Accepted Manuscript

Human Movement Analysis around a View Circle Using Time-Order Similarity Distributions

Chi-Hung Chuang, Jun-Wei Hsieh, Hui-Fen Chiang, Yi-Da Chiou

PII: S1047-3203(15)00027-9
DOI: http://dx.doi.org/10.1016/j.jvcir.2015.02.003
Reference: YJVCI 1480


Received Date: 26 November 2013
Accepted Date: 9 February 2015


This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
Human Movement Analysis around a View Circle Using Time-Order Similarity Distributions

Chi-Hung Chuang¹, Jun-Wei Hsieh², Hui-Fen Chiang², and Yi-Da Chiou³

¹Dept. of Learning and Digital Technology, Fo Guang University
   No.160, Linwei Rd., Jiaosi, Yilan 26247, Taiwan

²Dept. of Computer Science and Engineering, National Taiwan Ocean University,
   No.2, Beining Rd., Keelung 202, Taiwan
   shieh@ntou.edu.tw

³Department of Electrical Engineering, Yuan Ze University,
   135 Yuan-Tung Road, Chung-Li 320, Taiwan

Abstract

This paper presents a new behavior classification system to analyze human movements around a view circle using time-order similarity distributions. To maintain the view invariance, an action is represented not only from its spatial domain but also its temporal domain. After that, a novel alignment scheme is proposed for aligning each action to a fixed view. With the best view, the task of behavior analysis becomes a string matching problem. One novel idea proposed in this paper is to code a posture using not only its best matched key posture but also other unmatched key postures to form various similarity distributions. Then, recognition of two actions becomes a problem of matching two time-order distributions which can be very effectively solved by comparing their KL distance via a dynamic programming scheme.

Keywords: Video Surveillance, Centroid Context, Behavior Analysis, View Invariance, View Circle
1. Introduction

The analysis of human body movements can be applied in a variety of application domains, such as video surveillance, video context retrieval, human-computer interaction systems, and medical diagnoses. In some cases, the results of such analysis can be used to identify people acting suspiciously and other unusual events directly from videos. In the past, many approaches [1]-[26] have been proposed for video-based human movement analysis. For example, Weinland and Boyer [3] proposed a clustering scheme to select a set of key postures for converting action sequences to different feature vectors. Then, a Bayesian classifier is designed to classify them to different event categories. Fathi and Mori [6] detected corner features as well as their motion vectors to represent an event, and then classified it to different event types. In [7], Ju et al. used the SIFT detector to extract different feature points and their trajectories as event descriptors, and then trained a SVM (Support Vector Machine) -based classifier for event classification. In [8], based on dynamic Bayesian networks, Wu et al. integrated RFID and video streams to model various kitchen events. In [9], the MHI (Motion History Image) feature was used to represent human motions and further classified to different event types by a SVM-based classifier. Furthermore, Kratz and K. Nishino [10] used a set of local spatio-temporal motion volumes to represent actions and then proposed a HMM-based framework to analyze the overall behavior of an extremely crowded scene. Some approach decomposes actions into sequences of key atomic action units which are referred to as atoms. In [11], Gaidon, Harchaoui, and Schmid proposed an actom sequence model (ASM) to represent the temporal structure of actions and then recognized actions in videos using a sequence of “atoms”. Here, atoms which are specific to each action class should be identified by manual annotation.

In addition to videos, some approaches recognized human activities based on only still images. For example, in [12], Yao and Fei-Fei proposed a data-mining scheme to discover human-object interactions by encoding a set of highly related patches to grouplets with their
appearances, locations, and spatial relations. Maji, Bourdev, and Malik [13] trained thousands of poselets to form a poselet activation vector within the bounding box of a person so that the related actions can be recognized from still images. However, the prerequisite that body parts or poses must be well estimated makes this model-based scheme inappropriate for real-time analysis of human behaviors.

Another key problem in these approaches is “view invariance”. Due to perspective effects, an action will look different from different viewpoints. Large appearance changes will lead to the failure in recognizing an action from different views. To maintain the view invariance, some researchers analyzed human actions using 3-D human models. For example, Shakhnarovich et al. [14] stored a dataset of human models with known 3-D parameters and then estimated best 3-D posture models for action analysis. Another approach for arbitrary view action analysis is to directly estimate 3D shapes and poses from multi-view 2-D image data. For example, Balan et al. [15] uses a triangulated mesh model called SCAPE [31] which employs a low-dimensional, but detailed, parametric shape model for recovering 3-D models directly from a multi-camera system. Weinland et al. [4], [5] segmented the human actions into various primitives and clustered them to different action atoms using the feature “motion history volumes”. The view invariance can also be achieved by building the connections between different views via a view alignment or matching technique. For example, in [16]-[17], Farhadi et al. used a codebook technique to construct split-based descriptors and then counted their frequencies to build view-invariant latent models for object recognition and activity analysis. Lv and Nevatia [18] applied the PMK (Pyramid Match Kernel) algorithm [19] to recognize postures and then matched each action from arbitrary views using the Viterbi [20] algorithm. In [21], Souvenir and Babbs used a manifold learning technique to learn a set of action primitives from a single camera to form a viewpoint-invariant representation. In [22], Junejo et al. built trajectory-based
self-similarity matrices as pair-wise distances between 2D hand positions of persons for action recognition under different views. Furthermore, Cuzzolin [23] built different bilinear HMMs to analyze human gaits from multiple views. Karthikeyan, Gaur, and Manjunath [24] computed R transforms on action silhouettes and then proposed a probabilistic subspace similarity technique to recognize actions from multiple views by learning their inter-action and intra-action models. However, this scheme requires all action sequences from different views must be collected together before recognize an action.

In addition to 2-D and 3-D data, another trend for behavior analysis is to project the matching problem into a higher dimensional feature space. For example, Lee and Elgammal [25] projected features on a higher dimensional space and then used the techniques of singular value decomposition and manifold learning to model and analyze an articulated object around a view circle. Yan et al. [26] mapped an action to a 4-D model space and then derived its optimal model parameters to recognizing behaviors. Although 4-D features are more informative for human action recognition, their high computation cost makes this approach inappropriate for real-time applications.

This paper proposes a novel matrix-based scheme for recognizing human actions around a view circle using only 2D postures. The flowchart of our method is shown in Fig. 1. Two stages are included in our proposed system, i.e., the training and recognition stages. At the training stage, we sample the viewing angle around the actor and then divide the action space to several subspaces. The basis of each action subspace is a set of key postures which are extracted using a cluster technique. To achieve this clustering task, the centroid context descriptor is constructed to describe a posture up to its syntactic levels. To maintain the view invariance, a larger set of action subspaces must be constructed and thus leads to the inefficiency in action analysis. To reduce the action space, this paper uses the mirror symmetry for reducing the whole view space to only its quarter. At the recognition stage, a
novel view alignment scheme is then proposed to search the best view via the Viterbi algorithm. Thus, even though an action is captured from an unknown view angle, it still can be well analyzed and recognized. To avoid false matches and reduce the searching space, a table for recording the transition probability between any two key postures is built in the alignment scheme. Thus, two actions sequences can be compared by using not only their spatial similarities but also their temporal similarities. Once the best view is selected, each action can be converted to a string. However, many errors will happen during the converting process. The novelty of this paper is to code a posture (or frame) using not only its best matched key posture but also its similarities to other key postures. Then, recognition of an action taken from a view circle can be represented with a similarity distribution which changes along the time. After that, two actions can be matched by calculating the Kullback–Leibler distance between their time-order similarity distributions. The performance of the proposed method has been rigorously tested on a variety of behavior videos to prove its superiority in human behavior analysis from both efficiency and robustness perspectives.

