is the step parameter and is increased in each iteration. On the other hand, the nuclear norm of the matrix, $\mathbf{D}$, is minimized using the SVT algorithm [42], where the proximal operator, $\mathcal{S}_\epsilon[.]$, is applied on the singular values of the matrix, $\mathbf{D_0} - \mathbf{E_k} + \mu_k^{-1}\mathscr{L}_k$, to construct the matrix, $\mathbf{D}$, in each k^{th} iteration as:

$$\begin{aligned}(\mathbf{U}, \mathbf{S}, \mathbf{V}) &= svd(\mathbf{D_0} - \mathbf{E_k} + \mu_k^{-1}\mathscr{L}_k) \\ \mathbf{D_{k+1}} &= \mathbf{U}\mathcal{S}_{\mu_k^{-1}}[\mathbf{S}]\mathbf{V}^\top.\end{aligned} \tag{6}$$

The constraint, $\mathbf{D_1} = \mathbf{1}^\top$, is enforced by keeping the last row of $\mathbf{E_k}$ equal to $\mathbf{0}^\top$. Furthermore, for all unknown entries, $(i, j) \in \Omega_Y$, the choice of $\mathbf{E_k}(i, j) = 0$ holds [32].

In order to parallelize this algorithm, we need to distribute the entries present in $\mathbf{D_0}$ between the processing nodes. Therefore, we will have separate $\mathbf{E}$ matrices for each node, and accordingly, we will require the use of the corresponding Lagrangian multipliers. Suppose that we split the data matrix, $\mathbf{D} \in \mathbb{R}^{(n+m)\times(N_{tr}+N_{tst})}$, into N_p parts, $\mathbf{D_i} \in \mathbb{R}^{n_i\times(N_{tr}+N_{tst})}$. Therefore, we can assume that the original data matrix is formed as:

$$\mathbf{D} = [\mathbf{D_1}^\top, \mathbf{D_2}^\top, \ldots, \mathbf{D_{N_p}}^\top]^\top \in \mathbb{R}^{(n+m)\times(N_{tr}+N_{tst})}. \tag{7}$$

Therefore, the Lagrangian multiplier, $\mathscr{L}$, and the error matrix, $\mathbf{E}$, would also be split in the same manner. Now, we will have an equivalent problem, as in Equation (2), for each single processing node, i. The Lagrangian function, from each node's point of view, would be:

$$\gamma\|\mathbf{D}\|_* + f(\mathbf{E}_{\mathbf{X_i}}) + g(\mathbf{E}_{\mathbf{Y_i}}) + \langle\mathscr{L}_i, \mathbf{D_i} - \mathbf{D_{0_i}} - \mathbf{E_i}\rangle + \frac{\mu}{2}\|\mathbf{D_i} - \mathbf{D_{0_i}} - \mathbf{E_i}\|_F^2 \tag{8}$$

where $\mathscr{L}_i$s are the Lagrange multipliers. The only shared problem between the nodes is the minimization of the nuclear norm of the whole data matrix, where we need to calculate the SVDof the $\mathbf{J} = \mathbf{D_0} - \mathbf{E_k} + \mu_k^{-1}\mathbf{Y_k}$ matrix, collaboratively. First, suppose we want to compute $\frac{1}{N_p}\mathbf{J}^\top\mathbf{J}$:

$$\mathbf{C} = \frac{1}{N_p}\mathbf{J}^\top\mathbf{J} = \frac{1}{N_p}\sum_{i=1}^{N_p}\mathbf{J_i}^\top\mathbf{J_i} = \frac{1}{N_p}\sum_{i=1}^{N_p}\mathbf{C_i}. \tag{9}$$

$\mathbf{C_i} = \mathbf{J_i}^\top\mathbf{J_i}$ could be denoted as the local correlation matrix. As could be seen, this problem would be distributed on the nodes. This is very easy to compute through consensus, since it is a simple averaging

of data present in each node. Initially, each node has a local state, $\mathbf{c_i}(0) = \mathbf{C_i}$; in each iteration, nodes receive the internal state of their neighbors and update:

$$\mathbf{c_i}(t+1) = \mathbf{c_i}(t) + \mathcal{W}(t) \sum_{j \in \mathcal{N}_i} (\mathbf{c_j}(t) - \mathbf{c_i}(t)) \tag{10}$$

where $\mathcal{W}(t)$ is initially set to $(max_i\{d_i\})^{-1}$ and decreased through time. It is shown [43] that each state converges to the average of the initial values in each node ($\lim_{t\to\infty} \mathbf{c_i} = \mathbf{C}$), no matter how the network configuration is and if there is partial noise in the communications. The consensus would be achieved when $|\mathbf{c_i}(t+1) - \mathbf{c_i}(t)| \leq \epsilon$, with ϵ as a very small constant threshold.

Note that $\mathbf{C}$ is a $(n+m) \times (n+m)$ matrix, independent from N_p. Therefore, if the number of processing nodes and the number of data splits grow, $\mathbf{C}$ still could be correctly recovered. In order to compute the SVD of the matrix, $\mathbf{J}$, we need to calculate matrices, $\mathbf{U} \in \mathbb{R}^{(n+m)\times r}$, $\mathbf{V} \in \mathbb{R}^{r\times(N_{tr}+N_{tst})}$ and $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$, with r as the rank of the matrix: $\mathbf{J} = \mathbf{U\Sigma V}^\top$. To do this, we can compute the SVD of $\mathbf{C}$, which would be equal to $\mathbf{V}(\frac{1}{N_p}\mathbf{\Sigma}^2)\mathbf{V}^\top$. Therefore, after distributed averaging, each node can recover $\mathbf{V}$, and if they know N_p, they also can recover $\mathbf{\Sigma}$. These two matrices will be common for all the nodes and easy to calculate, and they can compute their own share of the matrix, $\mathbf{U}$ as: $\mathbf{U_i} = \mathbf{J_i V\Sigma}^{-1}$.

