Single View Human Action Recognition using Key Pose Matching and Viterbi Path Searching

Fengjun Lv   Ramakant Nevatia

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0273
{flv|nevatia}@usc.edu

Abstract

3D human pose recovery is considered as a fundamental step in view-invariant human action recognition. However, inferring 3D poses from a single view usually is slow due to the large number of parameters that need to be estimated and recovered poses are often ambiguous due to the perspective projection. We present an approach that does not explicitly infer 3D pose at each frame. Instead, from existing action models we search for a series of actions that best match the input sequence. In our approach, each action is modeled as a series of synthetic 2D human poses rendered from a wide range of viewpoints. The constraints on transition of the synthetic poses is represented by a graph model called Action Net. Given the input, silhouette matching between the input frames and the key poses is performed first using an enhanced Pyramid Match Kernel algorithm. The best matched sequence of actions is then tracked using the Viterbi algorithm. We demonstrate this approach on a challenging video sets consisting of 15 complex action classes.

1. Introduction

Recognizing basic human actions (*e.g.* walking, sitting down and waving hands) from a monocular view is an important task for many applications such as video surveillance, human computer interaction and video content retrieval. Recently, many research efforts have focused on recovering human poses [1, 6, 12], which is considered as a necessary step for view-invariant human action recognition. However, 3D pose reconstruction from a single viewpoint is a well known difficult problem in itself because of the large number of parameters that need to be estimated and the ambiguity caused by perspective projection.

Alternatively, example based methods [16, 17] store a database of example human figures with known 3D parameters and estimate 3D pose by searching for examples similar to the input image. Comparing with known examples is certainly easier than inferring unknown parameters but good coverage in a high dimensional parameter space needs a large number of examples. The difficulty in getting enough examples makes the pose recovered not highly accurate.

Rough estimates of poses may, however, still be sufficient to infer human actions by taking advantage of the contextual constraints imposed by actions. Such constraints are three-fold: First, the occurrence of poses within an action should follow some specific order; in "walking" for example, two-leg crossing should occur between left-leg stepping and right-leg stepping. Second, the transitions between different actions should not be arbitrary; for example, "sitting" can not become "walking" without "standing-up" in between. Third, change in actor's orientation should be smooth. By taking advantage of these contextual constraints, we can expect to eliminate many short-term errors caused by image noises and perspective ambiguity.

We present an example based action recognition system that explores the use of such constraints. These constraints are inherently modeled using a novel action representation scheme called *Action Net*. Action Net is a graph model. Each node in the Action Net contains the 2D representation of one view of an example 3D pose called a *key pose*. Each link specifies the possible transition of key poses within an action class or across different action classes.

In the learning phase, key poses of each action class are extracted from a small set of motion capture sequences. The key poses capture the essence of each action class even if there is variance in execution styles of the same action. The Action Net is automatically constructed by connecting actions with similar boundary key poses. The key poses are rendered from a variety of viewpoints using POSER, a human character animation software, to generate synthetic realistic looking human figures. The *Shape Context* [2] of the human silhouette is computed as the 2D representation of a human figure and stored in each node of the Action Net.

During recognition, the human silhouette in each input frame is extracted based on background subtraction. The shape context of the human silhouette is matched with all nodes in the Action Net using a modified *Pyramid Match Kernel* (*PMK*) algorithm. The speed of PMK allows us to cover a large number of viewpoints in near real time. We show that in the original PMK algorithm [7], two sets of high dimensional features tend to get a low matching score regardless of their similarity. We propose a fix to this problem by using a different feature space partitioning scheme. Finally, the action recognition (including segmentation) problem is formulated as finding the most likely sequence of nodes within the Action Net, which is achieved by applying the Viterbi algorithm [15].

1.1. Related Work

Human action recognition and pose recovery have been studied extensively in recent years. We give a brief overview here and refer the reader to [13] for a more comprehensive survey.

Many 2D approaches to action recognition [3, 4, 9, 11, 21] have been proposed. These approaches can be roughly grouped as space-time shape based [3, 21], interest point based [9, 11] and motion template based [4] and they work effectively under the assumption that the viewpoint is relatively fixed (usually from frontal or lateral view), possibly with small variance. The lack of a view-invariant action representation limits the applications of such 2D based approaches. To overcome this limitation, some approaches such as [4] resort to using multiple cameras.

A truly view-invariant approach needs the knowledge of 3D human poses, which can be robustly recovered from multiple views [10, 20]. A more challenging problem is recovering 3D poses from a single view. One group of methods learns a direct mapping from the image feature space (*e.g.* silhouette) to the parameter space (*e.g.* 3-D pose) using techniques such as regression [1] or manifold embedding [6]. However, such mapping is usually multi-valued and it is difficult for direct mapping to maintain multiple hypotheses over time. There are also approaches [12] that directly explore through the parameter space and search for an optimal solution. Due to the high dimensionality, such approaches usually use sampling based techniques and look for image evidences to guide the sampler, but the computational complexity is still very high.

