Distributed and Lightweight Multi-Camera Human Activity Classification

Gaurav Srivastava, Hidekazu Iwaki, Johnny Park and Avinash C. Kak
School of Electrical and Computer Engineering
Purdue University, West Lafayette, Indiana 47907-2035
{gsrivast, hiwaki, jpark, kak}@purdue.edu

***Abstract*—We propose a human activity classification algorithm that has a distributed and lightweight implementation appropriate for wireless camera networks. With input from multiple cameras, our algorithm achieves invariance to the orientation of the actor and to the camera viewpoint. We conceptually describe how the algorithm can be implemented on a distributed architecture, obviating the need for centralized processing of the entire multi-camera data. The lightweight implementation is made possible by the very affordable memory and communication bandwidth requirements of the algorithm. Notwithstanding its lightweight nature, the performance of the algorithm is comparable to that of the earlier multi-camera approaches that are based on computationally expensive 3D human model construction, silhouette matching using reprojected 2D views, and so on. Our algorithm is based on multi-view spatio-temporal histogram features obtained directly from acquired images; no background subtraction is required. Results are analyzed for two publicly available multi-camera multi-action datasets. The system's advantages relative to single camera techniques are also discussed.**

I. Introduction

Multiple view action recognition is a logical next step to fixed-view action recognition because it addresses a more realistic scenario: it does not assume that humans are performing the action in a specific direction relative to the camera, e.g. frontal or profile views. In addition, capturing an action from multiple views obtains additional features of the action, thereby potentially increasing the discriminative power of the recognition algorithm. In addition, it provides robustness to partial occlusions and suppresses in-class variations due to personal execution styles of different individuals.

Multi-camera algorithms generally involve some local processing specific to each camera and then the individual results from the cameras are aggregated to give the final classification output. Practical implementation of such algorithms on a distributed camera network places constraints on memory requirements, communication bandwidth, speed etc. which are elaborated in section I-B. In this paper, we present an algorithm that can potentially be implemented on a distributed wireless camera network and provides some level of invariance to the actor orientation and the camera viewpoint.

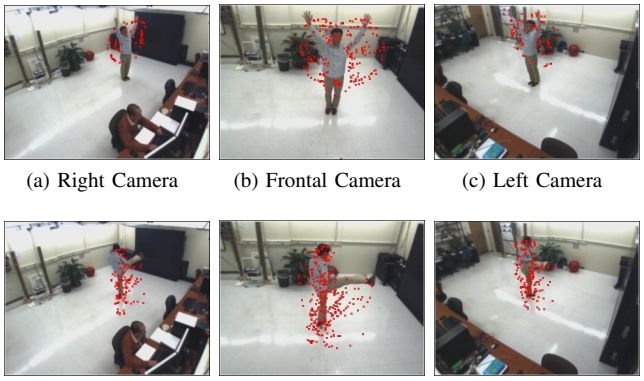
A. Related Work

Several approaches have been proposed to accomplish action recognition using multiple cameras. Weinland et al. [23] propose location and rotation invariant Fourier descriptors in cylindrical coordinates and compared 2 actions based on their 3D visual hull representation. Yan et al. [24] describe arbitrary view action recognition using *4D Action Feature Model* (4D-AFM), which is essentially a temporal sequence of 3D visual hulls on which the spatio-temporal action features are mapped. Actions are recognized by obtaining the best match between action features in the input action video and the features mapped on the 4D-AFMs of different actions. Farhadi and Tabrizi [5] describe an interesting approach to build invariant features between 2 views, for action discrimination. They do not need multiple views to extract their features. Along the same veins, Souvenir and Babbs [19] describe a framework for learning a viewpoint-invariant representation of primitive actions that can be used for video obtained from a single camera. The framework supports learning the variations in the appearance of an action with viewpoint changes and this is accomplished by the use of manifold learning technique to learn a low dimensional representation of action primitives. Syeda-Mahmood et al. [20] model the actions as generalized cylinders called action cylinders. Viewpoint invariance and reliable recognition is achieved by using a set of 8 corresponding points on the time-wise corresponding cross sections of the cylinders to recover the viewpoint transformation between the reference (model) and test action cylinders. Parameswaran and Chellappa [17] model actions in terms of view-invariant canonical body poses and trajectories in 2D invariance space, leading to a simple and effective way to represent and recognize human actions from a general viewpoint. Yilmaz and Shah [25] extend the multi-view epipolar geometry to accommodate for motion of the cameras and establish a correspondence between dynamic scenes that are captured from different viewpoints using the moving cameras.

Another set of approaches is based on describing the temporal action sequences using graphical models. Weinland et al. [22] describe a hidden Markov model where the actions are modeled as sequences of key poses or *exemplars* that are represented as 3D visual hulls. Recognition is performed by projecting the visual hulls to 2D and comparing with images of observed actions using silhouette matching. The approaches in [13] and [14] are similar; they represent actions as a series of multi-view 2D human poses on either a graph model called ActionNet or a conditional random field. The best sequence of actions is inferred using Viterbi search on the graphical model.