As a result, the SVD operation could be calculated in a distributed manner, and each node can recover the complete matrix, $\mathbf{\Sigma}$, and then it can apply the shrinkage operator and iterate to optimize the rank of the data matrix. In order to minimize the rank of the matrix, $\mathbf{D}$, in each k^{th} iteration, the following set of instructions should be executed on each node, i, until a consensus is achieved:

$$\begin{aligned}
&\mathbf{J_{i_k}} = \mathbf{D_{i_0}} - \mathbf{E_{i_k}} + \mu_k^{-1}\mathbf{Y_{i_k}} \\
&\mathbf{C_i} = \mathbf{J_{i_k}}^\top \mathbf{J_{i_k}} \\
&\text{Calculate } \mathbf{C} \text{ via consensus using Equation (10),} \\
&(\mathbf{V}, \frac{1}{N_p}\mathbf{\Sigma}^2, \mathbf{V}^\top) = svd(\mathbf{C}) \\
&\mathbf{U_i} = \mathbf{J_{i_k}} \mathbf{V\Sigma}^{-1} \\
&\mathbf{D_{i_{k+1}}} = \mathbf{U_i}\mathcal{S}_\tau[\mathbf{\Sigma}]\mathbf{V}^\top
\end{aligned} \tag{11}$$

In summary, this algorithm consists of two stages: first, calculating $\boldsymbol{C}$ via consensus over the network and, then, performing the iterative thresholding algorithm for minimizing the nuclear norm. The first stage is performed by iterating on Equation (10), while receiving the local $\boldsymbol{C_i}$ variables from the neighboring nodes, in each iteration. This is continued until the $\boldsymbol{C_i}$ variables converge. We can benefit from a joint treatment and create an inexact version of the algorithm, where the iterative operations for calculating the $\boldsymbol{C_i}$s is not performed completely to reach the convergence. Only one iteration gives us a fast good estimate of the $\boldsymbol{C}$ matrix and would satisfy the convergence properties of the whole algorithm. When a good estimate could be achieved for the optimization subproblem, ADM would still converge, probably with more numbers of iterations [32]. The distributed matrix completion algorithm on each processing node, i, is outlined in Algorithm 1.

Algorithm 1 Distributed matrix completion algorithm for recognition, on the i^{th} processing node.

Input: Initial portion of the data matrix for the i^{th} node, $\mathbf{D_i} = \mathbf{D_{0_i}}$, and parameter, λ.
Output: i^{th} portion of the completed matrix, $\mathbf{D_i}$

$\mathscr{L}_{i_0} = 0, \mu_k > 0, \rho = 1.1, \mathbf{E_{i_0}} = \mathbf{0}$
while not converged **do**

1. Fix all other variables and update $\mathbf{D_{i_{k+1}}} = \underset{\mathbf{D_i}}{\operatorname{argmin}} \frac{1}{\mu_k}\|\mathbf{D_{i_k}}\|_* + \frac{1}{2}\|\mathbf{D_{i_k}} - (\mathbf{D_{0_i}} + \mathbf{E_{i_k}} - \frac{\mathscr{L}_{i_k}}{\mu_k})\|_F^2$ by:
   $\mathbf{J_{i_k}} = \mathbf{D_{i_0}} - \mathbf{E_{i_k}} + \mu_k^{-1}\mathscr{L}_{\mathbf{i_k}}$
   $\mathbf{c_i}(k) = \mathbf{J_{i_k}}^\top \mathbf{J_{i_k}}$
   Send $\mathbf{c_i}(k)$ to all the neighbors, $\mathcal{N}_i$,
   Receive all $\mathbf{c_j}(k)$s from the neighbors, $\mathcal{N}_i$,
   $\mathbf{c_i}(k) = \mathbf{c_i}(k) + \mathcal{W}(k)\sum_{j\in\mathcal{N}_i}(\mathbf{c_j}(k) - \mathbf{c_i}(k))$
   $(\mathbf{V}, \frac{1}{N_p}\mathbf{\Sigma}^2, \mathbf{V}^\top) = svd(\mathbf{c_i}(k))$
   $\mathbf{U_i} = \mathbf{J_{i_k}}\mathbf{V}\mathbf{\Sigma}^{-1}$
   $\mathbf{D_{i_{k+1}}} = \mathbf{U_i}\mathcal{S}_\tau[\mathbf{\Sigma}]\mathbf{V}^\top$
2. Fix all other variables and update
   $\mathbf{E_{X_{i_{k+1}}}} = \underset{\mathbf{E_{X_i}}}{\operatorname{argmin}} \frac{1}{\mu_k} f(\mathbf{E_{X_{i_k}}}) + \frac{1}{2}\|\mathbf{E_{X_{i_k}}} - (\mathbf{D_{0X_i}} - \mathbf{D_{X_{i_{k+1}}}} + \frac{\mathscr{L}_{i_k}}{\mu_k})\|_F^2$
3. Fix all other variables and update
   $\mathbf{E_{Y_{i_{k+1}}}} = \underset{\mathbf{E_{Y_i}}}{\operatorname{argmin}} \frac{1}{\mu_k} g(\mathbf{E_{Y_{i_k}}}) + \frac{1}{2}\|\mathbf{E_{Y_{i_k}}} - (\mathbf{D_{0Y_i}} - \mathbf{D_{Y_{i_{k+1}}}} + \frac{\mathscr{L}_{i_k}}{\mu_k})\|_F^2$
4. Set the $\mathbf{E_{i_k}} = \left[\mathbf{E_{Y_{i_k}}}^\top \quad \mathbf{E_{X_{i_k}}}^\top \quad \mathbf{0}^\top\right]^\top$,
5. Update the multiplier, $\mathscr{L}_i$:
   $\mathscr{L}_{i_{k+1}} = \mathscr{L}_{i_k} + \mu_k(\mathbf{D_{i_{k+1}}} - \mathbf{D_{i_0}} - \mathbf{E_{i_{k+1}}})$
7. Update parameter, μ_{k+1}, as: $\mu_{k+1} = \min(\rho\mu_k, 10^{10})$ and $k = k + 1$.
8. Check the convergence condition:
   $(\mathbf{D_{i_k}} - \mathbf{D_{0_i}} - \mathbf{E_{i_k}} \to 0)$

end while

4. Distributed Activity Recognition

Our task is to recognize activities present in the scene, which are captured with a networked set of cameras, as also illustrated in Figure 2. The distributed environment, as described in Section 3.1, is composed of a number of cameras with processing power and communication skills. Each scene is represented with a fix-length feature vector from each camera's view point. The recognition task would be to classify these feature vectors into one of the predefined activity classes. This is performed in a distributed manner via consensus, as will be described in this section.