Many approaches to the pose tracking or the action recognition task use graph models such as *Hidden Markov Models* [16] and *Conditional Random Fields* [18] to exploit temporal constraints. The major difference between the Action Net and these graph models is that information such as camera viewpoint and action connectivity is explicitly modeled in the Action Net, while these graph models use parameters to encode such information.

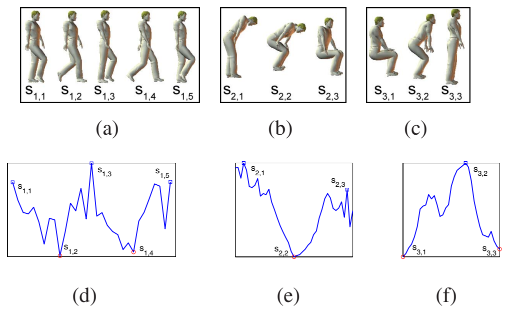


Figure 1. Automatically extracted key poses and the motion energy chart of three action sequences. Marked dots in the bottom row are the positions of the key poses.

2. Action Representation

Human action is characterized by a spatial element, which is the body pose at each time step, and a temporal element which is the evolution of the body poses over time. Instead of including body poses in all frames, we can have a more compact representation of a human action. For example, the following poses suffice to describe a walking action: $\{$two legs crossing $\rightarrow$ right leg forward $\rightarrow$ two legs crossing $\rightarrow$ left leg forward $\rightarrow$ two legs crossing$\}$, as shown in Fig.1(a). In Fig.1(b) and 1(c), a "sitting down" and a "standing up" action are clearly observed with only three poses each. We term such poses as key poses and an action class is modeled as a sequence of key poses.

This representation has two major advantages: First, because actions are recognized by comparison with known action models, a small number of key poses reduces the computational complexity; second, by focusing on the key poses only, we can capture the essence of an action class even if there is variance in execution styles of the same action.

2.1. Automatic Extraction of 3D Key Poses

We learn key poses using 3D motion capture data. First, we manually select a small set of motion capture sequences for each action class. Significantly different styles for the same action type (*e.g.* "sitting on the ground" *vs* "sitting on a chair") are treated as different action classes. Sequences of the same action class are aligned using the *Dynamic Time Warping* [14] algorithm, which is widely used to find an optimal alignment between variable length sequences.

Consider the averaged sequence for one action class, which contains a series of 3D body joint positions $\{P_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})\}$, where i and j are the frame index and the joint index, respectively. The motion energy at the i-th frame is defined as:

$$E_i = \sum_{j=1}^{n} |P_{i,j} - P_{i-1,j}|^2 \tag{1}$$

where $|.|$ denotes Euclidean distance and n is the number of body joints.

Key poses are defined as the poses with maximum or minimum motion energy within a sliding window, *i.e.* E_i equals $\max_{i'=i-L}^{i+L} E_{i'}$ or $\min_{i'=i-L}^{i+L} E_{i'}$, where L is the half length of the sliding window centered at the current frame i. The 3D key poses shown in Fig.1 (from the lateral view) are actual results obtained this way. The change of motion energy in each sequence is shown at the bottom and the positions of these key poses are marked. We choose $L = 15$ frames in our system which results in an average of about four key poses for each action class.

2.2. Generation of an Action Net

An action class is modeled as a chain of the extracted key poses, as shown in Fig.2(a). Each node (state) in this graph model corresponds to one key pose. Like many other graph models such as HMMs, each node has an underlying observation node (not shown in the figure).

We also consider other types of connectivity. A *back-link* connects backward two states in the same model, as shown in Fig.2(b). Back-links are useful to model periodic actions such as walking and actions with unknown number of repeated motions such as waving hands. An *inter-link* links one action to another, as shown in Fig.2(c). Inter-links are used to rule out unlikely transitions between different action models. Both back-links and inter-links are determined based on the similarity between two 3D key poses, *i.e.* pose i can connect to pose j *iff* $|pose_i - pose_j| \leq \delta$.

By connecting different action models, we build a more complex graph model called an *Action Net*. (See an example in Fig.2(d).) Action Net provides long-term contextual constraints for action recognition and segmentation in that the links within an action model specify the order of the key poses and the links across action models constrain the possible action transitions.

2.3. Pose Representation in 2D

Key poses are defined in 3D. However, because the input of the system is a video, we need to either infer 3D poses from the video, which is usually hard, or alternatively, store all 2D views for each 3D key pose. Although it is impossible to cover all viewpoints, it is feasible to cover a large number of viewpoints and make observation difference between adjacent viewpoints small enough.