B. Distributed Processing Considerations

The approaches summarized above focus on algorithmic aspects of multi-view action recognition; only a few have



(a) Right Camera (b) Frontal Camera (c) Left Camera

(d) Right Camera (e) Frontal Camera (f) Left Camera

Figure 1: Spatio-temporal interest points for multi view images of hand waving and kicking actions. These are represented as red markers overlapped on the images. Time dimension is not shown in the figure. See Section II-B for details.

considered the practical aspects of implementing these algorithms in a real setting, for instance with a distributed camera network [26]. Commonly, it is assumed that all the image data from the multiple cameras can be aggregated to create the training models. During the training and testing phases of the algorithm, entire images have to be transmitted from the cameras to the central processing server, as opposed to sending compact representative descriptors. This translates into a significant communication bandwidth requirement even for commonly used frame rates of 15-30 fps, image resolutions like 320×240 pixels and 5-10 cameras. In addition, operations such as 3D visual hull construction or estimation of rotation and translation of a human model before projecting onto a 2D view are computationally expensive. For the case of a wireless camera network, each node has a limited storage capacity that necessitates a compact representation of the features and the resulting training models of the different action categories, which will be used during the testing stage. Therefore, a considerable overhauling of the above mentioned algorithms would be necessary for a distributed, scalable, robust and real-time implementation—all equally important factors in the practical design of a distributed camera network.

Scalability from the standpoint of implementation on a distributed camera network implies that the system continues to be efficient and practical when there is an increase in the amount of input data or the number of participating cameras. Robustness means that the system does not break down when one or more of the cameras malfunction. Both of these factors suggest a modular design of the system where each camera independently processes its local sensory data, and only moderate amount of data from individual cameras is aggregated for final estimation or prediction.

C. Overview of Our Approach

In view of these considerations, we propose a multi-view algorithm suitable for a distributed camera network. In the later sections of the paper, we briefly describe a conceptual modular architecture for the distributed implementation (Section II-E) and outline the advantages of the proposed algorithm by considering the memory and communication bandwidth requirements (Section IV). The action representation is based on histograms of spatio-temporal features extracted from multiple view video sequences of an action. Spatio-temporal features have been used in many recent algorithms for single-view human action recognition [4], [8], [10], [16]. In these algorithms, the spatio-temporal features are integrated with discriminative learning approaches like SVM [18] or nearest neighbor classification [4]. Generative approaches like pLSA (probabilistic Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation) have also been used for action classification [15], [21], by modeling the action video sequences as distributions over spatio-temporal features.

Our approach bears similarity to the work in [4], in the sense that same spatio-temporal features are used. The distinguishing contributions of the paper are three-fold: 1) we extend the feature histogram representation to multiple cameras and present a novel and simple algorithm for aggregating the individual camera results, 2) we show that the proposed system achieves some level of invariance to the actor orientation and to previously unseen views and 3) we analyze the system's superior storage and communication bandwidth requirements, thus demonstrating its suitability for a distributed camera network implementation. To our knowledge, this is the first work that demonstrates the feasibility of implementing action classification algorithm in a manner that is suitable for wireless cameras. So intentionally we have chosen the action features to be locally computable and the aggregation algorithm is kept simple. We wish to emphasize that the context of wireless (or distributed, in general) camera platforms that have limited resources is quite important to appreciate the contributions of the paper. As we will see in the experimental section, despite the 'simplicity' of the algorithm, the results are comparable to those obtained with heavy-duty 3D algorithms that must be implemented on regular machines.

The rest of the paper is organized as follows: we describe the multi-view histogram representation and recognition algorithm in Section II and experimental results are presented in Section III. The distributed aspects of the paper are discussed in Sections II-E and IV. Finally, Section V summarizes the conclusions and future research.

II. Proposed Methodology

A. Orientation Invariance Using Multiple Cameras

The proposed system achieves invariance to orientation or the viewing direction of the actor. In other words, the actor is not restricted to a fixed orientation while performing actions. Rather she can have one of the multiple orientations and the number of such orientations depend on the number of cameras used in the setup. We use 6 cameras for training and testing in our experiments. Their spatial configuration is as shown in Figure 2. The dimensions of the room are $5.5m \times 5.5m$ and the cameras are wall mounted approximately uniformly around the room at about the same height ($\sim 2.4m$). In such

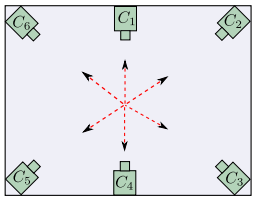
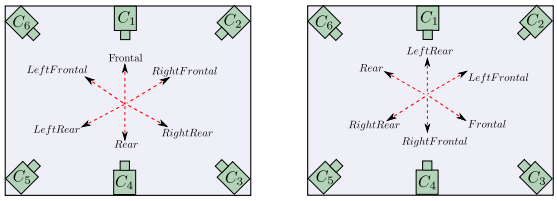


Figure 2: Camera spatial configuration. Subjects may perform actions approximately facing any camera, depicted by red arrows.



(a) Subject facing camera C_1 (b) Subject facing camera C_3

Figure 3: How orientation invariance is achieved using multiple cameras? The cameras (together) capture the same set of features, irrespective of person's viewing direction.

a configuration, the actor may perform actions approximately facing any of the 6 cameras and action classification can still be performed. We also show in our experimental section that even if the actor's orientation is not along any of the 6 camera views but in between them, high classification performance can still be achieved. This robustness to novel camera views (that are not part of the training stage) becomes possible due to accumulation of classification evidence from multiple cameras.