4.1. Scene Representation

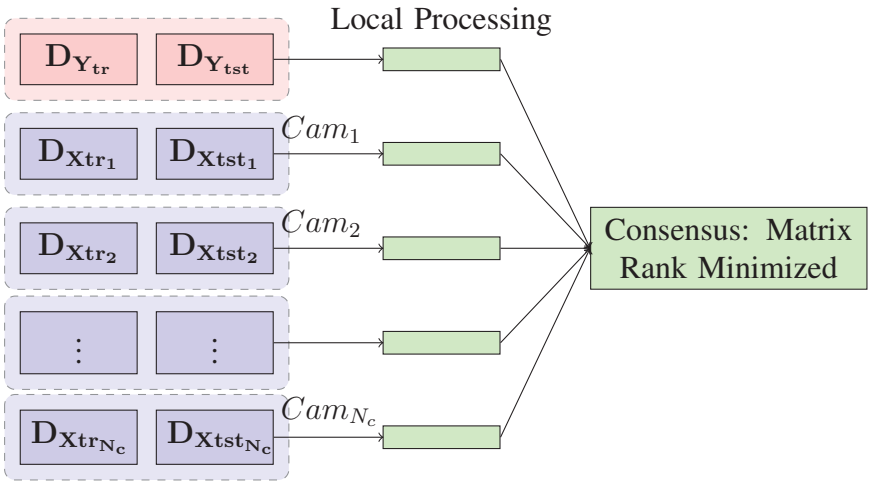
To represent each video from each view, we use histograms of densely sampled features, which extract features from space-time video blocks and sample from five dimensions, (x, y, t, σ, τ). σ and τ are the spatial and temporal scales, respectively. We use a histogram of gradient (HoG) and a histogram of optical flow (HoF) [13]. These histograms are computed on a regular grid at three different scales.

For each descriptor (HoG, HoF), an independent dictionary is used. This is done by using K-means and quantizing all descriptors to the closest ℓ_2 distance dictionary element. The concatenation of both histograms forms the scene descriptor from a camera's view point. These histogram features have been extensively used for object and activity recognition in a single view [8,23] and also extended for multi-view [10]. With these feature vectors, there is no need to perform any background subtraction, tracking or silhouette extraction, which makes the algorithm faster and independent from contextual noise. As a result, each scene, i, is composed of a histogram feature vector, $\mathbf{h_i^j}$, from the j^{th} view. Therefore, scene $\mathbf{S_i}$ is described by $\{\mathbf{h_i^1}, \mathbf{h_i^2}, \ldots, \mathbf{h_i^{N_c}}\}$. These sets of features are almost independent from variations in the activity orientation. However, in order to further make sure that the orientation of the activities with regard to the cameras does not strengthen noise and outliers, we employ a cycling approach, as proposed by [10]. This is explained in more detail in the next subsection.

4.2. Training and Testing Scenarios

We can assume that both train and test action sequences are captured by N_c cameras. With the above representation, each scene is described with a histogram of quantized features from each view. Therefore, each camera has its own part of the scene description. We can model the distribution of the data matrix, $\mathbf{D_0}$ for our case, as shown in Figure 4. The data matrix (as in Equation (1)) is split between the processing nodes, row-wise. Each node will hold one part of the data segment (both train and test). The label's sub-matrix (upper row in Figure 4) is also assigned to a single node.

Figure 4. A model for the data split between the processing camera nodes (distributing segments of each activity between the nodes).



We construct the matrix, $\mathbf{D_0}$, by assigning each column to training or testing samples. During the process of capturing the sequences of each action, the subject could be facing any of the cameras performing the action. For training, the samples are formed, such that all the sequences have the same orientation formation. Therefore, in order to enhance the recognition results, for each test sequence, we need to determine the orientation for which the action can best perform the recognition. The correspondence could be determined using a circular shift. For instance, consider an action scene, $\mathbf{S_i} = \{\mathbf{h_i^1}, \mathbf{h_i^2}, \mathbf{h_i^3}, \mathbf{h_i^4}\}$, in case of four camera views. The circularly shifted versions are:

$\{\mathbf{h_i^1}, \mathbf{h_i^2}, \mathbf{h_i^3}, \mathbf{h_i^4}\}$, $\{\mathbf{h_i^4}, \mathbf{h_i^1}, \mathbf{h_i^2}, \mathbf{h_i^3}\}$, $\{\mathbf{h_i^3}, \mathbf{h_i^4}, \mathbf{h_i^1}, \mathbf{h_i^2}\}$ and $\{\mathbf{h_i^2}, \mathbf{h_i^3}, \mathbf{h_i^4}, \mathbf{h_i^1}\}$, which cover all possible conditions, where the action may face any of the cameras.

When performing a matrix completion, for determining the labels, all four combinations are considered, and the one with the least amount of absolute error in the corresponding row of the error matrix, $\mathbf{E_X}$, is chosen, and the action class would be determined by its corresponding column in $\mathbf{D_Y}$.

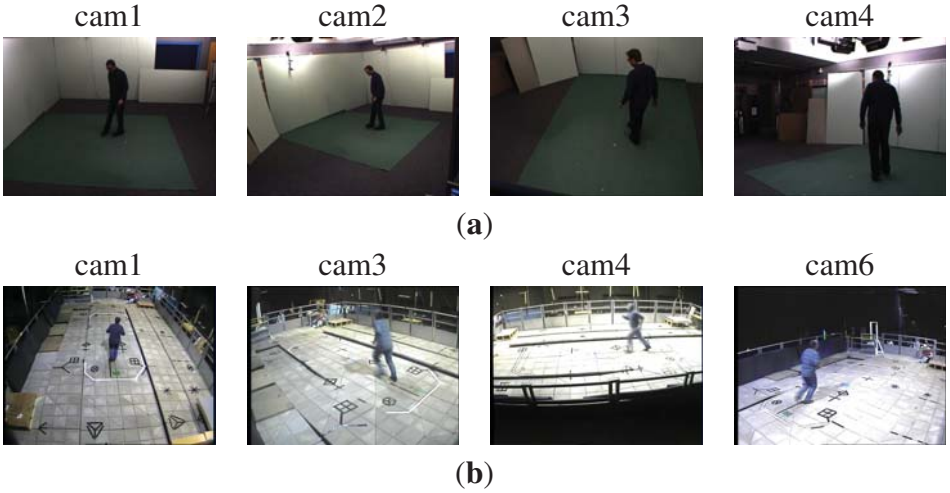
5. Experiments

In this section, we setup several experiments on some well-known multi-view activity datasets and compare the recognition results with some state-of-the-art distributed and centralized methods. We choose previous methods, which have reported results with the same experimental setup for comparisons. We also compare the execution times of our distributed matrix completion algorithm with those of the original centralized version of the algorithm, solving Equation (2) using ADM, on the same datasets. The recognition accuracies are calculated as the average of per-class recognition rates, for each experiment. Recognition results for each single view are also computed by running a matrix completion scheme on the features from that specific view.