The rest of this paper is organized as follows. In Section 2, the descriptor for posture representation will be described. The schemes for action space sampling and model construction are described in Section 3. Section 4 discusses the details of view alignment. The matrix-based representation for behavior analysis is discussed in Section 5. Section 6 reports experimental results and finally a conclusion will be presented in Section 7.

2. Posture Representation Using Centroid Context

To describe an action event, this paper uses the descriptor “centroid context” [27] to describe a posture up to its syntactic levels. The descriptor can tolerate large posture distortions than shape context [28]. In addition, its time complexity is much lower than the one of shape context. Other features like “R-transform” [29] are also good and used here for action
Given a posture $P$, this scheme partitions $P$ to different triangulation meshes. Then, $P$ can be decomposed to different body parts $\{b_i\}_{i=1,...,|P|}$, where $|P|$ denotes the number of body parts in $P$. For each body part $b_i$, its centroid distribution $h^p_{b_i}$ can be extracted by projecting all the centroids of triangulation meshes in $P$ on a polar coordinate where the center of $b_i$ is set as the origin. Then, the centroid context of $P$ is defined as follows:

$$P = \{h^p_{b_i}\}_{i=0,...,|P|-1},$$

(1)

Then, given two postures $P$ and $Q$, their distance can be calculated by the form

$$d_c(P, Q) = \frac{1}{2|P|} \sum_{i=0}^{|P|-1} w^p_i \min_{0 \leq j < |Q|} C(h^p_{b_i}, h^q_{b_j}) + \frac{1}{2|Q|} \sum_{j=0}^{|Q|-1} w^q_j \min_{0 \leq i < |P|} C(h^p_{b_i}, h^q_{b_j}),$$

(2)

where $w^p_i$ and $w^q_j$ are the area ratios of the $i$th and $j$th body parts residing in $P$ and $Q$, respectively. $C(h^p_{b_i}, h^q_{b_j})$ is the normalized interaction distance between $h^p_{b_i}$ and $h^q_{b_j}$, i.e.,

$$C(h^p_{b_i}, h^q_{b_j}) = 1 - \frac{1}{N_{\text{mesh}}} \sum_{k=1}^{\text{Bins}} \min \{h^p_{b_i}(k), h^q_{b_j}(k)\},$$

(3)

where $\text{Bins}$ is the number of used bins and $N_{\text{mesh}}$ denotes the number of used meshes.

Based on Eq.(2), different postures can be well compared.

3. Model Construction

In most of surveillance environments, the camera is fixed and mounted at a fixed height and the person can change freely his/her orientations to the camera. Thus, we use a view circle to model possible viewpoint variations and thus to avoid the task of camera calibration. A view circle is a set of observations captured by a camera moved around the actor with a fixed radius. With this view circle, we divide the action space to different subspaces. Based on the set of subspaces, different actions even observed from different views can be well recognized. This approach assumes only horizontal rotation of the body in 3D and the 2D silhouette of the actor is perspective-corrected onto a front-parallel view. Although this approach is not so
much view-invariant for all possible viewpoints, it is more realistic and can be done in real
time. In the surveys [1], [2], there are several 3D approaches which can achieve really
view-invariant action recognition. However, these approaches involve an exhaustively
searching task to find the best view via a nonlinear optimization process which enables them
to be inappropriate for real-time applications.

3.1 View Sampling
In most of surveillance cases, the camera is fixed and mounted at a fixed height. Under this
setup, a view circle is a possible approximation for capturing all possible viewpoint variations.
Like Fig. 2(a), we can sample the view circle (ranging from 0° to 360°) to construct the
action space. If a dense rate is adopted for sampling the view circle, this space will become
large and lead to inefficiency and ambiguity in action comparison. This paper proposes two
schemes to reduce the searching space by enlarging the sampling step and mirroring the
action space, respectively.

For the sampling step, since the “centroid context” [27] can tolerate large posture
distortions than the shape context [28], we adopt a larger interval 30° for view sampling.
Thus, a smaller set of action space can be constructed. For the mirroring step, two kinds of
mirror symmetry of postures are adopted to further reduce the action space. Assume the
front view is at the view angle 0° (see Fig. 2(a)). For a posture, its cloth colors and textures
are not used here for posture analysis. Under this situation, the first mirror symmetry
happens between two viewing angles if their angle difference is 180° (modulo by 360°).
Like Fig. 2(b), the postures of the actor viewed, respectively, at the angles 45° and −135°
are mirror-symmetric. Then, only the view angles ranging from 90° to −90° (see Fig. 2
(c)) need to be sampled and used for action representation. The second mirror symmetry
happens if the angle sum of two views is zero (modulo 360°). Like Fig. 3 (a), the “walking”
postures captured at the viewing angles −30° and 30° are mirror-symmetric. In Fig. 3(b),
the “squatting” postures at the view angles -90° and 90° are also mirror-symmetric. According to the two mirror symmetries, only the view angle ranging from 0° to 90° should be sampled and used in action representation. This technique can reduce the whole view space to only its quarter. When the sampling step 30° is chosen, only four action models at the view angles 0°, 30°, 60°, and 90° need to be really generated. Other action subspaces can be mirrored and re-generated for action analysis.

3.2 Action Space Construction via Clustering

When an action sequence is observed, some of its postures may be redundant, so they should be removed from the modeling process. This section will present a clustering technique for selecting a set of key postures from a collection of real-world video clips. Then, the action subspace for each view angle can be constructed for view-invariant action analysis.

At a specific view, a set, \( \Omega \), of training video clips is first collected. Given two adjacent postures, \( Q_{t-1} \) and \( Q_t \), in \( \Omega \), their distance, \( d_t \), can be calculated using Eq.(2). Assume \( T_d \) is the average value of \( d_t \) for all pairs of adjacent postures. For a posture \( Q_t \) in \( \Omega \), if \( d_t \) is greater than \( 2T_d \), we define it a posture-change instance. Given a posture-change instance \( Q_t \) and its next posture-change instance \( Q_{t+1} \), we can collected a new set \( \Omega_t \) of ordered postures; that is,

\[
\Omega_t = \{ Q_t | Q_t \in \Omega, t_i \leq t \leq t_{i+1} \}.
\]

Usually, discriminative key poses appear when the body is relatively motionless. Thus, from \( \Omega_t \), we can extract a potential key posture \( \overline{Q}_t \) that satisfies

\[
\overline{Q}_t = \arg \min_{Q_m \in \Omega \cap \Omega_t} \sum_{Q_n \in \Omega \cap \Omega_t} d_{nt} (Q_m, Q_n).
\]  

(4)

By collecting all the potential key postures from Eq.(4), we derive a set \( S_{KPC} \) of key posture candidates. However, \( S_{KPC} \) may still contain many redundant postures. To address this problem, we use a clustering technique to find a better set of key postures to represent actions. Initially, we assume each element \( e_t \) in \( S_{KPC} \) individually forms a cluster \( z_t \). Then,
given two cluster elements, \( z_i \) and \( z_j \), in \( S_{KPC} \), their distance is defined by:

\[
d_{\text{cluster}}(z_i, z_j) = \frac{1}{|z_i \cap \| z_j |} \sum_{e_m \in z_i} \sum_{e_n \in z_j} d_{cc}(e_m, e_n),
\]

where \( d_{cc}(\cdot, \cdot) \) is defined in Eq.(2) and \( |z_i| \) represents the number of elements in \( z_i \).