Even so, collecting multiple-view action data from real videos is a formidable task. A more practical way is to use synthetic human figures. We use *POSER*[1], a leading software for generating realistic looking human characters, to render key poses from a variety of viewpoints. We cover $90°$ of camera **tilt** angle at $5°$ intervals and $360°$ of **pan** angle at $10°$ intervals. This results in a total of 19×36=684 images

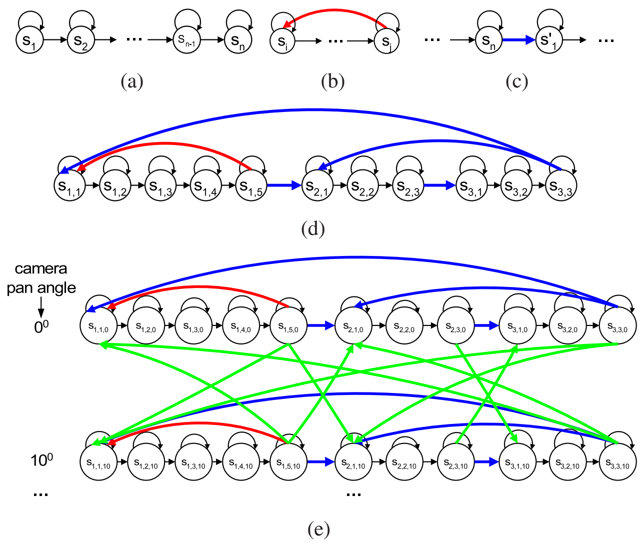


Figure 2. Action graph models. **(a)** The general model of a single action; **(b)** Back-link (in red); **(c)** Inter-link (in blue); **(d)** A simple Action Net consisting of the three actions shown in Fig.1; **(e)** The unrolled version of (d). Only models with the first two pan angles ($0°$ and $10°$) are shown.

for each pose. From a stationary camera, the tilt angle is fixed. If we assume tilt is given (*e.g.* from camera calibration), during recognition, we need to search from only 36 images (rendered from the given tilt angle) for each pose. We can not assume a given pan angle, however, because the orientation of the actor is unknown, which is equivalent to an unknown pan angle. The **roll** angle controls the image rotation. A rotation invariant representation of 2D poses is described later.

We choose the silhouette of a human as the image observation for a 2D pose. Silhouette is widely used [2, 3, 4, 6, 10, 21] due to some obvious advantages: it contains rich shape information of a 2D pose and it is relatively easier to extract a human contour (*e.g.* from background subtraction) than to detect body parts; it is insensitive to internal texture and color, and its shape has been extensively studied. Some drawbacks of this feature include the dependency on the quality of foreground segmentation and ambiguity caused by loss of internal detail. These issues can be overcome to some extent by using robust background subtraction algorithms such as Mixture of Gaussian [19] and considering the contextual constraints, as described in the introduction. Another issue is occlusion. In this work, we assume that only one person is in the scene or there are multiple persons with separate silhouettes, thus mutual occlusion is not present.

The *Shape Context* (*SC*) [2] of silhouettes is used to obtain a robust scale and translation invariant shape representation. As illustrated in Fig.3(a), the SC of each sampled edge point of a silhouette is a log-polar histogram of the

[1]Poser 5, Curious Labs (now e frontier Inc.)

coordinates of the rest of the point set measured using the reference point as the origin. Because only relative scale and position is used here, the representation is invariant to scale and translation. We use 12 angular $\times$ 5 radial bins for each SC and we uniformly sample 200 edge points on each silhouette, so the SC of a silhouette is represented by 200 feature points in a 60-dimensional space, or more intuitively, a 200×60 matrix, as shown in Fig.3(b).

In order to obtain image rotation invariance, we compute the principal axis of the silhouette using the second-order moments [5]. Before computing shape context, the rotation angle of the principal axis is compensated so that the principal axis is vertical.

2.4. Unrolled Action Net

The Action Net described in 2.2 is "unrolled" to model changes in the actor's orientation. First, each action model in the Action Net is unrolled 36 times representing the same action rendered from different camera pan angles (assuming tilt angle is given). We add an additional subscript to each node to represent the pan angle. Second, in this unrolled Action Net, action model i can inter-link to action model j *iff* **(1)** There is an inter-link (including self-loop) from i to j in the original Action Net and **(2)** they are rendered from the same or adjacent viewpoints (*i.e.* difference in pan angle is less than $10°$). For example, the unrolled version of the Action Net in Fig.2(d) is shown in Fig.2(e). The complete graph is quite complex so only the first two pan angles ($0°$ and $10°$) are shown here. The green links are the inter-links that connect action models rendered from adjacent viewpoints. These inter-links allow us to model gradual change in the actor's orientation.

3. Action Recognition and Segmentation

During recognition, human blobs in input frames are segmented using background subtraction and silhouettes of these blobs are extracted. The shape context of each silhouette is matched with all nodes in the unrolled Action Net. Then we consider the matching results as a whole by taking advantage of the contextual constraints imposed by the Action Net to get a robust action recognition result.