5.1. Human Action Datasets

In order to validate our approach, we carried out experiments using the IXMAS [44] and MuHAVi [45] datasets. Figure 5 shows some sample frames from these datasets.

Figure 5. Sample frames from the action datasets. (**a**) IXMAS; (**b**) MuHAVi.



The IXMAS dataset has 13 action classes (check watch, cross arms, scratch head, sit down, get up, turn around, walk, wave, punch, kick, point, pick up, throw over head and throw from bottom up) performed by 12 subjects, each 3 times. The scene is captured by 5 cameras, and the calibration/synchronization parameters are provided. In order to be consistent with a setup similar to those in the previous work [10,44], we discard images from camera 5, which is the top view and does not have much informative information for our purpose. This dataset is a challenging one, due to the fact

that subjects freely choose their position and orientation. Therefore, each camera has captured different viewing angles, which makes the recognition task harder.

The MuHAVi dataset contains 17 action classes (walk turn back, run stop, punch, kick, shotgun collapse, pull heavy object, pickup throw object, walk fall, look in car, crawl on knees, wave arms, draw graffiti, jump over fence, drunk walk, climb ladder, smash object and jump over gap) performed by 7 actors, recorded in 25 fps with challenging lighting conditions. In our experiments, we choose four (two side and two corner) cameras for evaluations. A manually annotated subset (MuHAVi-MAS) is also available, which provides silhouettes for two of these views (front-side and corner) for two actors, labeled 14 (called MuHAVi-14). We run our experiments on the whole dataset, since we did not require the manually annotated silhouettes, but we compare our method with some state-of-the-art methods on MuHAVi-14.

5.2. Experimental Setup

To setup this experiment, we have simulated the network environment, where each camera process is implemented in a single process on a processing core of a Corei7-3610QM CPU, and the communication is done via IPC. The network of the cameras is considered to have a fully connected topology.

For extracting the spatio-temporal interest points and to form the histogram feature vectors, we set $\sigma = 2$ and $\tau = 3$. For the feature extraction phase, the size of the space-time patches are considered to be 18×18 pixels and 10 frames. The samplings are also done with 50% overlap, as also introduced by [46]. For evaluating the experiments, the leave-one-out cross-validation strategy is employed, where videos of one subject are used for testing, and videos of the remaining subjects are considered as the training instances.

5.3. Results

IXMAS: Figure 6 shows the results of the classification on each individual camera for the IXMAS dataset, compared with the distributed algorithm that uses the data from all the views. This figure shows how the distributed algorithm can outperform each of the single views, and that is because it can describe each action in a more descriptive way from different views. Figure 7 outlines the confusion matrix of the distributed activity recognition, and Table 1 shows the overall recognition rate in comparison with some state-of-the-art methods. As is obvious, the WaveHand action is the most deceptive one and could be mistaken with other actions. Different experiments from different previous work use 11 or all 13 actions from the dataset. We run our method and report results on both. Figures 8 and 9 also show the class-level recognition accuracies in comparisons with some state-of-the-art methods. As could be seen in these figures, our method has better recognition rates, even for those actions that are not well-recognized by other competitors.

MuHAVi: The classification results for every individual camera using our method, in comparisons with our distributed algorithm, are shown in Figure 10, and as expected, the distributed algorithm achieves better recognition results. In this figure, the results for each camera indicate a training and testing scenario on that single view, while the all-view method trains and tests our distributed algorithm. The confusion matrix is also plotted in Figure 11 and the overall recognition rate in comparison

with some state-of-the-art methods is shown in Table 2. A class-level comparisons with another state-of-the-art method is provided in Figure 12. This dataset is not as challenging as the IXMAS dataset, since the subjects are not performing the actions freely. The subjects perform the actions with predefined orientations. That is why our method and most of the previous methods get better recognition results on this dataset, compared to the IXMAS dataset.

Figure 6. Recognition results for each of the single views and all four views, on the IXMAS dataset with training and testing on 11 actions and 10 subjects.

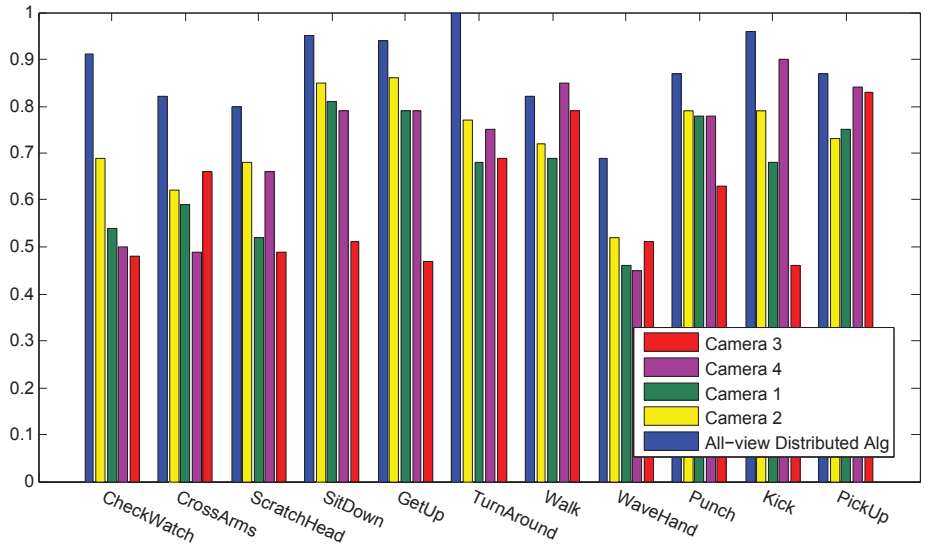


Figure 7. The confusion matrix of the recognition output on the IXMAS dataset.

	Check Watch	Cross Arms	Scratch Head	Sit Down	Get Up	Turn Around	Walk	Wave Hand	Punch	Kick	Pick Up
Check Watch	0.91	0	0	0	0	0.04	0	0.04	0	0	0
Cross Arms	0.09	0.82	0	0	0	0.02	0	0.02	0.04	0	0
Scratch Head	0	0.03	0.8	0	0	0.02	0	0.02	0.03	0.1	0
Sit Down	0	0	0	0.95	0.02	0	0.03	0	0	0	0
Get Up	0	0	0	0	0.94	0.06	0	0	0	0	0
Turn Around	0	0	0	0	0	1	0	0	0	0	0
Walk	0	0	0	0	0	0.18	0.82	0	0	0	0
Wave Hand	0.07	0.09	0.04	0.02	0.01	0	0	0.69	0.08	0	0
Punch	0	0	0	0.03	0	0.04	0	0	0.87	0	0.06
Kick	0	0	0	0	0	0	0.04	0	0	0.96	0
Pick Up	0	0	0	0.08	0	0	0	0.02	0.02	0	0.87

Table 1. Overall accuracy results on the IXMAS dataset, using all four cameras. # Sub. and # Act. in the table are the number of subjects and the number of actions taken into account for evaluation in the method, respectively.