With Eq.(5), we can execute an iterative merging scheme to find a compact set of key postures from \( S_{KPC} \). Let \( z'_i \) and \( Z' \) be the \( i \)th cluster and the collection of all clusters \( z'_i \) respectively at the \( i \)th iteration. At each iteration, we choose a pair of clusters \( z'_i \) and \( z'_j \) whose distance \( d_{\text{cluster}}(z'_i, z'_j) \) is the minimum among all pairs in \( Z' \), i.e.,

\[
(z'_i, z'_j) = \arg \min_{(z_m, z_n)} d_{\text{cluster}}(z_m, z_n), \text{ for all } z_m \in Z', z_n \in Z', \text{ and } z_m \neq z_n.
\]

If \( d_{\text{cluster}}(z'_i, z'_j) \) is less than \( T_d \), then \( z'_i \) and \( z'_j \) are merged to form a new cluster and thus constructing a new collection of clusters \( Z'^{i+1} \). The merging process is executed iteratively until no further merging is possible. Assume \( Z' \) is the final set of clusters after merging.

Then, from the \( i \)th cluster \( z'_i \) in \( Z' \), we can extract a key posture \( e_{i, \text{key}} \) that satisfies

\[
e_{i, \text{key}} = \arg \min_{e_m \in z'_i, e_n \in z'_i} \sum d_{cc}(e_m, e_n).
\]

Based on Eq.(6) and checking all clusters in \( Z' \), the set \( S_{KP} \) of key postures, i.e.,

\[
S_{KP} = \{ e_{i, \text{key}} \}
\]

can be constructed for human action sequence analysis. After integrating all subspaces extracted from different view angles, i.e., \( 0^\circ, 30^\circ, 60^\circ, \) and \( 90^\circ \), a compact action space can be formed for invariant action analysis.

4. View Alignment

After action space construction, this section will present a novel view alignment method to align an action sequence to a specific view.

4.1 Fast Posture Matching Using Ratio Constraint

Given an action \( A \), to recognize it around a view circle (denoted by \( V \)), we divide \( V \) into
several view subspace $V_i$. Each subspace $V_i$ is further represented by a set of key postures. Since $A$ is acquired from an unknown view, we should first find its best view point and then align it to a standard view $v_s$. With $v_s$, we can then recognize $A$ using the action subspace $V_s$ created at $v_s$. As shown in Fig. 4, this paper assumes that the standard view is set at the view angle $90^\circ$ (denoted by $v_{90^\circ}$). If $A$ is observed at the angle $0^\circ$, we will align $A$ to $V_{90^\circ}$ and then recognize it. Because the size of key postures created for action representation is larger, before alignment, a geometric constraint is first introduced to filter out impossible posture matches for efficiency improvement.

Let $q_i$ denote the $i$th posture in $A$ and $e$ be one of key postures in the database. In addition, let $\gamma_q$ and $\gamma_e$ denote the width-to-height ratios of $q_i$ and $e$, respectively. If $q_i$ and $e$ are similar, $\gamma_q$ and $\gamma_e$ should be also similar. Thus, it is unnecessary to calculate their distance $d_{cc}(q_i,e)$ if $\gamma_q$ and $\gamma_e$ are not similar enough. Thus, the rule for calculating $d_{cc}(q_i,e)$ will become

$$
\begin{align*}
&\begin{cases}
\text{Compute } d_{cc}(q_i,e), & \text{if } |\gamma_q - \gamma_e| < \tau, \\
\text{Ignore } e, & \text{otherwise},
\end{cases}
\end{align*}
$$

where $\tau$ is a threshold. Only the set of key postures whose width-to-height ratios are similar to $\gamma_q$ will be further compared. The constraint can also filter out false matches if postures are with similar shapes but different width-to-height ratios. As shown in Fig. 5, (b) shows the searching results of (a) without using this constraint. Several false matches were also retrieved because of their similar shapes. In (c), the false matches were further filtered out by using the geometrical constraint.

### 4.2 Candidate Refinement Using Transition Probability

Before action recognition, each action will be first converted to a series of time-ordered symbols. In some cases, two different posture types will share similar contours (as the
examples shown in Fig. 6). If only the spatial similarity is used, the similar contours will lead to many converting errors. In real conditions, it is seldom to see a “fallen” posture type will become a “standing” type immediately. This kind of temporal consistence can be used to improve the accuracy of posture matching. To achieve this goal, a transition table for recording the probability of one key posture transiting to another one must be created.

To build this table, a set $B$ of training action sequences is first collected. Let $\beta_i$ denote the $i$th action video clip in $B$. In addition, $|B|$ and $|\beta_i|$ represent the number of videos in $B$ and the number of postures in $\beta_i$, respectively. Then, the transition probability $p_{\text{transition}}(e_j \mid e_i)$ between any two key postures $e_i$ and $e_j$ can be estimated as follows:

$$p_{\text{transition}}(e_j \mid e_i) = \frac{1}{N_f} \sum_{l=1}^{n} \sum_{t=1}^{|\beta_l|} \alpha_{i,j},$$

where $N_f = \sum_{l=1}^{|B|} |\beta_l|$ and $\alpha_{i,j}$ is a transition flag. Let $p_i^t$ and $p_{i+1}^t$ denote two adjacent postures in $\beta_i$ at time $t$ and time $t+1$, respectively. $\alpha_{i,j}^{t,t+1}$ will be 1 if the types of $p_i^t$ and $p_{i+1}^t$ are $e_i$ and $e_j$, respectively; otherwise, it is 0. After training, we can create a table $T(e_i,e_j)$ for recording the probability of key posture $e_i$ transiting to $e_j$, i.e.,

$$T(e_i,e_j) = p_{\text{transition}}(e_j \mid e_i).$$

4.3 View Alignment

Given an action $A$, this section will propose a novel view alignment scheme to find its best viewpoint. This scheme is “dynamic” since it always refines the viewpoint when the actor performs his actions. First $A$ is decomposed into a series of postures, i.e., $A = \{q_0, q_1, \ldots, q_i, \ldots\}$. Then, given a posture $q_i$ in $A$, we want to determine its best viewpoint $v_i$. Let $e_{q_i}^v$ denote the best key posture of $q_i$ at the viewpoint $v$. Then, the problem for determining $v_i$ can be formulated as the form:
\[ v_i = \arg \max_{v \in V} \Pr(e^v | q_i) , \]  

where \( V \) is the collection of all viewpoints and \( \Pr(e | q_i) \) is the posterior probability of \( q_i \) which is similar to the key posture \( e \). Let \( S_v \) denote the set of key postures extracted from the \( v \)th viewpoint. With \( S_v \), \( e^v \) can be searched from the following equation

\[ e^v = \arg \max_{e \in \mathcal{S}_v} \Pr(e | q_i) , \]  

Plugging Eq.(10) into Eq.(9) implies

\[ v_i = \arg \max_{v \in V} \max_{e \in \mathcal{S}_v} \Pr(e | q_i) . \]  

Since the ‘max’ operation is used in Eq.(11), \( e^v \) also satisfies

\[ \Pr(e^v | q_i) = \max_{e \in \mathcal{S}} \Pr(e | q_i) , \]  

where \( \mathcal{S} = \sum_{v \in V} \mathcal{S}_v \). Actually, \( e^v \) is the key posture which best matches \( q_i \) for all key postures in \( \mathcal{S} \). It means that we can determine \( e^v \) first without finding the best view \( v_i \).