The camera positions satisfy the assumption that we make in order to accomplish orientation invariance. The assumption is that since our algorithm uses spatio-temporal features extracted from the multi-camera image sequence, *we can expect the set of features taken in their entirety from the 6 cameras to be substantially independent of the subject's viewing direction while performing any action*. From the subject's viewpoint, she should see the same spatial configuration of cameras irrespective of the direction she is facing. This is illustrated in Figure 3 (a) and (b). When the subject is facing camera C_1, the cameras C_6, C_1, C_2, C_5, C_4 and C_3 capture the left-frontal, frontal, right-frontal, left-rear, rear and right-rear features respectively. When facing C_3, cameras C_2, C_3, C_4, C_1, C_6 and C_5 respectively capture the same set of features.

Under the above assumption, we implement the action recognition algorithm as follows: During the training phase, subjects perform a set of actions facing a particular camera, say C_1. Their multi-view spatio-temporal feature histograms are stored as training data. During the testing phase, *the test subject is free to face any camera while performing the actions*. If, for instance, she faces C_3, then the features captured with C_3 will be similar to features captured with C_1 from the training data. Similarly, test features from C_2 would be similar to training features from C_6 and so on. Therefore, irrespective of the test subject's orientation, a one-to-one correspondence can be established between the cameras during the testing phase and the cameras during the training phase. With such a correspondence, we can readily determine the training action sequence whose feature histograms match best with the test sequence's feature histograms. During the testing stage, this correspondence is found automatically by the multi-view algorithm which is presented in Sections II-C and II-D.

Our current setup with the uniform positioning of the cameras around the room and their approximately uniform heights above the ground plane, ensures a circularly symmetric array of cameras that satisfies the orientation invariance assumption stated above. Such a setup, admittedly, can give only *some* degree of orientation and camera view invariance. In order to perform activity classification for arbitrary orientation and viewpoint, one may use more sophisticated algorithms, for example, building a full 3-d human model that can be reprojected to arbitrary camera views or by developing viewpoint-invariant features along the lines of [5], [19]. However, in the current work, our focus is on the distributed and lightweight aspects of an action classification algorithm, that offer a very affordable performance for factors like memory and communication bandwidth requirements.

B. Feature Extraction and Representation

Our approach is based on representing the image sequences as histograms over cuboid prototypes. Cuboid prototypes are spatio-temporal patterns that characterize different types of actions. These are obtained by extracting space-time interest points from a image sequence, as described in [4]; we present the method here briefly. The image sequence is convolved with a separable spatio-temporal linear filter to give a response function $R = (I \star g \star h_{ev})^2 + (I \star g \star h_{od})^2$ and the local maxima of R are the space-time interest points. Here I is the image sequence, $g(x, y; \sigma)$ is the 2D spatial smoothing Gaussian kernel and h_{ev}, h_{od} are 1D temporal Gabor filters defined as: $h_{ev} = -cos(2\pi t\omega)\, exp\left(-(t/\tau)^2\right)$ and $h_{od} = -sin(2\pi t\omega)\, exp\left(-(t/\tau)^2\right)$. Cuboids are the spatio-temporal volumes extracted around the space-time interest points and their sizes are determined by the spatial scale σ and temporal scale τ. As noted by Dollar et al. [4], cuboids correspond to regions in the image sequence where complex periodic or non-periodic motion takes place which results in strong response in the function R. For example, for a person performing hand waving action, the cuboid density will be higher near hand region while for kicking action, the density would be higher near the leg region (See Figure 1).

To obtain a compact and robust feature representation, the spatio-temporal gradients are calculated at all the pixels of a cuboid and concatenated to form a feature vector. PCA is applied to the feature vectors to retain the first few principal components. In our experiments, $\backsim$ 2875-length feature vectors are reduced to 25 dimensions. The dimensionality reduced feature vectors are aggregated from all the image sequences of different subjects and different actions in the training dataset and clustered using k-means clustering al-

gorithm. The resultant cluster centers are termed as cuboid prototypes. Each feature vector from an image sequence can be uniquely assigned to one of the clusters centered at the cuboid prototypes and therefore the image sequence can be viewed as a histogram over the cuboid prototypes.

C. Multiple View Action Descriptors

The approaches described in [4], [12], [15] have used the cuboid features (Section II-B) for single camera action recognition. We extend this representation for multi camera scenario. In our setup, $N\,(=6)$ cameras capture the action sequence from multiple views. The cuboid features are extracted on each view separately, thus creating N histograms which characterize each action sequence. This is different from single camera scenario where each action sequence is characterized by one histogram.

In the single camera action classification framework, training data is collected from multiple subjects, each performing different basic actions like walking, running, waving hands and so on. This data is used to generate multiple instances of histograms for each action category. If there are C action categories and M training subjects, then the training set can be represented as $\{H_{i_m}\}$, $i = 1,\ldots,C$, $m = 1,\ldots,M$. Here, H_{i_m} represents the histogram of i^{th} action sequence performed by m^{th} subject. The classification can be performed using k-NN (k-Nearest Neighbor) technique. Specifically, if we use 1-NN, then a given test action sequence can be assigned the action class of the closest training sequence where closeness is defined in terms of some distance measure between their histograms. In our experiments, we use the χ^2 distance measure. So for a test sequence with histogram H, the predicted action label is given by $\hat{i} = \underset{i,m}{arg\,min}\left\{d_{\chi^2}\left(H, H_{i_m}\right)\right\}$.