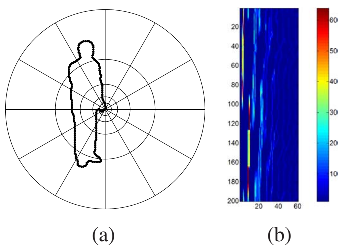


Figure 3. Shape Context Descriptor. **(a)** The shape context of the edge point at the origin, represented as a 60-D vector (12 angular$\times$5 radial bins); **(b)** The shape context of the silhouette shown in (a), represented by a 200×60 matrix.

3.1. Fast Pose Matching

Due to the large number of comparisons, a fast pose matching algorithm is crucial to our system. This makes the Pyramid Match Kernel (PMK) algorithm appealing. PMK is a feature set matching algorithm recently proposed by Grauman and Darrell in [7]. Their results show that PMK achieves comparable results at a significantly lower computational cost than other state-of-the-art approaches. PMK also works with unordered features, which fits the shape context features well. PMK has been applied to object recognition in [7]. We are not aware of PMK previously applied to the action recognition problem.

The basic idea of PMK is that instead of directly comparing features in two feature sets, it calculates intersections over multi-resolution histograms, which gives a linear complexity in the number of features. The similarity between the two sets is defined by the weighted sum of histogram intersections at each level; the weights are proportional to the resolution of each level, as shown in Eq.2

$$K_\Delta(\overbrace{\Psi(X), \Psi(Y)}^{\text{histogram pyramids}}) = \sum_{i=0}^{L} \frac{1}{2^i} \underbrace{\left(\tau(H_i(X), H_i(Y)) - \tau(H_{i-1}(X), H_{i-1}(Y))\right)}_{\text{number of newly matched pairs at level } i} \quad (2)$$

where the notations are as follows: (1) X,Y: feature sets; (2) $\Psi(X)$: histogram pyramid of X; (3) $\frac{1}{2^i}$: weight at level i; (4) $H_i(X)$: histogram of X at level i; (5) $\tau()$: The size of intersection of two sets.

However, when applied to high-dimensional features such as shape contexts, PMK produces a low matching score even when two silhouettes are quite similar. Take Fig.4 for an example. When Fig.4(c) and Fig.4(e) (two query silhouettes) are compared to Fig.4(a) (the reference silhouette), we expect a much higher matching score from Fig.4(c) because of its similar shape to the reference silhouette. But it turns out that both queries get low scores and the first one is only slightly higher.

Let us look at Fig.5(a), which shows the number of newly matched pairs between the reference silhouette and

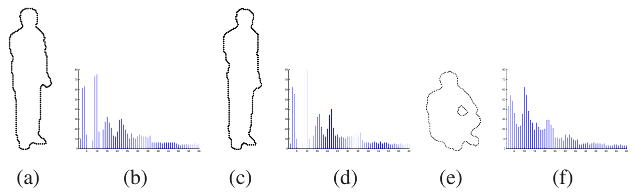


Figure 4. Three silhouettes and their range of feature point coordinates in each dimension of the shape context space. **(a)** Reference silhouette, **(c)** Query silhouette 1, **(e)** Query silhouette 2

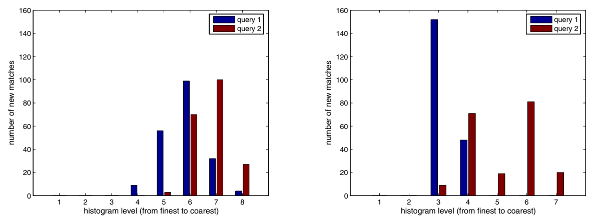


Figure 5. The number of newly matched pairs at each level using two different PMKs. **(a)** The original PMK; **(b)** PMK-NUP. The difference between two queries is enlarged in (b) because large-scale matches take place much earlier in query 1 than in query 2.

the query silhouette at each level of histograms. We can see that large-scale matches occur only at the last several coarsest levels for both queries. Because the weights associated with theses levels are very small (the weights decrease exponentially with the level), the overall matching scores are small in both cases. A further explanation is that a limited number of feature points usually scatter sparsely in a high-dimensional feature space, so at the finest levels, the probability of points falling into the same bin is small.

This issue is also reported by Grauman and Darrell in [8], where they propose to use hierarchical k-means clustering to partition the high-dimensional feature space into a hierarchy of non-uniformly shaped bins. In our experiments however, features are so sparse (200 points in a 60-D space) that they do not exhibit apparent aggregation, so it is hard to decide the proper number of clusters.