Let \( e^v \) denote the best matched key posture of \( q_i \) from \( \mathcal{S} \), i.e.,

\[ e^v = \arg \max_{e \in \mathcal{S}} \Pr(e | q_i) . \]  

After finding \( e^v \), we can define a view function \( \text{viewK}(e) \) to know which view \( e^v \) belongs to, i.e.,

\[ \text{viewK}(e) = v \text{ if } e \in \mathcal{S}_v \text{ for all } \mathcal{S}_v \subseteq \mathcal{S} . \]  

It is noticed that \( \text{viewK}(e) \) can be obtained in advance when \( \mathcal{S} \) is constructed. With \( e^v \), the best view \( v_i \) of \( q_i \) can be obtained from \( \text{viewK}(e^v) \). In what follows, the detailed algorithm for determining the best view \( v_i \) of \( q_i \) is given.
In real cases, when different views are observed, many similar key postures are also collected in $S$ for action analysis (see Fig. 6(a) and (b)), and lead to many false matches. To treat this problem, the temporal consistence between key postures should be considered into view alignment. Let $\phi_{cc}(e,q_i)$ denote the similarity between two postures $e$ and $q_i$, i.e.,

$$
\phi_{cc}(e,q_i) = \exp(-d_{cc}(e,q_i)),
$$

where $d_{cc}(e,q_i)$ is defined in Eq.(2). In addition, let $T(e,q_{i-1},e)$ denote the transition probability between $e_{q_{i-1}}$ and $e$ (defined in Eq.(8)). When both the spatial and temporal similarities are considered, $\Pr(e \mid q_i)$ in Eq.(13) can be further factorized to

$$
\Pr(e \mid q_i) = \Pr(e_{q_{i-1}} \mid q_{i-1})T(e_{q_{i-1}}, e)\phi_{cc}(e,q_i).
$$

Then, Eq. (13) becomes

$$
e_{q_i} = \arg\max_{e \in S} \Pr(e_{q_{i-1}} \mid q_{i-1})T(e_{q_{i-1}}, e)\phi_{cc}(e,q_i).
$$

In order to determine $e_{q_i}$ more accurately, we analyze $q_i$ not only using its current frame but also its previous several frames. Let $T_o$ denote the observation period. When more frames are observed, Eq.(17) can be further factorized to the form:

$$
e_{q_i} = \arg\max_{e \in S} \left[ \Pr(e_{q_{i-T_o}} \mid q_{i-T_o-1}) \prod_{i=T_o}^{i-1} T(e_{q_{i-1}}, e)\phi_{cc}(e,q_i) \right] T(e_{q_{i-1}}, e)\phi_{cc}(e,q_i).
$$

Furthermore, let $F_q(t)$ denote the accumulation of scores before time $t$, i.e.,
\[ F_q(t) = \Pr \left( q_{t-1} \mid q_{t-1}, \ldots, q_0 \right) \prod_{\mu=t}^{t-1} T(e_{\mu}, e_{\mu}) \phi_{cc}(e_q, q_t). \]  

(19)

It is noticed that \( F_q(t) \) is known in advance at time \( t-1 \). Then, we can obtain \( e_q \) by solving the equation:

\[ e_q = \arg \max_{e \in S} F_q(t) T(e_{\mu}, e) \phi_{cc}(e, q_t). \]  

(20)

The Viterbi algorithm [20] is a suitable tool for solving \( e_q \). Like Fig. 7, the vertical axis records each key posture and the horizontal axis records the query postures changing at different time. Each node \((i,t)\) keeps three values: the matching score (in black) between the \(i\)th key posture and the query posture \(q_t\), the best score \(F_q(t)\) (in red) along a path up to node \((i,t)\), and the previous node on this path. Every link between two nodes has its own transitional probability (the blue dotted line indicates \(T(3,4)\) for example). When the last query posture has been processed, the node with the maximum path score is found and returns the desired key posture \(e_q\). Once \(e_q\) is determined, its corresponding view can be found via \(\text{viewK}(e_q)\). Like Fig. 4, the best key posture is found at the view \(0^0\), its corresponding set of key postures at the view \(90^0\) will be used for action analysis.

4.4 Fast View Alignment Using Correspondence Table

In Eq.(20), given \(q_t\), its best key posture \(e_q\) can be selected from \(S\). If \(S\) is large, the searching task will be very time-consuming. This section will build a correspondence table to filter out impossible posture matches in advance. Then, only few candidates should be scanned to obtain \(e_q\) more efficiently from \(S\).

For any two key postures \(e_i\) and \(e_j\) in \(S\), if their similarity is large enough, they will form a correspondence. Since \(S\) is known before view alignment, we can check whether the similarity between \(e_i\) and \(e_j\) is larger enough. Then, a correspondence table \(C_S\) can be constructed to record this relationship, where
\[ C_S(e_i, e_j) = \begin{cases} 1, & \text{if } \phi_{cc}(e_i, e_j) > \tau \text{ and } i \neq j, \\ 0, & \text{otherwise.} \end{cases} \text{ for all } i, j = 1, \ldots, |S|, \]  

(21)

Here \( \phi_{cc}(e_i, e_j) \) is defined in Eq.(15) and \( \tau \) is a threshold (defined as 0.6 in this paper).

Then, given a key posture \( e \), a subset \( S_e \) can be built from \( S \) as follows:

\[ S_e = \{ e_i | e_i \in S \text{ and } C_S(e, e_i) = 1 \}. \]  

(22)

Each element \( e_i \) in \( S_e \) is a possible correspondence of \( e \). In (20), both the spatial similarity \( \phi_{cc}(e, q_i) \) and the transition probability \( T(e_{q_{t-1}}, e) \) are incorporated together to find the optimal solution \( e_{q_t} \). To speed up the matching efficiency, without considering the temporal consistence, the best match \( \tilde{e}_{q_t} \) of \( q_t \) is first searched from \( S \) using its centroid context, i.e.,

\[ \tilde{e}_{q_t} = \arg \max_{e_i \in S} \phi_{cc}(q_{t-1}, e_i). \]

Then, only the elements in \( S_{\tilde{e}_{q_t}} \) are needed for searching the best solution \( e_{q_t} \) from Eq.(20). Thus, to obtain \( e_{q_t} \), Eq.(20) can be rewritten as follows:

\[ e_{q_t} = \arg \max_{e_i \in S_{\tilde{e}_{q_t}}} F_q(t) \cdot T(e_{q_{t-1}}, e) \cdot \phi_{cc}(e, q_t). \]  

(23)

Since the candidates in \( S_{\tilde{e}_{q_t}} \) are few, \( e_{q_t} \) can be obtained from Eq.(23) more efficiently than Eq.(20). In addition to the efficiency improvement, this table can filter out most of false matches and thus also improve the accuracy of view alignment.

5. Behavior Analysis Using Time-Order Similarity Distributions

After view alignment, each action will be converted to a series of ordered symbols for action analysis. Details of the conversion work are described in Section 5.1. However, there will be many converting errors. To treat this problem, in Section 5.2, a novel representation scheme will be proposed to recognize actions more accurately based on their time-order similarity distributions.