Approach	# Act.	# Sub.	Method	Accuracy
Srivastava *et al.* [10]	10	11	Distributed	81.4%
Weinland *et al.* [44]	10	11	Centralized	81.3%
Our Method	10	11	Distributed	87.5%
Liu and Shah [47]	13	12	Centralized	82.8%
Reddy *et al.* [48]	13	12	Centralized	66.5%
Wu and Jia [49]	12	12	View-invariant	91.67%
Our Method	13	12	Distributed	85.9%

Many actions are very hard to recognize if they are viewed from a specific view point. However, our distributed algorithm achieves better recognition rates, compared to each single view of the same dataset. As could be seen, our method outperforms several distributed or centralized methods, both as an overall recognition system or in the class-level. Only Wu and Jia [49] achieve better results on these datasets. They use a non-linear classification method with a specific kernel designed for view-invariant classification, while our method enjoys a linear classification scheme, which is capable of being adapted for any large-scale or distributed classification problem.

Figure 8. Class-level recognition results of the IXMAS dataset with 11 actions, in comparison with Shao *et al.* [50] and Weinland *et al.* [44].

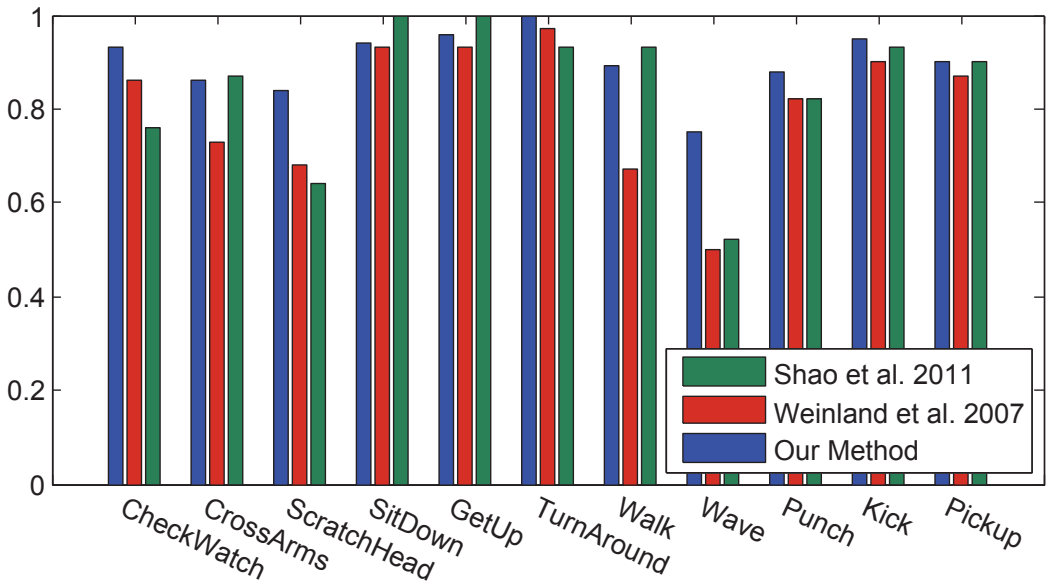


Figure 9. Class-level recognition results of the IXMAS dataset with 13 actions, in comparison with Reddy *et al*. [48] and Liu and Shah [47].

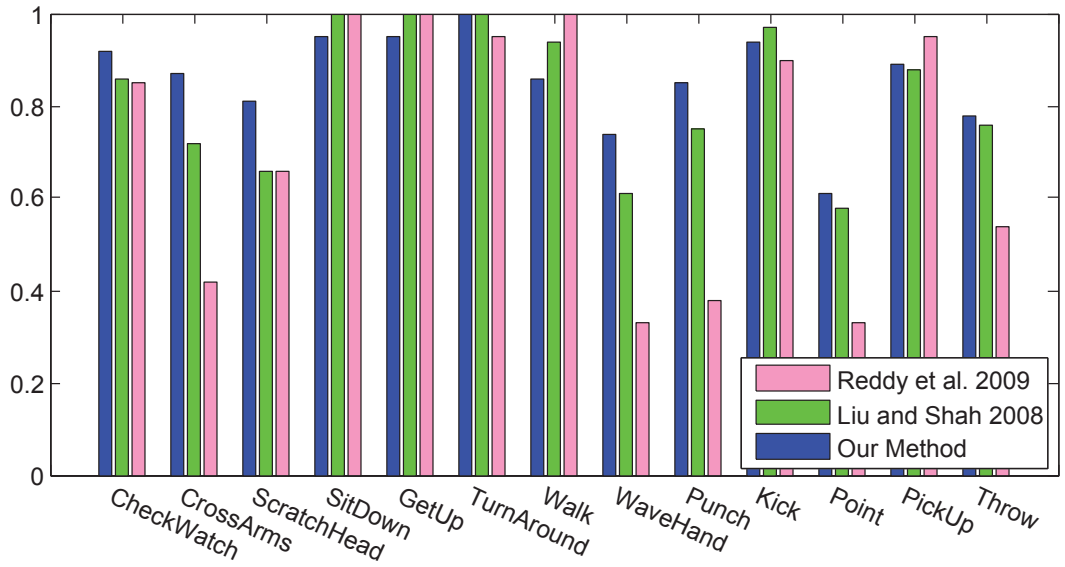


Figure 10. Recognition results for each of the single views and all four views, on the MuHAVi dataset.