We suggest a different enhancement instead. It is based on the observation that the shape context features are usually not distributed in a hyper-cubic space. In fact, data points spread more sparsely along some dimensions than others. (See Fig.4(b),4(d),4(f) for some real examples.) More intuitively, think of a stripe-shaped cloud of data points in a 2D space. The original PMK partitions the space using a set of square bins. When the side length of bins doubles at each coarser level, points quickly converge along the short side of the stripe. However, since the resizing speed is the same for both sides, it takes longer time for the long side of the stripe to converge. In this sense, it is the long side of the stripe that prevents matching at the finest levels.

We take advantage of this distribution and force data points to converge faster along these "long sides". Instead of using a common scale factor for all dimensions, we assign a different scale factor for each dimension based on the range of data in this dimension so that data points converge at the same speed in all dimensions. This results in a **n**on-**u**niform **p**artitioning in the feature space. This idea, termed as **PMK-NUP** in this paper, is illustrated and compared with the original PMK in Fig.6.

Consider a set of n feature points in a d-dimensional space: $\{(x_{1,1},...x_{1,d}),...,(x_{n,1},...x_{n,d})\}$. The range of

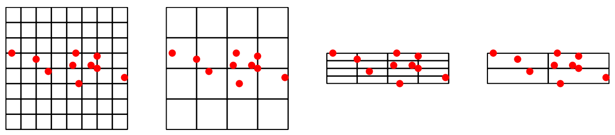


Figure 6. The original PMK uses a uniform partitioning scheme (left two). For a stripe-shaped distribution, the long side of the stripe converges slower than the short side. With PMK-NUP (right two), both sides converge at the same speed.

these points in the j-th dimension is:

$$r_j = \max_{i=1}^{n}(x_{i,j}) - \min_{i'=1}^{n}(x_{i',j}) \qquad j = 1,2,...,d \quad (3)$$

Suppose that the k-th dimension has the smallest value of range r_{min}. Considering only the k-th dimension, the original PMK (with a scale factor of 2) takes $log_2 r_{min}$ steps to converge to a single bin. For another dimension j, in order to converge at the same speed, s_j, the scale factor the j-th dimension, has to satisfy $log_{s_j} r_j = log_2 r_{min}$, and thus

$$s_j = max(r_j^{\frac{1}{\log_2 r_{min}}}, 2) \quad (4)$$

Here we impose a lower bound of 2 for s_j so that points will not converge too slowly along the j-th dimension.

In practice, r_{min} can be very small and thus result in large scale factors for all dimensions. That will force points converge to a single bin in all dimensions immediately. So we impose a lower bound of 8 on r_{min}.

$$r_{min} = \max(\min_{j=1}^{n}(r_j), \quad 8) \quad (5)$$

PMK is changed to PMK-NUP as follows: when a bin is being resized, the scale factor s_j (instead of 2) is applied to the j-th side of the bin. Note that the scale factors are computed based on the reference silhouette. The same scale factors are applied to the query silhouette when it is compared to the reference silhouette.

The question is: will PMK-NUP also increase the matching score between two different silhouettes? The answer is yes, but not as much as the increase between two similar silhouettes. This is because different silhouettes have different distributions. Take Fig.4 for an example. The ranges in some dimensions of the "sitting" silhouette are significantly larger than the same dimensions of the "standing" silhouettes. Because the partitioning is based on the reference silhouette, those dimensions will converge at a much lower speed in the "sitting" query, which ends up preventing large-scale matches in the finest levels. Compare the newly matched pairs of two query silhouettes using the original PMK and PMK-NUP, as shown in Fig.5(a) and 5(b) respectively, we can see that with PMK-NUP, although large-scale matches take place earlier in both queries, the difference between the two queries is actually enlarged.

3.2. Searching for the Best Action Sequence

Suppose there are N nodes in the unrolled Action Net and T frames in the input video. After the previous step, we get an $N \times T$ array of matching scores. To find out which action is being performed at the i-th frame, the simplest solution is to assign the action label from the best matched key pose for the i-th frame. However, this solution may be oversimplified because it overlooks the following issues. **(1)** Silhouettes can be easily distorted by a bad foreground segmentation; **(2)** Even if silhouettes are clean, different actions can share similar 3D poses and different 3D poses can get similar silhouettes (from specific viewpoints); **(3)** Because only a small set of key poses are used, some input silhouettes may not find a good match anyway and thus the matching results for these silhouettes are left undefined. All these issues indicate that the decision based on an individual matching score is unreliable.

To robustly recognize actions from unreliable individual observations, we take advantage of the contextual constraints imposed by the Action Net and formulate the action recognition and segmentation problem as finding the most likely sequence of nodes within the Action Net. The famous Viterbi algorithm [15] is a suitable tool for this task. The toy example shown in Fig.7 illustrates how the Viterbi algorithm is employed in our system. Consider the Action Net in Fig.7(a) and the corresponding array of matching scores in Fig.7(b). The goal is to find a path (called the Viterbi Path) through the array from left to right that has the maximum sum of matching scores. See Fig.7 for a brief description of the algorithm. For details of the Viterbi algorithm, we refer to the excellent tutorial in [15].