5.1 Action Representation Using Strings
For each posture $q_i$ in an action $A$, the view alignment technique is first adopted to find its best view $v_i$ by using Eq.(23). Then, with $v_i$, $q_i$ will be converted into an action primitive in the standard view $v_s$. In our terminology, the action spaces in $v_i$ and $v_s$ are represented by $S_{v_i}$ and $S_{v_s}$, respectively. In addition, each key posture $e_{v_i}^v$ in $S_{v_i}$ and its corresponding key posture $e_{v_s}^v$ in $v_s$ share the same index. Then, for $q_i$, its key posture $e_{v_i}^v$ in $v_i$ and its key posture $e_{v_s}^v$ in $v_s$ can be associated with the form:

$$e_{v_i}^v = S_{v_i}[id_{v_i}(e_{v_i}^v)], \tag{24}$$

where $id_{v_i}(e_{v_i}^v)$ denotes the index of $e_{v_i}^v$ in $S_{v_i}$. With Eq.(24), $A$ can be then converted into a string as the form

$$A = \{e_{v_i}^v, ..., e_{v_i}^{v_l}, ..., e_{v_i}^{v_{l+s}}\}, \tag{25}$$

where $|A|$ denotes the length of $A$. Due to noise, there would be various conversion errors which often lead to the failure of action recognition. In Section 5.2, a novel scheme to recognize actions using their time-order similarity distributions will be proposed to treat this problem.

**5.2 Action Analysis Using Time-Order Similarity Distributions**

To reduce the effect of converting errors, this paper proposes a novel representation scheme to code a posture not only using its best matched symbol but also using other unmatched symbols. Fig. 8 shows the details of our scheme to match two actions. First, similar to the string representation, the view alignment technique is used for view mapping. Then, given a posture $q_i$ in $A$, we first use Eq.(24) to find its best matched key posture $e_{v_i}^v$ in $S_{v_i}$. In real cases, there must be some key postures in the database which are also very similar to $q_i$. Thus, the correct type $e_{v_i}^v$ cannot be always found. To tackle this ambiguity, we represent $q_i$ using not only $e_{v_i}^v$ but also other key postures in $S_{v_i}$ to form a time-order similarity
distribution (TOSD) matrix for action analysis.

Given a key posture $e_i$ in $S_{v_i}$, we use $h^A_i(i)$ to denote the normalized similarity between $e_i$ and $e^A_{q_i}$, i.e.,

$$h^A_i(i) = \frac{1}{|\phi_{cc}|} \phi_{cc}(e^A_{q_i}, e_i),$$

where $|\phi_{cc}| = \sum_{e_s \in S_{v_i}} \phi_{cc}(e^A_{q_i}, e_s)$. Then, a feature vector $h^A_i$ to represent $q_i$ is formed, i.e.,

$$h^A_i = (h^A_i(0), ..., h^A_i(i), ..., h^A_i(|S_{v_i}| - 1))^T,$$  \hspace{1cm} (26)

where $|S_{v_i}|$ is the size of $S_{v_i}$. $h^A_i$ is a column vector which keeps the distribution of different similarities between $q_i$ and postures in $S_{v_i}$. Even though the best key posture is wrongly matched, the higher similarity between the posture and $q_i$ is still kept in $h^A_i$. Thus, $h^A_i$ can forms a robust feature to describe $q_i$. After browsing all postures in $A$, a TOSD matrix $H_A$ is constructed to describe $A$ more robustly, where

$$H_A = \{h^A_i\}_{i=0, ..., |A|-1}. \hspace{1cm} (27)$$

The dimension of $H_A$ is $|A| \times |S_{v_i}|$. Like Fig. 9, a TOSD matrix $H_A$ is used to represent a “squatting” action $A$. The horizontal and veridical axes denote the time index and the index of key posture, respectively. $H_A[i, i]$ records the similarity $\phi_{cc}(e^A_{q_i}, e_i)$ between $e^A_{q_i}$ and the $i$th key posture $e_i$ in $S_{v_i}$.

To compare a query action $A$ and an action type $D$ in the database, we first convert them to two TOSD matrices $H_A$ and $H_D$, respectively. For the $i$th element $h^A_i$ in $H_A$ and the $j$th element $h^D_j$ in $H_D$, the KL distance [30] is used to measure their dissimilarity:

$$\text{cost}(i, j) = \sum_{k=0}^{\text{max}(i,j)-1} h^A_i(k) \left( \log \frac{h^A_i(k)}{h^D_j(k)} \right).$$  \hspace{1cm} (28)

Let $\xi_{A,D}(i, j)$ denote the cost between the two subsequences $\{h^A_i\}_{i=0, ..., j}$ and $\{h^D_j\}_{i=0, ..., j}$. 
\( \xi_{A,D}(i, j) \) can be recursively calculated by the form:
\[
\xi_{A,D}(i, j) = \min(\xi_{A,D}(i-1, j) + \text{cost}(i, j), \xi_{A,D}(i, j-1) + \text{cost}(i, j), \xi_{A,D}(i-1, j-1) + \text{cost}(i, j)).
\] (29)

Based on Eq.(29), the distance between \( A \) and \( D \) is measured as follows:
\[
\xi_{A,D}(|A|,|D|),
\] (30)
where \( |A| \) and \( |D| \) are the numbers of postures in \( A \) and \( D \), respectively.

6. Experimental Results

To analyze the performances of our proposed scheme, a real-time system to analyze different action events from different views was implemented. Fig. 10 shows a snap of our proposed system for analyzing various actions from different views under different lighting conditions. In this figure, the left-hand side shows a walking sequence, where the right-hand side shows a set of pre-determined key postures used for view alignment and action analysis. The \( v \)th row records the set of key postures captured from the \( v \)th view, while the first row corresponds to the standard view. The key posture surrounded the yellow rectangle matches the current input posture. Then, the current viewpoint can be found for view alignment and then each input action can be represented by a TOSD matrix. With the TOSD matrices, different actions can be more robustly and effectively matched from various viewpoints.

To well represent an input action, we divide the viewing circle to seven orientation, \( i.e., \) front, right, left, front-right, front-left, rear-right, and rear-left. Fig. 11 shows the seven views used for observing actors when a shooting action was performed. Five action types were collected in this paper, \( i.e., \) walking, squatting, shooting, picking up, and waving hand (see Fig. 12). Each action type was performed by six actors twice; thus there were 60 action sequences collected for each view. The actor can freely change his positions and orientations to the camera. The dimension of video frame is \( 320 \times 240 \) pixel elements.
Table 1: Accuracy analyses of view alignment among different action types and view angles using only one frame ($T_0 = 1$).

