Computer Vision and Image Understanding 115 (2011) 177–193. Elsevier.
DOI:10.1016/j.cviu.2010.11.012.

Robust Sequence Alignment for Actor-Object Interaction Recognition: Discovering Actor-Object States

Roman Filipovych and Eraldo Ribeiro*

Computer Vision and Bio-inspired Computing Laboratory,
Florida Institute of Technology,
Department of Computer Sciences,
Melbourne, Florida, U.S.A.

Abstract

In this paper, we address the problem of recognizing atomic human-object interactions from videos. Our method is based on the observation that, at the moment of physical contact with the object, both the motion and appearance (i.e., shape) of the interacting person are constrained by the target object. We introduce the concept of *actor-object states* as the instantaneous configuration of actor and object that usually corresponds to the moment of physical contact. We argue that the information content in frames belonging to the actor-object states is descriptive of the specific interaction. We use the actor-object state concept to propose an approach in which human-object interactions are represented by a combination of image patches and velocity information extracted along tracked body-point trajectories. However, determining the set of video frames corresponding to actor-object states is challenging as, before and after physical contact, human motion and appearance may vary significantly for the same interaction type. We address this issue by means of a robust sequence-matching algorithm that discovers actor-object states by matching pairs of misaligned sequences of features. We then show how these discovered actor-object states can be used for the recognition of basic interactions with objects. Finally, we evaluate the proposed concept on classification tasks performed on a new dataset of atomic human-object interactions.

Key words: activity analysis, human-object interaction, sequence matching, dynamic programming

1. Introduction

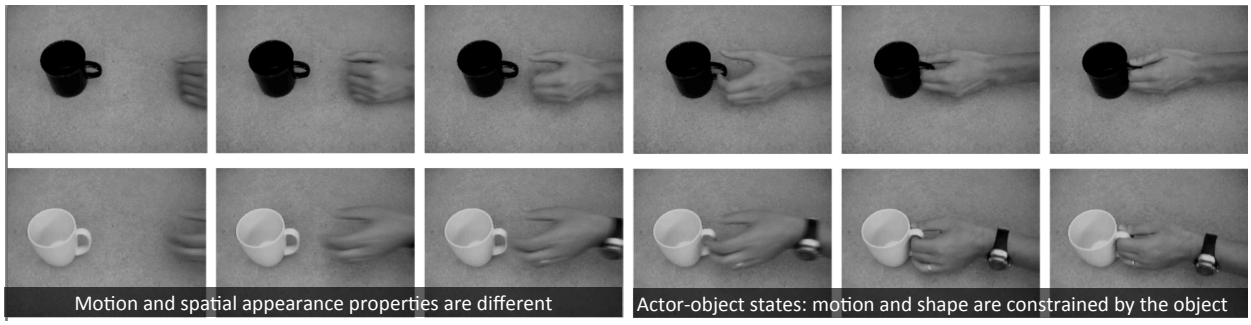
Advancing automatic human-action recognition methods is of relevance to both the scientific and industrial communities. Applications of action recognition algorithms include surveillance, video indexing and summarization, etc. However, despite significant efforts by the computer-vision community, human-action recognition is still an open problem. Even for scenarios with static single-view camera, spatio-temporal variations in actions performed by different individuals make action recognition a challenging task. Additionally, the presence of background clutter introduces irrelevant information that is hard to be isolated.

In this paper, we focus on the problem of atomic (i.e., simple and short-duration) human-object interactions. This is a different problem from that of traditional action recognition. In fact, for videos containing interactions with objects, motion cues alone may not be sufficient to achieve higher-level reasoning about the underlying activities. Consider for example the "grasp a fork" and "grasp a spoon" interactions shown in Figure 1(a). While the motions performed by the hands are essentially the same, the static appearance of the two target objects has to be considered to disambiguate the interactions. On the other hand, interactions such as "grasp a fork" and "touch a fork" cannot be classified based on the object's appearance alone. Thus, it is natural to expect that recognition approaches that use various types of video information (e.g., motion, structure, shape, etc) would perform better than, or at least at the same level of, methods based on single information type.