In order to formulate the multi camera action classification problem, assume that both the training and test action sequences are captured with N cameras. Based on the view invariant discussion in Section II-A, we can conclude that for the *same action category*, the set of N histograms of a test action sequence will be similar to the histograms of a training action sequence, for some particular correspondence between the cameras during the test stage and the training stage. This correspondence can be determined by circularly shifting the set of N test histograms and finding the circular shift that results in minimum distance between the test histograms and the N histograms of one of the training action sequences. The action label of the training action sequence that minimizes the distance between the two sets of N histograms, is assigned to the test sequence.

More formally, let $A_1 = \left\{H_1^1, H_1^2, \ldots, H_1^N\right\}$ and $A_2 = \left\{H_2^1, H_2^2, \ldots, H_2^N\right\}$ be the multi-view histogram representations of two examples of the same action, not necessarily performed in the same orientation. Here N is the number of camera views and superscripts indicate camera indices. Let the set $\{\pi_j(1), \pi_j(2), \ldots, \pi_j(N)\}$ represent a circularly shifted version of the set $\{1, 2, \ldots, N\}$, where the subscript j refers to a particular circular shift. The subscript j is needed because N circular shifts are possible, e.g., for $N = 3$, the circularly shifted versions will be $\{1,2,3\}, \{3,1,2\}, \{2,3,1\}$. Then the distance between the two histogram representations can be defined as

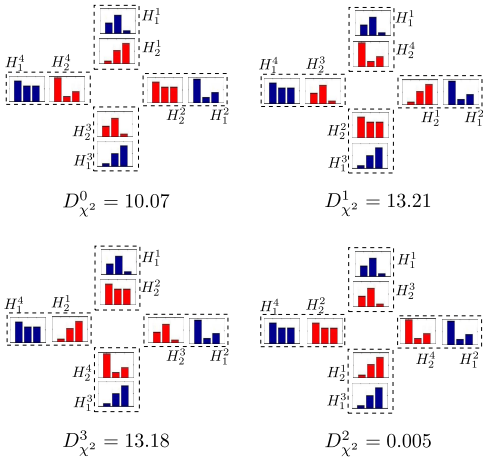


Figure 4: An example of computing the distance between multi-view histograms of two actions. It is assumed that the number of camera views $N = 4$. Blue and red histograms represent respectively actions A_1 and A_2. The distances are computed for different circular shifts of the A_2 histograms, depicted in the 4 sub-figures. See Section II-C for details.

$$D_{\chi^2}\left(A_1, A_2\right) = \min_j \left\{D^j_{\chi^2}\left(A_1, A_2\right)\right\} \quad (1)$$

$$where \quad D^j_{\chi^2}\left(A_1, A_2\right) = \sum_{n=1}^{N} d_{\chi^2}\left(H_1^n, H_2^{\pi_j(n)}\right) \quad (2)$$

This procedure for finding the distance between two multi-view histogram representations is depicted pictorially through an example in Figure 4. The example assumes $N = 4$ cameras. The sub-figures show the computation of $D^j_{\chi^2}\left(A_1, A_2\right)$ for $j = 0, 1, 2, 3$. We can infer from the figure that $D_{\chi^2}\left(A_1, A_2\right) = 0.005$.

D. Multiple View Action Classification

Using the above distance formulation, we can state the multi-view action classification problem as follows: let a training action sequence be represented as $A_{i_m} = \left\{H^1_{i_m}, H^2_{i_m}, \ldots, H^N_{i_m}\right\}$ for $i = 1,\ldots,C$, $m = 1,\ldots,M$. Here C is the number of action categories and M is the number of training subjects. The set of all training action sequences will be $\{A_{i_m}\}$. Then, given a test action sequence A, the predicted class label is given by $\hat{i} = \underset{i,m}{arg\,min}\left\{D_{\chi^2}\left(A_{i_m}, A\right)\right\}$.

In the prediction rule above, we assumed that training and test sequences are acquired using the same number of cameras. The prediction rule can also be adapted easily to the scenario where the test cameras are a subset of training cameras. Rather than giving a general derivation, for notational simplicity, we take an example scenario where only cameras C_1, C_2 and C_5 (refer to Figure 2) are present during test sequence capture. In this case, the test sequence will be represented as $A = \left\{H^1, H^2, \mathbf{0}, \mathbf{0}, H^5, \mathbf{0}\right\}$. In this representation, we need to

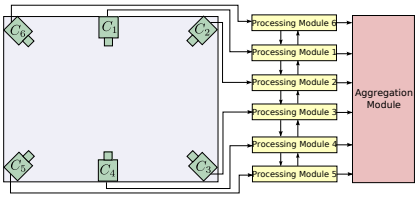


Figure 5: Conceptual distributed processing architecture for implementing the multi-view action classification algorithm. See section II-E for details.

know the missing camera ids in order to maintain the proper ordering of cameras. The $\mathbf{0}$ entries are symbolic histogram representations for missing cameras. The distance formulation of Eq. (1) can be used directly here, with the modification that the distance between a $\mathbf{0}$ histogram and another "normal" histogram is set to 0. A typical instance of this scenario is when only 1 test camera is available. This case has been evaluated in our experiments.