<table>
<thead>
<tr>
<th>Types</th>
<th>Walking</th>
<th>Shooting</th>
<th>Squatting</th>
<th>Picking Up</th>
<th>Waving</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>View 135°</td>
<td>0.83</td>
<td>0.81</td>
<td>0.79</td>
<td>0.79</td>
<td>0.82</td>
<td>0.808</td>
</tr>
<tr>
<td>View 90°</td>
<td>0.90</td>
<td>0.87</td>
<td>0.84</td>
<td>0.91</td>
<td>0.76</td>
<td>0.856</td>
</tr>
<tr>
<td>View 45°</td>
<td>0.83</td>
<td>0.85</td>
<td>0.81</td>
<td>0.86</td>
<td>0.82</td>
<td>0.834</td>
</tr>
<tr>
<td>View 0°</td>
<td>0.78</td>
<td>0.74</td>
<td>0.74</td>
<td>0.77</td>
<td>0.89</td>
<td>0.784</td>
</tr>
<tr>
<td>View -45°</td>
<td>0.85</td>
<td>0.82</td>
<td>0.82</td>
<td>0.84</td>
<td>0.81</td>
<td>0.828</td>
</tr>
<tr>
<td>View -90°</td>
<td>0.88</td>
<td>0.86</td>
<td>0.85</td>
<td>0.89</td>
<td>0.75</td>
<td>0.846</td>
</tr>
<tr>
<td>View -135°</td>
<td>0.82</td>
<td>0.83</td>
<td>0.76</td>
<td>0.83</td>
<td>0.82</td>
<td>0.842</td>
</tr>
<tr>
<td>Average</td>
<td>0.841</td>
<td>0.825</td>
<td>0.801</td>
<td>0.841</td>
<td>0.81</td>
<td>0.824</td>
</tr>
</tbody>
</table>

To evaluate the performance of view alignment, the correct viewing angles of each testing action sequence were manually labeled. It is noticed that the labeling information is only used for performance evaluation. Then, the estimated view angle of each posture in the testing database can be compared and analyzed with the labeling results. In real cases, different frame periods (see Eq.(19)) will affect the accuracy of view alignment. Table 1 shows the accuracy analyses among different action types and viewing angles when only one frame period ($T_0 = 1$ in Eq.(19)) was adopted. Since many similar key postures were included in the action space, some false matches were unavoidably found when only one frame was used. The lowest accuracy was got from the ‘squatting’ action type since its contours are similar to the ‘picking-up’ type. For the waving action type, some of its posture contours are almost similar if they were not observed from the front view. Thus, the lower accuracy was also got from this type. For the front view (the viewing angle=0°), due to the problem of hand occlusions, many posture contours look similar. Like Fig. 13, the persons’ contours are similar but with different hand gestures. When different viewpoints were compared, the lowest accuracy was found from the front view.

Table 2 lists the accuracy of view alignment when a longer observation period was used, i.e., $T_0 = 3$. Since more postures were used for voting, the accuracy of the “$T_0 = 3$” case is higher than the “$T_0 = 1$” case. Larger accuracy improvements can be found from the ‘walking’ and ‘shooting’ action types. Noticeably, when an actor is observed from a sided
view (view angles at 90° or -90°), his hand gestures will cause more posture changes than the front view. Thus, higher accuracies of view alignment were gained from the viewing angles 90° and -90°. Table 3 shows the average accuracies of view alignment at different views with different observation periods. A longer period is observed, a higher accuracy is obtained. However, a longer period also means less efficient. It is noticeable that when $T_0$ is larger than 7, the accuracy is almost not changed. After trading off between the accuracy and efficiency, the suggested period $T_0$ is 7 and used for all the following experiments.

Table 2: Accuracy analyses of view alignment using three frames ($T_0 = 3$) among different action types and view angles.

<table>
<thead>
<tr>
<th>Types</th>
<th>Walking</th>
<th>Shooting</th>
<th>Squatting</th>
<th>Picking-Up</th>
<th>Waving</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>View 135°</td>
<td>0.89</td>
<td>0.87</td>
<td>0.81</td>
<td>0.85</td>
<td>0.85</td>
<td>0.854</td>
</tr>
<tr>
<td>View 90°</td>
<td>0.93</td>
<td>0.91</td>
<td>0.91</td>
<td>0.93</td>
<td>0.79</td>
<td>0.894</td>
</tr>
<tr>
<td>View 45°</td>
<td>0.90</td>
<td>0.89</td>
<td>0.84</td>
<td>0.89</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>View 0°</td>
<td>0.84</td>
<td>0.81</td>
<td>0.77</td>
<td>0.78</td>
<td>0.93</td>
<td>0.826</td>
</tr>
<tr>
<td>View -45°</td>
<td>0.89</td>
<td>0.86</td>
<td>0.86</td>
<td>0.89</td>
<td>0.88</td>
<td>0.876</td>
</tr>
<tr>
<td>View -90°</td>
<td>0.95</td>
<td>0.90</td>
<td>0.91</td>
<td>0.93</td>
<td>0.79</td>
<td>0.896</td>
</tr>
<tr>
<td>View -135°</td>
<td>0.88</td>
<td>0.88</td>
<td>0.83</td>
<td>0.86</td>
<td>0.87</td>
<td>0.864</td>
</tr>
<tr>
<td>Average</td>
<td>0.897</td>
<td>0.874</td>
<td>0.847</td>
<td>0.875</td>
<td>0.856</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Table 3: Accuracy analyses of view alignment using different observation periods.

<table>
<thead>
<tr>
<th>Periods</th>
<th>$T_0 = 1$</th>
<th>$T_0 = 3$</th>
<th>$T_0 = 5$</th>
<th>$T_0 = 7$</th>
<th>$T_0 = 9$</th>
<th>$T_0 = 11$</th>
<th>$T_0 = 13$</th>
</tr>
</thead>
<tbody>
<tr>
<td>View 135°</td>
<td>0.82</td>
<td>0.84</td>
<td>0.91</td>
<td>0.94</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>View 90°</td>
<td>0.85</td>
<td>0.90</td>
<td>0.93</td>
<td>0.94</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>View 45°</td>
<td>0.84</td>
<td>0.89</td>
<td>0.92</td>
<td>0.93</td>
<td>0.94</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>View 0°</td>
<td>0.77</td>
<td>0.84</td>
<td>0.90</td>
<td>0.93</td>
<td>0.93</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>View -45°</td>
<td>0.84</td>
<td>0.88</td>
<td>0.92</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>View -90°</td>
<td>0.86</td>
<td>0.90</td>
<td>0.94</td>
<td>0.94</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>View -135°</td>
<td>0.82</td>
<td>0.86</td>
<td>0.90</td>
<td>0.93</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>Average</td>
<td>0.828</td>
<td>0.873</td>
<td>0.917</td>
<td>0.935</td>
<td>0.940</td>
<td>0.942</td>
<td>0.942</td>
</tr>
</tbody>
</table>

Table 4: Accuracy analyses of behavior recognition when the scheme of view alignment is used or not.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Types</th>
<th>Walking</th>
<th>Shooting</th>
<th>Squatting</th>
<th>Picking-Up</th>
<th>Waving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without view alignment</td>
<td>$T=3$</td>
<td>0.79</td>
<td>0.71</td>
<td>0.73</td>
<td>0.75</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>$T=7$</td>
<td>0.89</td>
<td>0.87</td>
<td>0.84</td>
<td>0.87</td>
<td>0.85</td>
</tr>
<tr>
<td>With view alignment</td>
<td>$T=3$</td>
<td>0.90</td>
<td>0.87</td>
<td>0.85</td>
<td>0.86</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>$T=7$</td>
<td>0.95</td>
<td>0.94</td>
<td>0.92</td>
<td>0.94</td>
<td>0.93</td>
</tr>
</tbody>
</table>
After analyzing the task of view alignment, we also need to evaluate the performances of behavior analysis under different conditions. As we know, a posture type changes differently under different viewpoints. If the task of view alignment is not used, many input postures will be wrongly converted. Table 4 lists the accuracy comparisons when the alignment scheme is used or not. Clearly, the accuracy improvement is significant if the alignment scheme is adopted. In addition to view alignment, this paper also uses the property “mirror symmetry” to reduce the size of action space. The following experiment was used to examine the effects on behavior analysis. Table 5 lists the accuracy comparisons when this symmetry property is used or not. The reducing effect will lead to some accuracy loss in view alignment. But, the loss in behavior analysis is mirror. Thus, the mirror property is used for all the following experiments.