Considering atomic interactions such as the ones depicted in Figure 1, an interesting but relatively unexplored aspect in computer vision is how the person's perceived motion and shape appearance are constrained by the target object during the interaction. However, when different actors perform the same interaction activity, their motions may differ considerably. Similar uncertainty occurs for the spatial appearance properties of the actors. Spatial appearance in this case represents the spatial configuration or shape of the body part interacting with the object as well as the object's appearance itself. For example, in the "grasp a cup" activity in Figure 1(b), the hands approach the cups at different speeds and have different spatial properties (e.g., clutched, in the black-cup sequence, and slightly open, in the white-cup sequence). However, at the moment of physical contact, the actors' motions, appearances, and actor-object spatial configurations become constrained by the target object (last two frames on the right-hand side of Figure 1(b)). We argue that the information content in these constrained motion and spatial configurations of both actor and object is descriptive of the specific interaction. We refer to these instantaneous appearances and joint actor-object configurations as *actor-object states* [1]. Indeed, actor-object states can be observed at different levels of

* *Corresponding author: eribeiro@cs.fit.edu (E. Ribeiro)*

(a) Insufficiency of single information source for recognition of actor-object interactions.



(b) Constrained actor-object states.

Figure 1: (a) Motion information alone may not be enough to discriminate between the "grasp a fork" and "grasp a spoon" interactions. On the other hand, object information is not relevant for discrimination of the "grasp a fork" and "touch a fork" interactions. (b) On physical contact, the actors' motions, appearances, and actor-object spatial configurations are constrained by the target object.

detail (e.g., entire human body, body parts, etc.). For example, consider the "push a car" interaction in Figure 2. The plot in the figure shows the vertical velocity values extracted along the trajectories of the point finger. Notice how these values become constrained during the time of physical contact. This example suggests that actor-object states can be considered at both the entire body (e.g., Figure 1(b)) and individual body parts levels (e.g., Figure 2).

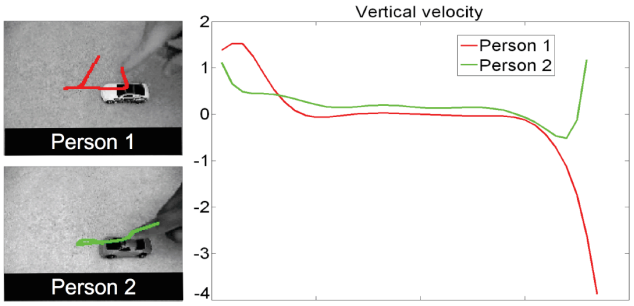


Figure 2: Vertical velocities for tracked finger tips of two persons pushing a toy car. The velocity curves coincide at the moment of physical contact.

In this paper, we present a method for recognizing atomic human-object (actor-object) interactions from single-view cameras. Our goal is to bridge the gap in the computer-vision literature between lower-level action recognition and higher-level interaction analysis. Our method is based on the observation that, at the moment of physical contact, the actors' motions and appearances are constrained by the target object. We propose a deterministic approach where human-object interactions are represented by a combination of image patches and velocity information extracted along tracked body-point trajectories. Here, an actor-object state is considered to be an instantaneous configuration of the object and an individual body part. Discovering the location of the actor-object states is a challenging task. We propose a robust sequence-matching algorithm that discovers the actor-object states as short-term constrained interaction patterns by matching pairs of misaligned sequences of features. The method iteratively removes subsequences that do not represent the actor-object state pattern. Finally, a distance measure between sequences is computed based on the point-correspondences assignment obtained for the discovered patterns. We introduce a novel actor-object interactions dataset, and show that actor-object states can effectively represent interactions with objects. Our contributions in this paper are summarized as follows:

Contributions

- We introduce the concept of actor-object states. The visual information in frames belonging to these states are descriptive instantaneous configurations of actor and object that usually correspond to the moment of physical contact with the object. We argue that actor-object states carry important visual information about underlying interactions.

- We develop a robust sequence-matching method that tolerates significant amount of noise in video sequences. This method is based on a dynamic programming procedure for robust alignment of time series. We propose a subsequence pattern-discovery algorithm that extracts actor-object states at the detail level of individual body parts. Extract pattern subsequences are then used for interaction recognition.

- We propose a solution to the problem of recognizing prim-

itive interactions between human and objects. Our method combines motion and shape appearance information extracted from videos. Our experiments are performed on a new actor-object interactions database that is made publicly available from the authors' website: `http://www.cs.fit.edu/~eribeiro/actor_object_interactions/`

The remainder of this paper is organized as follows. In Section 2, we comment on the related literature. In Section 3, we introduce our interaction-representation approach. We describe our interaction learning algorithm in Section 4. In Section 5, we present our experimental results. Finally, conclusions are presented in Section 6.