E. Suitability for Distributed Implementation

Our action classification algorithm is based on comparing the set of N test histograms with different sets of N training histograms. Each camera view has its associated training data that comprises of single-view histograms of multiple subjects performing different actions. For M subjects and C action categories, the training data has $M \times C$ histograms per camera. According to Eq. (2), the N cameras can compute the terms $d_{\chi^2}\left(H_1^n, H_2^{\pi_j(n)}\right)$ in parallel for any circular shift j and for a given training and test action sequence. Subsequently, these terms can be aggregated to give $D^j_{\chi^2}(A_1, A_2)$. This feature of the algorithm can be exploited to realize a distributed implementation where each camera can process the test action's data independently and can transmit its final results for aggregation with those from the other cameras. In order to compute the minimum distance according to Eq. (1), all N circular shifts have to be considered. This implies that a camera must compute the distances between each of the N test histograms and its own training histograms, thereby generating $M \times C \times N$ distance values. In order to emulate the circular shifts of the test histograms between the cameras, each camera could transmit its own test histogram to the rest of the cameras. Please see Section IV-B for an analysis of the bandwidth requirements for transmitting these data values.

Figure 5 shows the conceptual schematic of a distributed processing architecture on which the action classification algorithm can be implemented. The current generation wireless cameras like WiCa [9] and Imote2 [3] are some examples of the camera platforms on which the algorithm can potentially be implemented. The cameras $C_1 - C_6$ act as image acquisition modules, each of which is associated with a processing module that executes the steps of the algorithm that are relevant to individual cameras. The processing modules can communicate with each other to exchange the test action's histograms and can transmit the final distance values to the aggregation module. It is emphasized that entire images need not be transmitted from the processing modules to the aggregation module, as is the case with other recognition algorithms that rely on fully centralized processing of multi-view image sequences.

III. Experiments

We conducted experiments on two publicly available multi-view multi-action datasets: 1) recently acquired Purdue dataset [1], and 2) the IXMAS dataset [2].

A. Purdue Dataset

We use 6 cameras (Dragonfly2, Point Grey Research Inc.) to capture images at 30 fps with a resolution of 320×240 pixels. 12 subjects perform a set of 9 actions: walking, running, bending, kicking, sitting down, standing up, hand waving, hand clapping and boxing. Each action is synchronously captured by the cameras for 6 seconds. Such single-person action video sequences have been used as benchmarks in many previous papers in action classification e.g. [6], [18], [22]. For the purpose of experiments, the subjects perform the actions in one of the 3 orientations O_1, O_2 or O_3, that correspond to facing cameras 1, 3 or 5 respectively. *But as noted in Section II-A, the action recognition algorithm is not limited to these 3 orientations.* During the training phase, all the 12 subjects perform the actions in orientation O_1. In the testing stage, 6 of the 12 subjects repeat the actions in orientation O_2 while the other 6 in orientation O_3.

A subject's image in different camera views may have different heights due to unequal distances of the subject from the cameras. In order to have scale invariance, different camera images need to be normalized so that a subject has approximately the same height in all views (as shown in Figure 6). This normalization can be accomplished by using any tracking algorithm to determine the current position of the subject on the ground plane and combining it with camera-to-groundplane homography information to estimate the subject's heights in different viewpoints. Similar ideas have been reported in [7], [11]. In our experiments, for simplicity, we bypass the tracking step and ask the subjects to perform the actions at a fixed prespecified position on the groundplane. We scale the images from different camera views roughly in proportion to the camera distances to the fixed position. Our algorithm is not influenced by the different sized images in different views.

For spatio-temporal feature extraction, we set spatial scale $\sigma = 2$, the temporal scale $\tau = 2.5$ and the size of cuboids as $13 \times 13 \times 17$ pixels, which corresponds to ~ 2875 pixels. The number of cuboid prototypes is set to 50 and therefore, each action histogram has 50 bins.

We employed the leave-one-out strategy to evaluate the performance of the algorithm. During each run, 11 subjects' O_1 action sequences were used to construct the training data and action classification was performed on the sequences of the remaining subject, captured either in O_2 or O_3 orientation. Final classification accuracy was obtained by averaging over all the runs where each subject was chosen as the test subject exactly once. 3 experimental scenarios were considered: (1) Multi-view training / multi-view testing (MM)—all 6 camera views were used during both the training and testing phases;