Table 5: Performance analysis of behavior recognition when the mirror symmetry is used or not.

<table>
<thead>
<tr>
<th>Types</th>
<th>Methods</th>
<th>Walking</th>
<th>Shooting</th>
<th>Squatting</th>
<th>Picking-Up</th>
<th>Waving</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=3</td>
<td>Without Mirror</td>
<td>0.91</td>
<td>0.89</td>
<td>0.86</td>
<td>0.90</td>
<td>0.89</td>
</tr>
<tr>
<td>T=7</td>
<td>Without Mirror</td>
<td>0.97</td>
<td>0.96</td>
<td>0.93</td>
<td>0.96</td>
<td>0.94</td>
</tr>
<tr>
<td>T=3</td>
<td>With Mirror</td>
<td>0.90</td>
<td>0.87</td>
<td>0.85</td>
<td>0.86</td>
<td>0.87</td>
</tr>
<tr>
<td>T=7</td>
<td>With Mirror</td>
<td>0.95</td>
<td>0.94</td>
<td>0.92</td>
<td>0.94</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Table 6: Efficiency improvement of behavior recognition when the correspondence table is used or not (seconds/frame).

<table>
<thead>
<tr>
<th>Types</th>
<th>Methods</th>
<th>Walking</th>
<th>Shooting</th>
<th>Squatting</th>
<th>Picking Up</th>
<th>Waving</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=3</td>
<td>Without Correspondence</td>
<td>0.172</td>
<td>0.19</td>
<td>0.164</td>
<td>0.192</td>
<td>0.208</td>
<td>0.185</td>
</tr>
<tr>
<td>T=7</td>
<td>Without Correspondence</td>
<td>0.088</td>
<td>0.093</td>
<td>0.084</td>
<td>0.088</td>
<td>0.092</td>
<td>0.089</td>
</tr>
<tr>
<td>Improvement Ratio</td>
<td>1.96</td>
<td>2.0</td>
<td>1.95</td>
<td>2.18</td>
<td>2.26</td>
<td>2.07</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Accuracy analyses of posture matching when the correspondence table is used or not.

<table>
<thead>
<tr>
<th>Types</th>
<th>Methods</th>
<th>Walking</th>
<th>Shooting</th>
<th>Squatting</th>
<th>Picking-Up</th>
<th>Waving</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=3</td>
<td>Without Correspondence</td>
<td>0.81</td>
<td>0.80</td>
<td>0.75</td>
<td>0.79</td>
<td>0.74</td>
<td>0.778</td>
</tr>
<tr>
<td>T=7</td>
<td>Without Correspondence</td>
<td>0.87</td>
<td>0.85</td>
<td>0.84</td>
<td>0.84</td>
<td>0.85</td>
<td>0.850</td>
</tr>
<tr>
<td>T=3</td>
<td>With Correspondence</td>
<td>0.90</td>
<td>0.87</td>
<td>0.85</td>
<td>0.86</td>
<td>0.87</td>
<td>0.870</td>
</tr>
<tr>
<td>T=7</td>
<td>With Correspondence</td>
<td>0.95</td>
<td>0.94</td>
<td>0.92</td>
<td>0.94</td>
<td>0.93</td>
<td>0.936</td>
</tr>
</tbody>
</table>

To speed up the efficiency of behavior analysis, a correspondence table (see Section 5.4) is built for filtering out impossible posture matches. Table 6 shows the efficiency improvement in posture matching if the correspondence table is used or not. The fourth raw
records the improvement ratio of matching efficiency. In the ‘squatting’ type, there are too many similar posture types (caused by the occlusions between body parts). Thus, its efficiency improvement is the worst. The correspondence table can also filter out many false posture matches in advance. Table 7 lists the accuracy comparisons of behavior recognition when this correspondence table is used or not. Clearly, significant improvements in recognition accuracy were gained from this table.

Table 8: Accuracy analysis of behavior recognition when the matrix-based method is used or not.

<table>
<thead>
<tr>
<th>Types</th>
<th>Methods</th>
<th>Walking</th>
<th>Shooting</th>
<th>Squatting</th>
<th>Picking-Up</th>
<th>Waving</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T=3</td>
<td>0.83</td>
<td>0.82</td>
<td>0.75</td>
<td>0.77</td>
<td>0.76</td>
<td>0.786</td>
</tr>
<tr>
<td></td>
<td>T=7</td>
<td>0.89</td>
<td>0.85</td>
<td>0.81</td>
<td>0.85</td>
<td>0.84</td>
<td>0.848</td>
</tr>
<tr>
<td></td>
<td>Matrix-based</td>
<td>0.90</td>
<td>0.87</td>
<td>0.85</td>
<td>0.86</td>
<td>0.87</td>
<td>0.870</td>
</tr>
<tr>
<td></td>
<td>T=3</td>
<td>0.95</td>
<td>0.94</td>
<td>0.92</td>
<td>0.94</td>
<td>0.93</td>
<td>0.936</td>
</tr>
<tr>
<td></td>
<td>T=7</td>
<td>0.95</td>
<td>0.94</td>
<td>0.92</td>
<td>0.94</td>
<td>0.93</td>
<td>0.936</td>
</tr>
</tbody>
</table>

After view alignment, this paper uses a TOSD matrix representation for representing each action. Table 8 lists the accuracy comparisons between the string-based method and the TOSD matrix. In real conditions, each posture in an action sequence cannot be always converted to a correct symbol. Thus, many converting errors are embedded in the string-based representation. As to the TOSD matrix, it represents a posture using not only the best matched key posture but also other similar posture types. Thus, it can provide higher accuracies for behavior recognition than the string-based scheme. From Table 8, clearly, large accuracy improvements were gained from the TOSD matrix.

Table 9: Accuracy analyses of behavior recognition among different viewing angles and action types.

<table>
<thead>
<tr>
<th>Views</th>
<th>Types</th>
<th>Walking</th>
<th>Shooting</th>
<th>Squatting</th>
<th>Picking Up</th>
<th>Waving</th>
</tr>
</thead>
<tbody>
<tr>
<td>View 135</td>
<td>0.94</td>
<td>0.93</td>
<td>0.90</td>
<td>0.93</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>View 90</td>
<td>0.98</td>
<td>0.97</td>
<td>0.94</td>
<td>0.93</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>View 45</td>
<td>0.95</td>
<td>0.95</td>
<td>0.91</td>
<td>0.95</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td>View 0°</td>
<td>0.92</td>
<td>0.92</td>
<td>0.95</td>
<td>0.94</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>View -45</td>
<td>0.95</td>
<td>0.95</td>
<td>0.91</td>
<td>0.95</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td>View -90</td>
<td>0.98</td>
<td>0.97</td>
<td>0.94</td>
<td>0.93</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>View -135</td>
<td>0.95</td>
<td>0.93</td>
<td>0.92</td>
<td>0.94</td>
<td>0.95</td>
<td></td>
</tr>
</tbody>
</table>

Table 9 lists the average accuracy of behavior recognition among different viewing
angles. When different action types were compared, the lowest accuracy was found from the ‘squatting’ action type since postures in this type are very similar to the ‘picking-up’ type. For the front view, due to the occlusion problem, two postures sometimes have similar contours even though their corresponding event types are different (like shooting and squatting events). Thus, when different viewing angles were compared, we got the worst accuracy from the front view. Table 10 shows the confusion matrix of behavior behaviors among different action events. Similar to Table 9, the lower accuracy was got from the ‘squatting’ type than other action types. The average accuracy of our system is 94%.