2. Related Work

In general, activity analysis can be performed at two different levels of granularity. The first of these is the low-level activity recognition. This level focuses on the analysis of atomic events (e.g., "walking", "pushing a toy car"). The second level is the higher-level activity analysis. Here, it is often assumed that a series of low-level events have been previously detected. The higher-level analysis is usually semantic-based and aims at recognizing complex activities such as "greeting a person" or "preparing a toast". Despite various attempts by the computer vision community to model complex human activities [2, 3, 4, 5, 6, 7], limited attention has been dedicated to human-object atomic interaction recognition. As a result, higher-level activity recognition approaches tend to be based on information regarding primitive interactions that is either provided manually, by means of reference points [5, 8] or by using tag data [2]. At the same time, some recent works suggest that combination of motion information with the information about the context in which an activity is happening improves recognition [9].

Lower-level activity recognition is often named action recognition, focusing mainly at recognizing primitive activities. Related approaches can be grouped into data-driven and model-based methods. Data-driven approaches operate directly on the data. For example, Dollar et al. [10] classify actions in the space of extracted spatio-temporal features using a support-vector machine classifier. Leo et al. [11] describe an unsupervised clustering algorithm for motion classification using histograms of binary silhouette's horizontal and vertical projections. These methods are computationally efficient and achieve good classification performance. However, data-driven methods may be inadequate in scenarios where local image features are highly ambiguous. On the other hand, model-based approaches explicitly include higher-level knowledge about the data. Despite their computational and mathematical elegance, the performance of model-based approaches strongly depends on both the choice of model and the availability of prior information about the data at hand. Graphical models [12, 13] represent a suitable solution to this problem as they allow for efficient learning and inference techniques while simultaneously providing a span of models with rich descriptive power.

Interest in the problem of recognizing primitive interactions between humans and objects has increased in the past five years. An effort in this direction was made by Gupta et al. [14]. They present a Bayesian approach that unifies the inference processes involved in object recognition and action recognition. The method simultaneously estimates object type, location, movement segments, and the effect of movements on objects. However, interactions here are limited to those that can be described in terms of a sequence of reach motion, trajectory-like manipulation motion, and object reaction. Similar interaction structure was considered by Kjellstrom et al. [15]. Peursum et al. [16] suggest the importance of action understanding in object recognition tasks. In their object recognition method, they use human activity to infer both the location and identity of objects without any shape analysis. Similarly, Moore et al. [17] designed a system where Hidden Markov models of human motion are combined with object information within a Bayesian action recognition system. While this early work is limited to object information being manually provided in the form of bounding regions, it is consistent with results obtained by Gupta et al. [14], and suggests that both motion and object information should guide interaction recognition. The idea that action information improves object recognition is also supported by studies using objects' functional parts [18, 19, 20].

Video-based analysis of human activities is often performed using trajectories of tracked objects (e.g., body parts, vehicles, etc.) [21, 22]. Related learning and inference methods require reliable measures of trajectory similarity. In this case, trajectories can be treated as sequential data that can be analyzed by pattern-matching methods such as Hidden-Markov Models (HMM) and Dynamic-Time Warping (DTW). HMM is a powerful probabilistic time-series representation commonly used in motion recognition approaches [23, 24]. Another popular sequence-matching approach used in human-motion recognition is the Dynamic-Time Warping (DTW) algorithm [25, 26]. DTW provides a way to match temporally misaligned sequences using dynamic programming [27]. Sensitivity to abrupt changes in amplitude is a key problem in DTW-based sequence matching. A less sensitive sequence alignment method is the Longest Common Subsequence (LCSS) algorithm [28].

In this paper, we focus ourselves on the problem of recognizing human-object atomic interactions. We propose an algorithm that, given two misaligned sequences of static features, discovers short-term constrained interactions pattern subsequences that are associated with actor-object states as defined in this paper. The output of our method is expected to serve as an input to higher-level activity recognition algorithms.

3. Actor-object States at the Body-Part Level

We will now make use of the actor-object state concept to develop an algorithm for the recognition of atomic actor-object interactions. We commence by summarizing the actor-object state concept based on the description and example provided in Section 1.