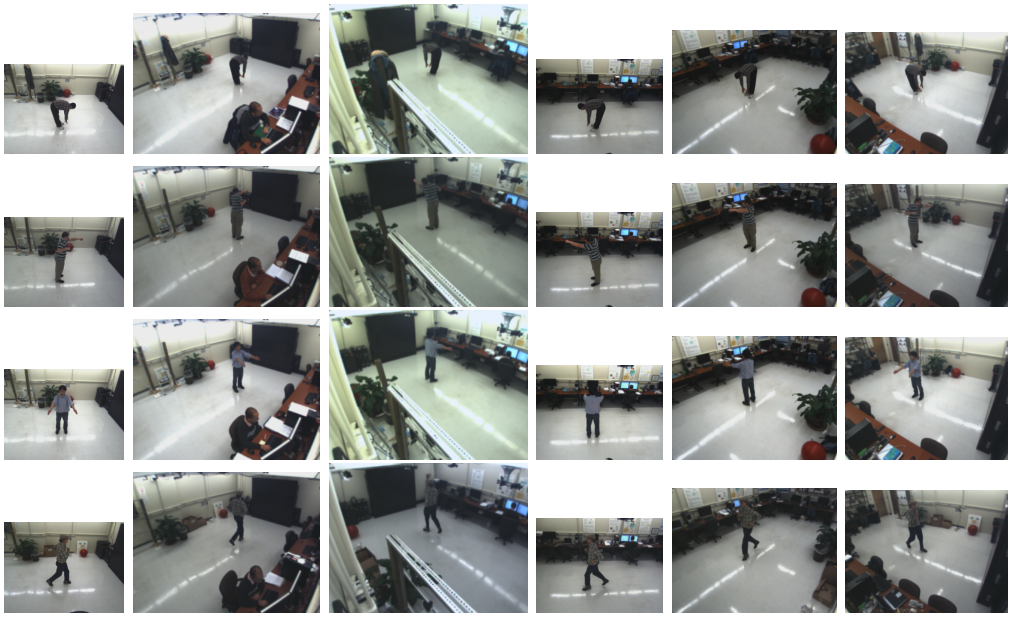


Figure 6: Few examples of multi-camera action sequences from Purdue dataset: bending, boxing, handclapping and running. Images from each camera view are scaled differently, so that the subjects have roughly the same height in every view.

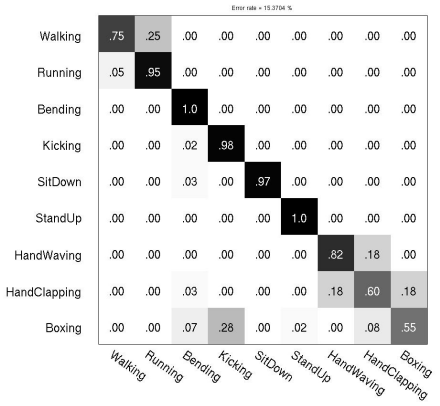


Figure 7: Confusion matrix for the multi-view training/multi-view testing on Purdue dataset. Vertical and horizontal axes represent the true and predicted action labels respectively.

(2) Multi-view training / single-view testing (MS)—all 6 camera views were used for training, but only one view for testing; and (3) Single-view training / single-view testing (SS)—one view used for training, one view for testing. In the 2^{nd} and 3^{rd} scenarios, single-view testing is performed using either the frontal view sequences (camera C_3 in orientation O_2, C_5 in O_3), left-frontal view (camera C_2 in O_2, C_4 in O_3) or right-frontal view (camera C_4 in O_2, C_6 in O_3).

The confusion matrix for scenario 1 (MM) is shown in Figure 7 and Table I summarizes the classification rates for all the scenarios considered above. Several observations can be made from the results. The confusion matrix shows high classification rates for most actions, except boxing and hand-clapping. These classification rates demonstrate the superior discriminative ability of our proposed algorithm even amidst the severe background clutter in action sequences (see Figure 1). Boxing and handclapping actions are classified with lower accuracy, possibly because of the local spatial extent of the actions, due to which most camera views could not capture sufficient discriminative features. From Table I, we can observe that the classification accuracy of the MM scenario is better than the MS, which in turn is better than SS. This indicates that multi-view approach improves the classification accuracy and provides a greater degree of orientation invariance.

	Multi View Testing	Single View Testing		
		Left	Front	Right
Multi View Training	84.6	73.18	82.96	64.82
Single View Training	N/A	56.48	78.89	45.37

Table I: Action classification rates on Purdue dataset

In the MS and SS scenarios, frontal view test sequences result in better performance as compared to left-frontal or right -frontal views, indicating that the classification performance degrades significantly when there is even a slight misalignment between the training and testing videos. This is an important observation for comparing the proposed algorithm with single-camera algorithms like the ones described by Dollar et al. [4] and Niebles et al. [15]. They report classification accuracy of 78.2% and 83.3% respectively. Their results are reported for training and testing both on frontal views. Even a small change between the training and testing views would degrade the performance significantly, as we observed in our experiments. But our multi-camera setup would give high classification accuracy because the system makes the decision not based on just one camera's captured features but on the full feature set captured from all the cameras in the setup.

The result reported in Table I for MM scenario are for the case when same set of cameras are used during the training

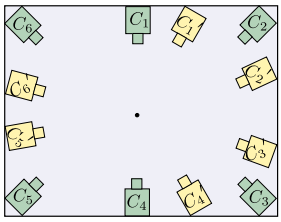


Figure 8: Action classification from previously unseen views. Green cameras represent training set views, yellow cameras represent testing set views.