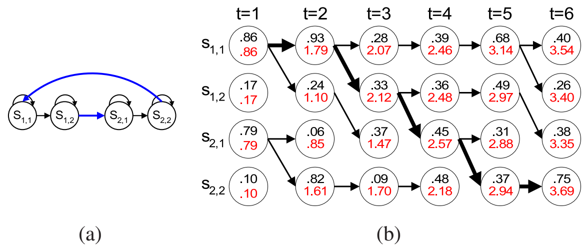
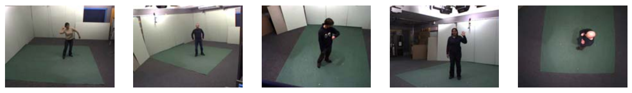


Figure 7. Action recognition and segmentation using the Viterbi algorithm. **(a)** An Action Net consisting of two simple action models. Note that an unrolled Action Net is actually used. **(b)** The Viterbi algorithm finds the most likely sequence of 2D poses (and thus actions) called the **Viterbi Path**. Each element (i, t) of this 2D array keeps three values: the matching score (in black) between node i in the Action Net and input frame t, the best score (in red) along a path up to (i, t) and the previous element on this path. At time t, element (i, t) looks at every node that links to node i in the Action Net and chooses the one with the maximum path score. Element (i, t) then links itself to this path, updates the path score by adding its own matching score and records the previous element. When the last frame has been processed, the element with the maximum path score is found and the Viterbi Path (in bold) is back tracked. The complexity of the Viterbi algorithm for a fully ergodic graph model is $O(N^2T)$, where N is the number of nodes in the graph and T is the number of frames. For an Action Net, because the average in-degree of each node is small, the overall complexity reduces to $O(NT)$.

Note that we assume a uniform transitional probability for each link (and thus neglected) of the Action Net. This is because modeling transitional probability for each link of such a complex graph (with thousands of links) requires a huge training set, which is not applicable in practice.

4. Experimental Results

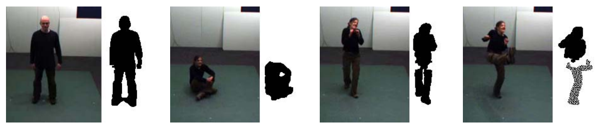
We demonstrate the proposed approach on a public video set[2], which contains 180 video clips (36 shots taken from 5 cameras) of one of 12 actors performing 15 action classes. The average length of a clip is 1165 frames. One example from each camera is shown in Fig.8. The actors can freely choose position and orientation in these clips. The order and the number of action instances performed in each clip also vary. The large number of action classes and the large variance in viewpoints and subjects (and thus the execution styles) make action recognition a challenging task.



(a) Camera 1 (b) Camera 2 (c) Camera 3 (d) Camera 4 (e) Camera 5

Figure 8. The Inria Xmas dataset. One example from each camera is shown.

Human blobs are provided with the dataset. The quality of these blobs is generally good but many defects are also present. See Fig.9 for some examples. Morphological closing operation is applied to repairing some of the defects but this is ineffective for the severely contaminated blobs.



(a) (b) (c) (d)

Figure 9. Examples of defects in the provided blobs, including **(a)** shadows **(b)** missing body parts **(c)** broken contours and **(d)** corrupted blobs

The 15 included action classes are listed in the first column of Table 1. Based on the observation of actions performed by two actors (one male and one female), we manually select motion capture sequences with the motions similar to the observed actions from a large motion capture

[2]Videos are obtained from Inria Xmas Motion Acquisition Sequences, available at https://charibdis.inrialpes.fr/html/non-secure-sequence.php?s=IXMAS

database. The extracted key poses for each action class are shown in Fig.10. Videos of both male and female actors are included because their execution styles in actions such as "punch" and "kick" are significantly different. Such variance is handled using different action models. As seen in Fig.10, "punch", "kick" and "point" each has two models. The symmetric counterparts of some actions (e.g. "wave left hand" for "wave right hand") are also included (not shown in the figure). In total, we get 177 key poses. Since cameras are calibrated, with the known tilt angle, each key pose is rendered from 36 pan angles. So the unrolled Action Net (for each camera) has 6372 nodes. The shape contexts of the 2D silhouettes are computed and stored in these nodes.