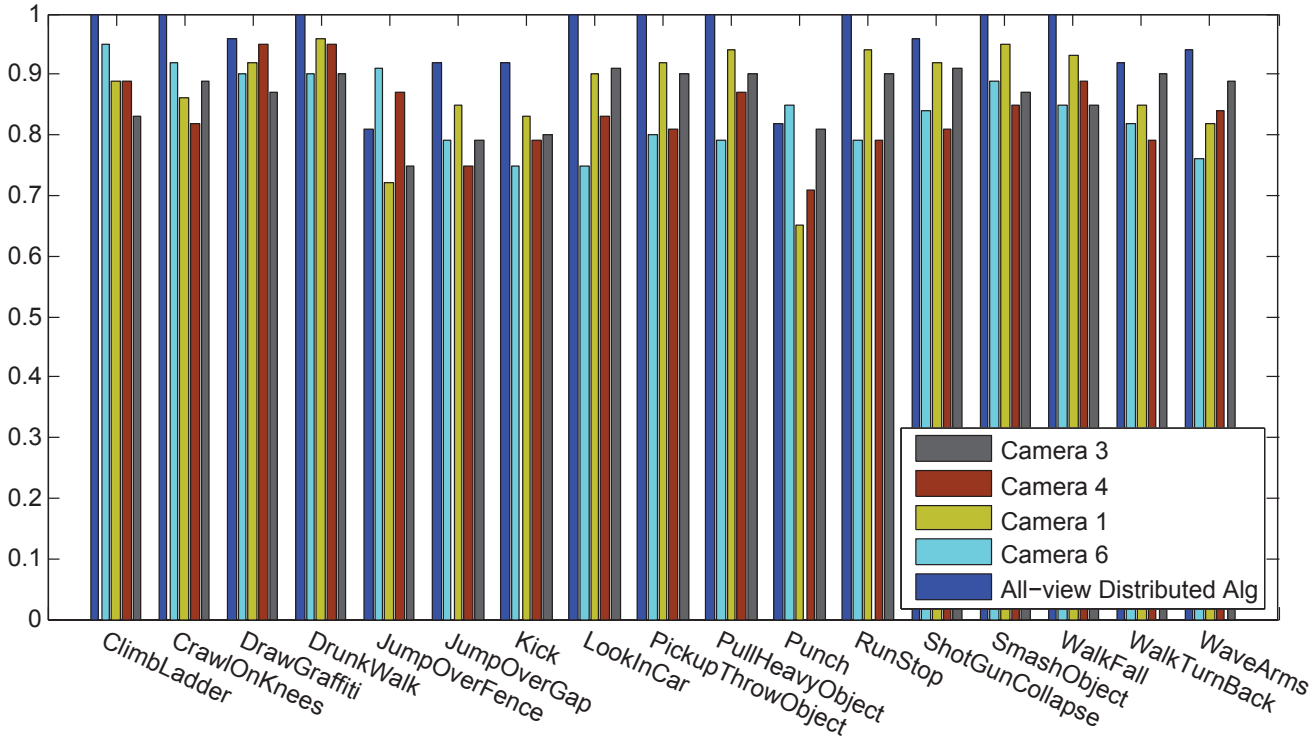
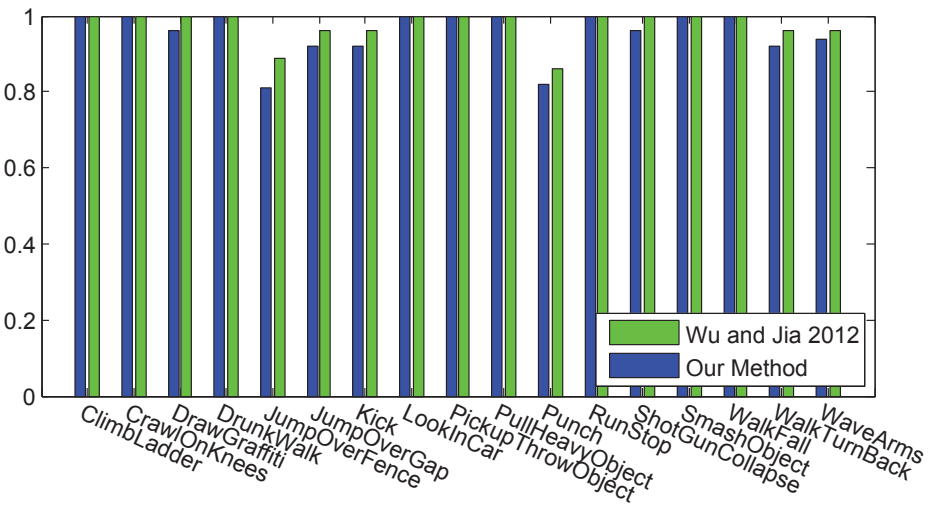


Figure 11. The confusion matrix of the recognition output on the MuHAVi dataset.

	ClimbLadder	CrawlOnKnees	DrawGraffiti	DrunkWalk	JumpOverFence	JumpOverGap	Kick	LookInCar	PickupThrowObject	PullHeavyObject	Punch	RunStop	ShotgunCollapse	SmashObject	WalkFall	WalkTurnBack	WaveArms
ClimbLadder	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CrawlOnKnees	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DrawGraffiti	0	0	.96	0	0	0	0	0	0	0	0	0	0	.04	0	0	0
DrunkWalk	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0
JumpOverFence	0	.08	.06	0	.81	.05	0	0	0	0	0	0	0	0	0	0	0
JumpOverGap	0	0	0	.08	0	.92	0	0	0	0	0	0	0	0	0	0	0
Kick	0	0	0	0	0	0	.92	0	0	0	.08	0	0	0	0	0	0
LookInCar	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0
PickupThrowObject	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0
PullHeavyObject	0	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0
Punch	0	0	0	0	0	0	.10	0	.08	0	.82	0	0	0	0	0	0
RunStop	0	0	0	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0
ShotgunCollapse	0	0	0	0	0	0	0	0	0	0	0	0	.96	0	.04	0	0
SmashObject	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0	0	0	0
WalkFall	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0	0	0
WalkTurnBack	0	0	0	.02	0	0	0	0	0	0	0	.06	0	0	0	.92	0
WaveArms	0	0	0	0	0	0	0	.06	0	0	0	0	0	0	0	0	.94

Figure 12. Class-level recognition results of the MuHAVi dataset, in comparison with Wu and Jia [49].



In order to evaluate the boost in the run time, the execution times of the runs on the two versions of the algorithm are calculated. Figure 13 shows the execution time of each set of data together with the communication and load overheads. As is obvious, the distributed algorithm gets the same recognition results in a shorter time, as expected. The centralized matrix completion algorithm is run on the same machine in which the distributed algorithm was simulated, but on a single core. Note that these reported execution times do not include the circular shifting between the cameras.

Figure 13. Execution times for the distributed and centralized matrix completion on human activity recognition datasets. (**a**) IXMAS Dataset; (**b**) MuHAVi Dataset.