# Previously Unseen Views	Classification Accuracy
1	83.70%
2	82.78%
3	82.22%
4	83.52%
5	78.70%
6	76.30%

Table II: Classification performance as a function of number of previously unseen views during testing stage

and testing stage. We performed another experiment to demonstrate that the camera views need not be the same between the two stages. Even if some previously unseen camera views are used during the testing stage, the system continues to give high classification performance. Figure 8 shows the spatial configuration of cameras $C_1', \ldots, C_6'$ (yellow-colored cameras) that are used during the testing stage and are different from training set of cameras (green-colored cameras). Results in Table II show the classification performance when some of the cameras C_i are replaced with new set of cameras C_i' during the test stage. It is quite significant that even when all the training set cameras are replaced with new camera views, the system still gives a reasonably good accuracy of 76.3%.

B. IXMAS Dataset

This dataset comprises of 13 action categories, performed 3 times by each of the 12 subjects. The subjects perform the actions in arbitrary positions and orientations. This dataset has been used by several groups researching on view-invariant action recognition [13], [22], [24]. In order to be consistent with the experimental methodology of [22], we only use the data of 10 subjects and 11 action categories. Further, we exclude camera 5 from our analysis because it captures the overhead view of the subjects and does not generate discriminative features for the classification task. At the same time, it violates the setup of the circular array of cameras located at approximately the same height, which is an assumption we make for orientation invariance. Multi-view classification is carried out using varying number of cameras and results are compared with those reported in [22] and [24]. Note that, our approach is applied directly on the image sequence and does not require additional information such as background subtracted silhouettes or 3D visual hull volumes, which are needed for the other two compared approaches. Table III summarizes the average classification rates.

From the results, it is quite evident and quite significant that our simple and lightweight algorithm performs equally well or better compared to the other two algorithms that are based on full 3D human model reconstruction. Also if we observe the classification accuracy of the proposed approach, this dataset clearly brings out the fact that more cameras used during the testing stage lead to superior performance in action classification.

# Cameras in testing stage	Our approach	Weinland et al.	Yan et al.
4	81.4%	81.3%	78%
3	79.1%	70.2%	60%
2	75.6%	81.3%	71%
1	69.1%	-	-

Table III: Action classification comparison for 3 algorithms on IXMAS dataset.

IV. Advantages for Distributed Implementation

In this section, we highlight the advantages of our proposed approach for implementation of the multi-view action classification algorithm on a distributed camera network. We present a comparative analysis with [22] and base the numerical arguments on the IXMAS dataset. For any distributed implementation on a camera network, it is necessary to pay special attention to memory requirements of each camera and the communication bandwidth requirements of the network. These two issues are considered in the following discussion:

A. Memory Requirements

We consider the amount of memory required to store the model constructed during the learning phase, as it would be utilized repeatedly during the testing phase for classifying the test action's image frames. In [22], the model is a set of 52 exemplars, each of which is a 3D visual hull with $64 \times 64 \times 64$ voxels. Since each voxel value is stored using 1 bit, the total amount of required memory is $52 \times 64^3 \times \left(\frac{1}{8} \times 10^{-6} Mbytes\right) \backsimeq 1.72 Mbytes$. In our approach, the model is simply the set of multi-view histograms stored per camera, for all the training subjects and for all the actions. Assuming that we have 10 training subjects, 11 actions and that each action histogram has 50 bins, the amount of memory required for histogram storage per camera is $10 \times 11 \times 50 \times \left(4 \times 10^{-6} Mbytes\right) = 0.022 Mbytes$. Additionally, for spatio-temporal feature extraction, we need to store the codebook of 50 cluster prototypes each of length 25 and a PCA reprojection matrix of size 25×2875 where each cuboid has 2875 pixels. This requires an additional memory of $(50 \times 25 + 25 \times 2875) \times \left(4 \times 10^{-6} Mbytes\right) = 0.293 Mbytes$ per camera. Therefore the total required memory will be $0.315 Mbytes$ per camera.

B. Communication Bandwidth Requirements

We assume that for both the algorithms, multi-view observations are used during the recognition. During the recognition phase of [22], the image frame rate is 23 fps. The image size is 390×291 pixels but only the silhouette information is used which is restricted to 64×64 ROI. Each pixel in the ROI can be represented using 1 bit, so the required bandwidth is $23 \times 64 \times 64 \times \left(\frac{1}{8} \times 10^{-3} Kbytes\right) = 11.77\, Kbytes/s/camera$. For 4 cameras, the required bandwidth is $\sim 47\, Kbytes/s$. In our

proposed approach, each camera must know the multi-view histogram representation of the action sequence. This implies that every camera needs to broadcast it's test action histogram representation to other cameras. An action histogram has 50 bins and therefore requires $50 \times 4 = 200\,bytes$. If we assume that the duration of every action is 3 seconds and so the histograms are generated and broadcasted every 3 seconds, the required bandwidth will be $4 \times 200\,bytes \times \frac{1}{3\ seconds} \backsimeq 0.3Kbytes/s$. Every camera generates $M \times C \times N$ distance values and sends them to the aggregation module. Here, M is the number of training subjects, C is the number of action categories and N is the number of cameras. Therefore, for all the cameras combined, $M \times C \times N^2$ distance values are transmitted for aggregation every 3 seconds. Using $M = 10, C = 11, N = 4$, the bandwidth required during data aggregation is $\sim 2.4Kbytes/s$. So the overall requirement is $\sim 2.7Kbytes/s$ which is a significant bandwidth saving primarily because the cameras need not send the entire frames to any central computing server.