Action	c1	c2	c3	c4	c5	Overall
stand still	73.9	71.2	68.9	73.5	70.1	71.1
check watch	82.8	82.1	81.9	84.2	81.4	82.5
cross arms	83.2	84.3	79.9	84.4	80.6	82.1
scratch head	81.1	81.5	80.1	80.6	77.1	80.2
sit down	86.3	85.3	83.2	82.1	81.4	83.7
get up	85.8	86.7	82.5	84.4	79.1	84.3
turn around	81.1	80	80.5	78.9	75.3	78.8
walk in a circle	79.2	83.3	79.3	79.8	74.4	79.7
wave a hand	80.2	82.3	77.6	81.3	78.1	79.9
punch	87.1	87.7	84.4	88.3	84.6	86.8
kick	89.1	89.6	83.3	89.4	85.3	87.7
point	81.5	83.6	87.1	80.2	79.5	82.7
pick up	83.8	85.9	84	85.1	79.4	83.2
throw over head	81.6	82.3	78.9	80.1	83.3	81.3
throw from bottom	80.1	81.9	81.5	82.4	85.4	82.4
Overall	81.5	82.1	80.1	81.3	78.4	80.6

Table 1. The percentage of correctly labeled frames for each action class. Each column contains results from one of the five cameras.

We tested our system on 50 videos clips of the 10 actors that are not included in the training data. The total length of these clips (10 actors $\times$ 5 cameras) is more than 58,000 frames. For each clip, after the best action sequence is tracked, we compare the resulting action label at each frame with the provided ground truth. The recognition rate is defined as the percentage of correctly labeled frames. The recognition rate for each action class from each camera as well as the overall recognition rate are listed in Table 1. Some result frames are shown in Fig.11. Please see the supplementary material for some result videos.

Our approach achieves an overall action recognition rate of 80.6% using a single camera. This is very promising considering the complexity of the test and the small amount of data that we used for training. In [20], Weinland et al. report a higher action classification rate (93.3%) on the same dataset. They use all five cameras to build visual hulls and classify actions based on a 3D action representation called Motion History Volumes. However, instrumenting with many cameras, each highly calibrated is not feasible for many applications and reconstruction of 3D shape from multiple views usually is computationally expensive.

Our results show that among all 15 action classes, "kick" and "punch" seem to be the easiest actions to recognize. This is probably because the motion in these two actions is more noticeable than in other actions. The high scores for "sit down", "get up" and "pick up" can also be attributed to this reason. Some arm related actions such as "scratch head", "wave a hand" and "throw over head" got relatively low scores mostly because they have several (similar) key poses in common. The same explanation applies to "walk in a circle" and "turn around". "Stand still" received the lowest score which is not surprising because it has only one key pose and this key pose is similar to the boundary key poses of other actions, such as the staring key pose of "turn around" and the ending key pose of "get up". So there is not a clear cut between "stand" and these actions.

The performance in general is consistent with respect to the camera viewpoints, but there are some notable differences. For example, two "throw" actions are better recognized in camera 5 because the motion appears more salient in this viewpoint. The performance for "sit down" and "get up" in camera 5 is lower than average for the opposite reason.

To justify the use of the PMK-NUP and Viterbi Path searching, we also tested the following approaches using the same experimental setup: (1) Original PMK only, (2) PMK-NUP only, (3) Original PMK with Viterbi. For the first two approaches, at each frame, the action label of the pose with the largest matching score is selected. The overall recognition rates of these approaches are listed in Table 2.

	original PMK	PMK-NUP
without Action Net	38.4%	44.1%
with Action Net	56.7%	80.6%

Table 2. Comparison of different approaches. PMK-NUP combined with Viterbi significantly outperforms the other three approaches.

It is interesting to observe that after applying the Action Net, PMK-NUP performs significantly better than the original PMK (from 44.1% to 80.6%). This is because for each input frame, it is likely that the original PMK and PMK-NUP find the same best matched pose. However, the difference between a good match and a bad match in PMK-NUP is larger than in the original PMK. When combined with Viterbi Path searching, this difference in PMK-NUP is amplified over time and finally makes the best pose sequence easily recognizable. The effectiveness of the Action Net is clearly seen from this comparison.

Our system runs at 5.1 frames/sec on a single P4 3GHz CPU. Most of the CPU time is spent on silhouette matching. Because silhouette matches are computed independently, this system can be easily parallelled to give a real time speed.

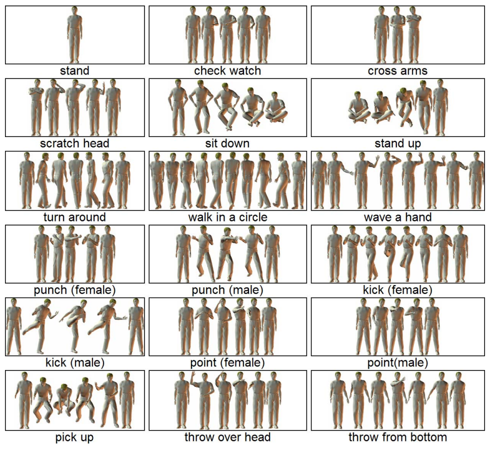


Figure 10. The extracted key poses for each action class.