Process Communication + Load Centralized MC 1815 25 Distributed MC 410 151 .tex Process Communication +

(**a**)

Table 2. Overall accuracy results on the MuHAVi dataset. The data column shows the subset of the data used for evaluation for each of the methods.

Approach	Data	Method	Accuracy
Singh *et al.* [45]	MuHAVi-14	Centralized	82.4%
Chaaraoui *et al.* [51]	MuHAVi-14	Centralized	91.2%
Wu and Jia [49]	All of the dataset	View-invariant	97.48%
Our method	All of the dataset	Distributed	95.59%

6. Conclusion and Discussions

In this paper, we have described a distributed action recognition algorithm, based on low-rank matrix completion. We have proposed a simple distributed algorithm to minimize the nuclear norm of a matrix, and then, we have adapted an inexact augmenting Lagrangian multiplier method to solve the matrix completion problem. We have tested the algorithm on IXMAS and MuHAVi datasets and achieved good results. With the experiments outlined in this paper, we show that our matrix completion framework could be well adapted for the classification of a scene in a distributed camera network. Therefore, it is a proof-of-concept study for using such algorithms in distributed computer vision algorithms.

As mentioned before, we have developed a distributed classification framework for human action recognition, which can also be used for distributed classification tasks. Matrix completion is a great tool for dealing with noisy data. As could be seen in the formulations, the error and outliers are identified during the minimization task. Activity recognition data, due to its many variations across subjects and

imaging/illumination conditions, is a set of data with many potential outliers, and that is why our method could achieve acceptable results, compared to the other state-of-the-art method.

As a direction for future work, we need to perform the training and testing procedures incrementally, where huge amounts of data could be summarized into smaller matrices and used for testing purposes.

Conflict of Interest

The authors declare no conflict of interest.

References

1. Tron, R.; Vidal, R. Distributed computer vision algorithms. *IEEE Signal Process. Mag.* **2011**, *28*, 32–45.
2. Aghajan, H.; Cavallaro, A. *Multi-Camera Networks: Principles and Applications*; Academic Press: Waltham, MA, USA, 2009.
3. Choi, J.; Dumortier, Y.; Prokaj, J.; Medioni, G. Activity Recognition in Wide Aerial Video Surveillance Using Entity Relationship Models. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 2012 (SIGSPATIAL '12), Redondo Beach, CA, USA, 6–9 November 2012; pp. 466–469.
4. Ng, L.L.; Chua, H.S. Vision-Based Activities Recognition by Trajectory Analysis for Parking Lot Surveillance. In Proceedings of 2012 IEEE International Conference on Circuits and Systems (ICCAS), Kuala Lumpur, Malaysia, 3–4 October 2012; pp. 137–142.
5. Zhu, Y.; Xu, G.; Kriegman, D.J. A real-time approach to the spotting, representation, and recognition of hand gestures for human-computer interaction. *Comput. Vis. Image Underst.* **2002**, *85*, 189–208.
6. Martnez-Prez, F.E.; Gonzlez-Fraga, J.N.; Cuevas-Tello, J.C.; Rodrguez, M.D. Activity inference for ambient intelligence through handling artifacts in a healthcare environment. *Sensors* **2012**, *12*, 1072–1099.
7. Fatima, I.; Fahim, M.; Lee, Y.K.; Lee, S. A unified framework for activity recognition-based behavior analysis and action prediction in smart homes. *Sensors* **2013**, *13*, 2682–2699.
8. Aggarwal, J.; Ryoo, M. Human activity analysis: A review. *ACM Comput. Surv.* **2011**, *43*, doi:10.1109/NAMW.1997.609859
9. Xu, X.; Tang, J.; Zhang, X.; Liu, X.; Zhang, H.; Qiu, Y. Exploring techniques for vision based human activity recognition: Methods, systems, and evaluation. *Sensors* **2013**, *13*, 1635–1650.
10. Srivastava, G.; Iwaki, H.; Park, J.; Kak, A. Distributed and Lightweight Multi-Camera Human Activity Classification. In Proceedings of Third ACM/IEEE International Conference on Distributed Smart Cameras, Como, Italy, 30 August–2 September 2009; pp. 1–8.
11. Song, B.; Kamal, A.; Soto, C.; Ding, C.; Farrell, J.; Roy-Chowdhury, A. Tracking and activity recognition through consensus in distributed camera networks. *IEEE Trans. Image Process.* **2010**, *19*, 2564–2579.

12. Ramagiri, S.; Kavi, R.; Kulathumani, V. Real-Time Multi-View Human Action Recognition Using a Wireless Camera Network. In Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras, Ghent, Belgium, 22–25 August 2011; pp. 1–6.
13. Laptev, I.; Marszałek, M.; Schmid, C.; Rozenfeld, B. Learning Realistic Human Actions from Movies. In Proceedings of Conference on Computer Vision & Pattern Recognition, Anchorage, AK, USA, 24–26 June 2008; pp. 1–8.
14. Ryoo, M.S.; Aggarwal, J.K. Spatio-Temporal Relationship Match: Video Structure Comparison for Recognition of Complex Human Activities. In Proceedings of International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009; pp. 1593–1600.
15. Candès, E.J.; Recht, B. Exact matrix completion via convex optimization. *Found. Comput. Math.* **2009**, *9*, 717–772.
16. Zhang, Z.; Matsushita, Y.; Ma, Y. Camera Calibration with Lens Distortion from Low-Rank Textures. In Proceedings of 2011 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 20–25 June 2011; IEEE Computer Society: Washington, DC, USA, 2011; pp. 2321–2328.
17. Dai, Y.; Li, H.; He, M. Element-Wise Factorization for N-View Projective Reconstruction. In Proceedings of the 11th European Conference on Computer Vision: Part IV, Heraklion, Greece, 5–11 September 2010; pp. 396–409.
18. Cheng, B.; Liu, G.; Wang, J.; Huang, Z.; Yan, S. Multi-Task Low-Rank Affinity Pursuit for Image Segmentation. In Proceedings of International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2439–2446.
19. Cabral, R.; de la Torre, F.; Costeira, J.P.; Bernardino, A. Matrix completion for weakly-supervised multi-label Image classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, in press.
20. Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **2011**, *3*, 1–122.
21. Schuldt, C.; Laptev, I.; Caputo, B. Recognizing Human Actions: A Local SVM Approach. In Proceedings of IEEE Computer Society International Conference on Pattern Recognition, Cambridge, UK, 23–26 August 2004; Volume 3, pp. 32–36.
22. Wu, S.; Oreifej, O.; Shah, M. Action Recognition in Videos Acquired by a Moving Camera Using Motion Decomposition of Lagrangian Particle Trajectories. In Proceedings of International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 1419–1426.
23. Raptis, M.; Kokkinos, I.; Soatto, S. Discovering Discriminative Action Parts from Mid-Level Video Representations. In Proceedings of International Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 1242–1249.
24. Bobick, A.F.; Davis, J.W. The recognition of human movement using temporal templates. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 257–267.
25. Wu, C.; Khalili, A.H.; Aghajan, H. Multi-view activity recognition in smart homes with spatio-temporal features. In Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras, Atalanta, GA, USA, 31 August–4 September 2010; pp. 142–149.