V. Conclusions and Future Research

In this paper, we have described a distributed and lightweight action classification algorithm based on a compact multi-view feature representation . We have shown that for the multi-camera action classification task, our algorithm effectively and efficiently combines the information from 2D views and achieves results at par with other algorithms based on full 3D human model construction. In addition, our approach has the advantage of simple implementation. It has substantially less storage and lower communication bandwidth requirements compared to the 3D-based algorithms. This makes the proposed approach suitable for distributed camera networks where the cameras can individually process their local sensory data and their evidence combined to obtain the final classification result. In the future, we would like to use other feature representations that relax the assumptions we make in the current paper, regarding the cameras' uniform spatial positioning. It is desirable to have feature representations that are transferable between arbitrary camera views so that full flexibility of arbitrary view action recognition systems can be accomplished, while at the same time maintaining the advantage of low resource usage and distributed implementation. For automatic scale invariance, the camera calibration and the camera-groundplane homography information can be used together with subject's current position from a tracking module, to normalize the image sizes.

VI. Acknowledgments

The authors would like to thank Piotr Dollar for providing the Image and Video toolbox for spatiotemporal feature computation. This work was supported by the Sensor Networks Research initiative at Olympus Corporation, Japan.

References

[1] http://cobweb.ecn.purdue.edu/rvl/research/humanactivityclassification/index.html.

[2] https://charibdis.inrialpes.fr/html/non-secure-sequence.php?s=ixmas.

[3] http://www.xbow.com/products/productdetails.aspx?sid=253.

[4] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. IEEE International Workshop on*, pages 65–72.

[5] A. Farhadi and M. K. Tabrizi. Learning to recognize activities from the wrong view point. *European Conference on Computer Vision*, pages 154–166, 2008.

[6] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[7] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2:2137–2144, 2006.

[8] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. *Computer Vision, IEEE International Conference on*, 1:166–173, 2005.

[9] R. Kleihorst, B. Schueler, A. Danilin, and M. Heijligers. Smart Camera Mote With High Performance Vision System. In *Proceedings of the International Workshop on Distributed Smart Cameras (DSC-06)*, Oct 2006.

[10] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, September September 2005.

[11] Z. Lin, L. Davis, D. Doermann, and D. DeMenthon. Hierarchical part-template matching for human detection and segmentation. *Computer Vision, 2007. IEEE 11th International Conference on*, pages 1–8, Oct. 2007.

[12] J. Liu, S. Ali, and M. Shah. Recognizing human actions using multiple features. *Computer Vision and Pattern Recognition, 2008. IEEE Conference on*, pages 1–8, June 2008.

[13] F. Lv and R. Nevatia. Single view human action recognition using key pose matching and viterbi path searching. *Computer Vision and Pattern Recognition, 2007. IEEE Conference on*, pages 1–8, June 2007.

[14] P. Natarajan and R. Nevatia. View and scale invariant action recognition using multiview shape-flow models. *Computer Vision and Pattern Recognition, 2008. IEEE Conference on*, pages 1–8, June 2008.

[15] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, Sept. 2008.

[16] A. Oikonomopoulos, I. Patras, and M. Pantic. Spatiotemporal salient points for visual recognition of human actions. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 36(3):710–719, June 2005.

[17] V. Parameswaran and R. Chellappa. View invariance for human action recognition. *International Journal of Computer Vision*, 66(1):83–101, January 2006.

[18] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. *Pattern Recognition, 2004. Proceedings of the 17th International Conference on*, 3:32–36, Aug. 2004.

[19] R. Souvenir and J. Babbs. Learning the viewpoint manifold for action recognition. *Computer Vision and Pattern Recognition, 2008. IEEE Conference on*, pages 1–7, June 2008.

[20] T. Syeda-Mahmood, A. Vasilescu, and S. Sethi. Recognizing action events from multiple viewpoints. *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, pages 64–72, 2001.

[21] Y. Wang, P. Sabzmeydani, and G. Mori. Semi-latent dirichlet allocation: A hierarchical model for human action recognition. *Lecture Notes in Computer Science*, 4814/2007:240–254, 2007.

[22] D. Weinland, E. Boyer, and R. Ronfard. Action recognition from arbitrary views using 3d exemplars. *Computer Vision, 2007. IEEE 11th International Conference on*, pages 1–7, Oct. 2007.

[23] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2-3):249–257, 2006.

[24] P. Yan, S. Khan, and M. Shah. Learning 4d action feature models for arbitrary view action recognition. *Computer Vision and Pattern Recognition, 2008. IEEE Conference on*, pages 1–7, June 2008.

[25] A. Yilmaz and M. Shah. Recognizing human actions in videos acquired by uncalibrated moving cameras. *Computer Vision, 2005. Tenth IEEE International Conference on*, 1:150–157 Vol. 1, Oct. 2005.

[26] Z. Zivkovic, V. Kliger, R. Kleihorst, A. Danilin, B. Schueler, G. Arturi, C.-C. Chang, and H. Aghajan. Toward low latency gesture control using smart camera network. *Computer Vision and Pattern Recognition Workshop*, 0:1–8, 2008.