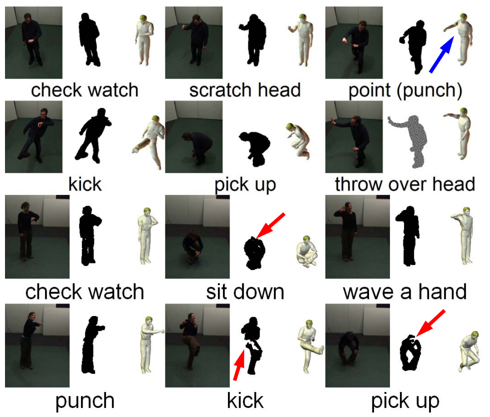


Figure 11. Some result frames. Each example shows the original image, the provided human blob, the recovered pose and the action label, respectively. Due to the contextual constraints, our approach is not sensitive to the short-term defects in human blobs, as indicated by the red arrows. The blue arrow shows one failed example, in which a punching action is recognized as pointing. Those two actions, however, do appear similar.

5. Conclusions

We have presented an example based single view action recognition system and demonstrated it on a challenging test set consisting of 15 action classes. Our major contributions include: (1) a novel action representation scheme called Action Net, which inherently models the contextual constrains for action recognition and segmentation, and (2) an enhanced Pyramid Match Kernel algorithm that takes advantage of feature distribution and greatly improves the matching score between two similar feature sets.

Acknowledgements

This research was funded, in Part, by the U.S. Government VACE program and, in part, by Office of Naval Research, under contract #N00014-06-1-0470.

References

[1] A. Agarwal, B. Triggs, “3D Human Pose from Silhouettes by Relevance Vector Regression”, *CVPR*, pp. 882-888, 2004.

[2] S. Belongie, J. Malik, J. Puzicha, “Shape Matching and Object Recognition Using Shape Contexts”, *PAMI* 24(4), pp. 509-522, 2002.

[3] M. Blank, L. Gorelick, E. Shechtman, M. Irani, R. Basri, “Actions as space-time shapes”, *ICCV*, pp. 1395-1402, 2005.

[4] A.F. Bobick, J.W. Davis, “The recognition of human movement using temporal templates”, *PAMI* 23(3), pp. 257-267, 2001.

[5] K. R. Castleman, “Digital Image Processing”, Prentice Hall, pp. 494-496, 1996.

[6] A. Elgammal, C.S. Lee, “Inferring 3D body pose from silhouettes using activity manifold learning”, *CVPR*, pp. 681-688, 2004.

[7] K. Grauman, T. Darrell, “The pyramid match kernel: discriminative classification with sets of image features”, *ICCV*, pp. 1458-1465, 2005.

[8] K. Grauman, T. Darrell, “Approximate Correspondences in High Dimensions”, *NIPS*, 2006.

[9] Y. Ke, R. Sukthankar, M. Hebert, “Efficient Visual Event Detection using Volumetric Features”, *ICCV*, pp.166-173, 2005.

[10] R. Kehl, M. Bray, L.J. Van Gool, “Full Body Tracking from Multiple Views Using Stochastic Sampling”, *CVPR*, pp. 129-136, 2005.

[11] I. Laptev, T. Lindeberg, “Space-time interest points”, *ICCV*, pp. 432-439, 2003.

[12] M. W. Lee, I. Cohen, “Proposal Maps Driven MCMC for Estimating Human Body Pose in Static Images”, *CVPR*, pp. 334-341, 2004.

[13] T. B. Moeslund, E. Granum, “A survey of computer vision-based human motion capture”, *CVIU*, 81(3):231-268, 2001.

[14] C. S. Myers, L. R. Rabiner, “A comparative study of several dynamic time-warping algorithms for connected word recognition”, The Bell System Technical Journal, 60(7):1389-1409, 1981.

[15] L. R. Rabiner, “A tutorial on Hidden Markov Models and selected applications in speech recognition”, In *Proc. of the IEEE*, 77(2):257-286, 1989.

[16] D. Ramanan, D. A. Forsyth, “Automatic Annotation of Everyday Movements”, *NIPS*, 2003.

[17] G. Shakhnarovich, P. Viola, T. Darrell, “Fast Pose Estimation with Parameter-Sensitive Hashing”, *ICCV*, pp.750-757, 2003.

[18] C. Sminchisescu, A. Kanaujia, Z. Li, D. Metaxas, “Conditional models for contextual human motion recognition”, *ICCV*, pp. 1808-1815, 2005.

[19] C. Stauffer, W.E.L. Grimson, “Adaptive background mixture models for real-time tracking”, *CVPR*, pp. 246-252, 1999.

[20] D. Weinland, R. Ronfard, and E. Boyer, “Free Viewpoint Action Recognition using Motion History Volumes”, *CVIU*, 103(2-3), pp. 249-257, 2006.

[21] A. Yilmaz, M. Shah, “Actions sketch: a novel action representation”, *CVPR*, pp. 984-989, 2005.