Table 10: Confusion matrix for behavior analysis.

<table>
<thead>
<tr>
<th>Types</th>
<th>Analysis Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Walking</td>
</tr>
<tr>
<td>Walking</td>
<td>0.95</td>
</tr>
<tr>
<td>Shooting</td>
<td>0.03</td>
</tr>
<tr>
<td>Squatting</td>
<td>0</td>
</tr>
<tr>
<td>Picking Up</td>
<td>0.01</td>
</tr>
<tr>
<td>Waving</td>
<td>0</td>
</tr>
</tbody>
</table>

In addition to our database, we also tested our method on a public dataset, i.e., the CMU motion of body (MoBo) database [32]. It includes twenty-five actors performing four different types of walking: slow walk, fast walk, inclined walk, and slow walk holding ball. The four walking types are shown in Fig. 14. Six views were used to capture the MoBo database (see Fig. 15). The confusion matrix for analyzing the accuracy of view alignment is shown in Table 11. In the front and back views (view angles at 0° and 180°), the contours of the same posture type look very similar. The similar posture contours make our method confused in classifying the two view angles to correct ones. Except these two view angles, the accuracies of our alignment scheme in all other views are still higher than 95%. Table 11 shows the accuracy analysis of behavior recognition among different types. The average accuracy of our system on this data set is 94.07%. Even though these actions are captured from different views, our method still well recognize them. All the above experiments have proved that the proposed method is a robust, accurate, and powerful tool for
behavior recognition from arbitrary views.

Table 11: Confusion matrix of view alignment among different viewing angles.

<table>
<thead>
<tr>
<th>Degrees</th>
<th>-90$^\circ$</th>
<th>-45$^\circ$</th>
<th>0$^\circ$</th>
<th>90$^\circ$</th>
<th>45$^\circ$</th>
<th>180$^\circ$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-90$^\circ$</td>
<td>96.2%</td>
<td>3.8%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-45$^\circ$</td>
<td>2.55%</td>
<td>97.45%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0$^\circ$</td>
<td>0</td>
<td>0</td>
<td>64.4%</td>
<td>0</td>
<td>0</td>
<td>35.6%</td>
</tr>
<tr>
<td>90$^\circ$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>97.54%</td>
<td>2.46%</td>
<td>0</td>
</tr>
<tr>
<td>45$^\circ$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.23%</td>
<td>97.77%</td>
<td>0</td>
</tr>
<tr>
<td>180$^\circ$</td>
<td>0</td>
<td>0</td>
<td>59.5%</td>
<td>0</td>
<td>0</td>
<td>60.5%</td>
</tr>
</tbody>
</table>

Table 12: Confusion matrix of action type recognition.

<table>
<thead>
<tr>
<th>Types</th>
<th>Ball</th>
<th>Fast-walk</th>
<th>Incline</th>
<th>Slow-walk</th>
</tr>
</thead>
<tbody>
<tr>
<td>ball</td>
<td>95.55%</td>
<td>0.24%</td>
<td>1.46%</td>
<td>2.75%</td>
</tr>
<tr>
<td>Fast-walk</td>
<td>0.23%</td>
<td>94.64%</td>
<td>2.6%</td>
<td>2.53%</td>
</tr>
<tr>
<td>Incline</td>
<td>0.17%</td>
<td>0.43%</td>
<td>92.65%</td>
<td>6.76%</td>
</tr>
<tr>
<td>Slow-walk</td>
<td>0.33%</td>
<td>0.72%</td>
<td>5.53%</td>
<td>93.42%</td>
</tr>
</tbody>
</table>

8. Conclusions

This paper has presented a novel behavior analysis system to recognize human actions from different views. The contributions of this paper are summarized as follows:

(a) A novel behavior recognition scheme was proposed to recognize human actions from arbitrary views without using any stereo or 3-D data.

(b) A novel sampling technique was proposed to construct a more compact action space so that the view invariance of action event can be maintained.

(c) A property of mirror symmetry was used to reduce the whole view space to only its quarter from which different actions can be represented more efficiently.

(d) A novel view alignment scheme was proposed to search the best viewpoint for recognizing an event even though different viewpoints are adopted.

(e) A correspondence table was proposed to filter out impossible candidates in advance.

(f) A novel TOSD matrix was proposed to more robustly and accurately match two action events even though they are captured from arbitrary views.

Experimental results have proved the superiority of our proposed system in classifying human behaviors from different views.
References


High Lights

- We propose a novel scheme for view-changeable action event analysis.
- A view alignment scheme is proposed for action analysis around a view circle.
- A property of mirror symmetry is proposed for reducing the whole view space.
- A novel time-order similarity distribution matrix is proposed for robust event analysis.
Figures

(a) Training stage.

(b) Recognition stage.

Fig. 1 Flowchart of the proposed system for recognizing behaviors from arbitrary views. (a) Training stage. (b) Recognition stage.

Fig. 2 (a) A view circle used to observe an actor. (b) Mirror symmetry of a “standing” posture (observed at $45^\circ$ and $-135^\circ$, respectively). (c) The action space is constructed by sampling the view angles ranging from $-90^\circ$ to $90^\circ$.

Fig. 3 (a) Mirror symmetry of the “walking” posture happening between the view angles $-30^\circ$ and $30^\circ$. (b) Mirror symmetry of the “squatting” posture between the view angles $-30^\circ$ and $30^\circ$. 
Fig. 4 View alignment from the view angle 0° to a specific view.

Fig. 5 Matching results using the width-to-height ratio constraint. (a) Input query posture. (b) and (c): Results of posture matching without/with the ratio constraint.

Fig. 6 Similar postures extracted from different actions. (a) “Walking”. (b) “Shooting”.

Fig. 7 The Viterbi algorithm.
Fig. 8: Flowchart of the action analysis system.

Fig. 9: A time-order similarity distribution matrix $H_A$ is constructed for representing a “squatting” action $A$.

Fig. 10: A snapshot of our proposed system to analyze various actions from different views.

Fig. 11: Seven views used to observe a shooting action. (a) Right View. (b) Left View. (c) Front-right view. (d) Front-left view. (e) Rear-right view. (f) Rear-left view. (g) Front view.
Fig. 12: Five action types used for performance evaluation. (a) Walking. (b) Squatting. (c) Shooting. (d) Rucking Up. (e) Waving.

Fig. 13: Different posture types but with similar contours. (a) and (b): Shooting. (c) and (d): Standing.

Fig. 14: Four action types used for performance evaluation. (a) Ball. (b) Fast-walk. (c) Incline. (d) Slow-walk.
Fig. 15: Six views used to observe actors. (a) Right View. (b) Left View. (c) Front-right view. (d) Front-left view. (e) Rear-right view. (f) Rear-left view.