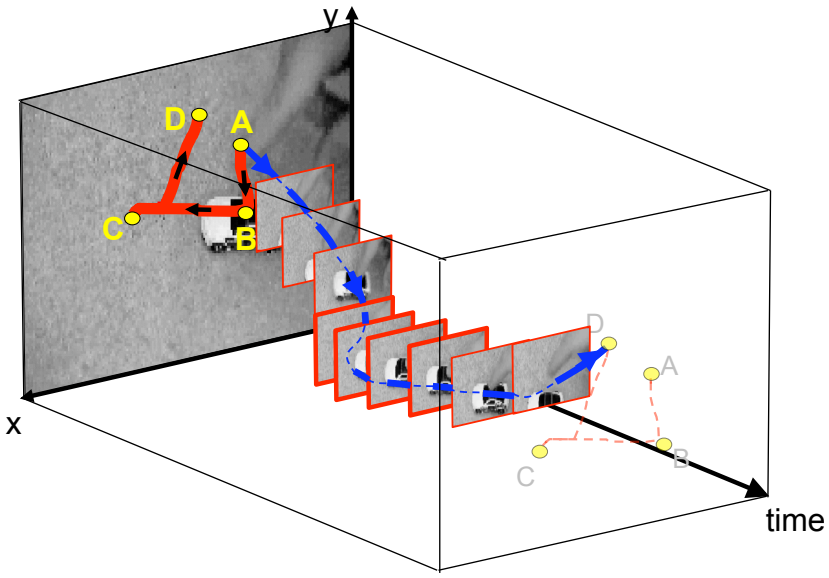


Figure 3: Interaction representation. Blue curve represents a spatio-temporal trajectory (image coordinates + time). Red curve shows the actual tracked trajectory superimposed on a frame of the video sequence (dynamic information).

Actor-Object States – Let us consider a number of different persons (actors) performing the same physical interaction with an object (e.g., touching, manipulating). The actors' motion and spatial appearance configuration (shape) may differ considerably before and after the interaction. However, at the moment of physical contact, the actors' motion and appearance become constrained by the target object. We call this constrained spatio-temporal configuration the *actor-object state*.

In this section, we propose an approach in which human-object interactions are represented by a combination of image patches and velocity information extracted along tracked body-point trajectories. Measurements performed on body-point trajectories provide a good description of the underlying motion. By extracting spatial subregions around each tracked body-point, we can obtain a representation of the appearance information in the video. Thus, a human-object interaction can be represented by two main components. The first component consists of point-trajectory measurements describing the underlying motion. The second component consists of a sequence of local appearance descriptors extracted along the corresponding trajectory.

Let $\mathcal{V} = \{\mathrm{T}_1, \ldots, \mathrm{T}_K\}$ represent a set of body-point trajectories tracked from videos of a human-object interaction. We assume the availability of a set of trajectories obtained by tracking interest points on the body. These trajectories can be obtained using a feature-tracking method [29]. According to the actor-object state concept, trajectories T_i will be constrained during the physical contact with the target object. We argue that subsequences containing static and motion features extracted along the trajectory T_i occurring during these constrained actor-object states are descriptive of the specific interaction type. As a result, a human-object interaction can be represented by the set

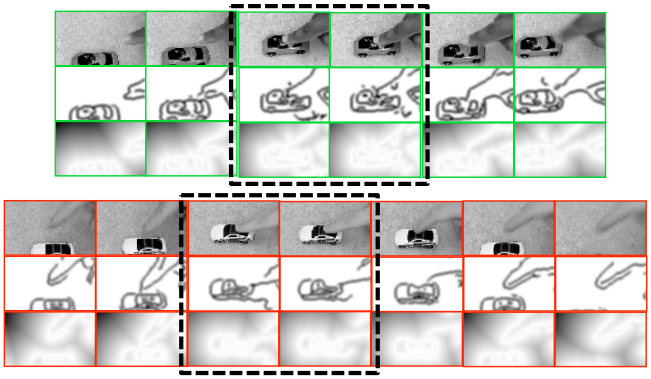


Figure 4: Sequence of extracted patches (appearance). First row: patches; Second row: edge maps; Third row: distance transform of the edge maps. Dashed-line rectangle include frames describing the moment of contact. Number of frames was reduced for illustration.