26. Holte, M.; Tran, C.; Trivedi, M.; Moeslund, T. Human pose estimation and activity recognition from multi-view videos: Comparative explorations of recent developments. *IEEE J. Sel. Top. Signal Process.* **2012**, *6*, 538–552.
27. Gu, Y.; Yuan, C. Human Action Recognition for Home Sensor Network. In Proceedings of the 2012 International Conference on Information Technology and Software Engineering; Lu, W., Cai, G., Liu, W., Xing, W., Eds.; Springer Berlin Heidelberg: Berlin, Germany, 2013; Volume 212; pp. 643–656.
28. W Parameswaran, V.; Chellappa, R. View invariance for human action recognition. *Int. J. Comput. Vis.* **2006**, *66*, 83–101.
29. Banos, O.; Damas, M.; Pomares, H.; Rojas, I. On the use of sensor fusion to reduce the impact of rotational and additive noise in human activity recognition. *Sensors* **2012**, *12*, 8039–8054.
30. Fazel, M. Matrix Rank Minimization with Applications. Ph.D. Thesis, Stanford University, Stanford, CA, USA, March 2002.
31. Xu, Y.; Yin, W.; Wen, Z.; Zhang, Y. An alternating direction algorithm for matrix completion with nonnegative factors. *Front. Math. China* **2012**, *7*, 365–384.
32. Lin, Z.; Chen, M.; Wu, L.; Ma, Y. *The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices*, University of Illinois at Urbana-Champaign Technical Report UILU-ENG-09-2215 (2009); 2009; pp. 1–20.
33. Goldberg, A.B.; Zhu, X.; Recht, B.; Xu, J.M.; Nowak, R.D. Transduction with Matrix Completion: Three Birds with One Stone. In *Neural Information Processing Systems*; Curran Associates, Inc.: Vancouver, BC, Canada, 6–9 December 2010; pp. 757–765.
34. Yang, J.; Yuan, X. Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization. *Math. Comp.* **2013**, *82*, 301–329.
35. Gemulla, R.; Nijkamp, E.; Haas, P.J.; Sismanis, Y. Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent. In Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011; ACM: New York City, NY, USA 2011; pp. 69–77.
36. Liu, C.; Yang, H.C.; Fan, J.; He, L.W.; Wang, Y.M. Distributed Nonnegative Matrix Factorization for Web-Scale Dyadic Data Analysis on Mapreduce. In Proceedings of the 19th International Conference on World Wide Web (WWW'10), Raleigh, NC, USA, 26–30 April 2010; ACM: New York, NY, USA, 2010; pp. 681–690.
37. Teffiioudi, C.; Manshadi, F.M.; Gemulla, R. Distributed Matrix Completion. In Proceedings of the IEEE International Conference on Data Mining, Brussels, Belgium, 10–13 December 2012; pp. 655–664.
38. Recht, B.; Ré, C. Parallel stochastic gradient algorithms for large-scale matrix completion. *Math. Prog. Comp.* **2013**, *5*, 201–226.
39. Mardani, M.; Mateos, G.; Giannakis, G. Distributed nuclear norm minimization for matrix completion. In Proceedings of IEEE 13th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Cesme, Turkey, 17–20 June 2012; pp. 354–358.

40. Cabral, R.S.; de la Torre, F.; Costeira, J.P.; Bernardino, A. Matrix Completion for Multi-Label Image Classification. *Advances in Neural Information Processing Systems (NIPS)*, Curran Associates, Inc.: Granada, Spain, 12–17 December 2011; pp. 190–198.
41. Candès, E.; Li, X.; Ma, Y.; Wright, J. Robust principal component analysis? *J. ACM* **2011**, *58*, 11.
42. Cai, J.F.; Candès, E.J.; Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* **2010**, *20*, 1956–1982.
43. Tron, R.; Vidal, R. Distributed Computer Vision Algorithms through Distributed Averaging. In Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CA, USA, 21–25 June 2011; IEEE Computer Society: Washington, DC, USA, 2011; pp. 57–63.
44. Weinland, D.; Boyer, E.; Ronfard, R. Action Recognition from Arbitrary Views using 3D Exemplars. In Proceedings of 2007 IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil, 14–20 October 2007; pp. 1–7.
45. Singh, S.; Velastin, S.A.; Ragheb, H. MuHAVi: A Multicamera Human Action Video Dataset for the Evaluation of Action Recognition Methods. In Proceedings of the 7th IEEE International Conference on Advanced Video and Signal-Based Surveillance, Boston, MA, USA, 29 August–1 September 2010; IEEE Computer Society: Washington DC, USA, 2010; pp. 48–55.
46. Wang, H.; Ullah, M.M.; Kläser, A.; Laptev, I.; Schmid, C. Evaluation of Local Spatio-Temporal Features for Action Recognition. In Proceedings of British Machine Vision Conference, London, UK, 7–10 September 2009; p. 127.
47. Liu, J.; Shah, M. Learning Human Actions via Information Maximization. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08), Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
48. Reddy, K.K.; Liu, J.; Shah, M. Incremental Action Recognition Using Feature-Tree. In Proceedings of IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009; pp. 1010–1017.
49. Wu, X.; Jia, Y. View-Invariant Action Recognition Using Latent Kernelized Structural SVM. In Proceedings of 12th European Conference on Computer Vision—Volume Part V, Firenze, Italy, 7–13 October 2012; pp. 411–424.
50. Shao, L.; Wu, D.; Chen, X. Action Recognition Using Correlogram of Body Poses and Spectral Regression. In Proceedings of the IEEE International Conference on Image Processing, Brussels, Belgium, 11–14 September, 2011; pp. 209–212.
51. Chaaraoui, A.A.; Climent-Prez, P.; Flrez-Revuelta, F. Silhouette-based human action recognition using sequences of key poses. *Pattern Recognit. Lett.* **2013**, in press.