of feature subsequences $\{\mathcal{T}_i\}$ extracted along the trajectories at the moments of constrained actor-object states, and given by $\mathcal{T}_i = \{(\mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,N_i}), (\mathbf{v}_{i,1}, \ldots, \mathbf{v}_{i,N_i})\}$, where N_i is the number of body part-level actor-object states for the trajectory $\mathcal{T}_i$. Here, $\mathbf{a}_i = (\mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,N_i})$ is the sequence of feature vectors representing the local image appearance, and $\mathbf{v}_i = (\mathbf{v}_{i,1}, \ldots, \mathbf{v}_{i,N_i})$ is the sequence of instantaneous velocities obtained from Bezier approximations of the underlying trajectory. Figure 3 illustrates our actor-object interaction representation. Notice that different trajectories may have different number of actor-object states. This leads to a flexible interaction model as different body parts may be constrained by the target object at different time periods during a single interaction. For simplicity, we assume that the trajectory-correspondence problem is resolved, and the ex-

tracted trajectories are continuous. Additionally, representing the motion-dynamics information with velocities allows for a translation-invariant representation. It is worth pointing out that the feature vector **a** can be obtained in a number of ways [30]. We use the distance transform described in [31] to represent image subregion appearance. Figure 4 shows an example of the sequences of static features extracted along the tracked point-finger trajectories for two individuals performing "push a car" interaction. Frame subregions corresponding to actor-object states are indicated by the dashed-line rectangles.

Our interactions classification method is based on a robust sequence-matching technique. There are two main tasks in our algorithm. First, we discover the subsequences of $\mathcal{T}_i$ that correspond to the actor-object states. Secondly, we classify interactions by determining the similarity between the discovered subsequences. Each of the two components in $\mathcal{T}_i$ is a sequence representing local appearance changes and motion dynamics of the underlying interaction. The information in each sequence is compared individually based on a weighted average of similarity scores obtained from our pattern discovery method described later in this paper. In the description that follows, the subsequences containing actor-object states are the *pattern subsequences* in our approach. Depending on the type of information under consideration, the patterns are subsequences of the motion measurements or appearance descriptors. We will refer to the subsequences that do not contain actor-object states as *non-pattern subsequences*. Next, we present an iterative algorithm for extracting subsequence patterns corresponding to actor-object states.

4. Discovering Actor-Object States Using Robust Sequence Alignment

Our pattern-discovery method is a robust sequence-alignment algorithm based on dynamic programming. Our main goal is to be able to discover actor-object states represented by data sequences extracted from the tracked body-part trajectories as described in Section 3. Sequence alignment has been widely used for human motion recognition [32, 33, 34]. However, in the case of human-object interactions, only limited portions of the sequences will align. These portions usually correspond to the constrained actor-object states, and are the target pattern sequences we aim at discovering with our method. Additionally, real-world tracking data can be distorted by tracking errors, abrupt variations in illumination, and other sources of noise. This will cause significant problems for DTW-based methods as these methods will necessarily assign correspondences to points outside the pattern of interest. We illustrate this problem with an alignment example. Let us consider the pattern described by the two sequences **X** and **Y** shown in Figure 5(a). A good-quality alignment is produced by the DTW algorithm (Figure 5(b)). As an example, we corrupted both pattern sequences with 10% random additive noise and inserted them into arbitrary sequences. The noise distorted parts of the sequence into non-pattern sequences. Resulting sequences are shown in Figure 5(c). The time interval corresponding to the pattern subsequence is shown in the plots' non-shaded central region. Figure 5(d) shows the point-correspondences produced by DTW. Notice how points outside the central pattern of interest, as well as noisy points, are involved in the DTW's warping function.

The data presented in the above example is similar to that obtained for human-object interactions and shown in Figure 2. More specifically, at the moment of physical contact with the object, the tracked trajectories are more likely to become aligned as the actor is constrained by the target object. In this paper, we assume that actor-object states mostly appear in the inside portions of the tracked trajectories. Our pattern-discovery algorithm is divided into two main steps. First, it performs an approximate sequence alignment using a dynamic programming forward-alignment technique. This alignment is achieved while minimizing the inclusion of points outside pattern subsequences (i.e., points in the grey region of Figure 5(c)). In this step, the initial location of the pattern subsequence is determined. In the second step, a backward alignment is performed on the subsequence extracted from the previous step. The algorithm iterates between these two steps until convergence. Details of the steps are provided as follows.

4.1. Step 1 – Approximate Sequence Alignment

4.1.1. Dynamic Time Warping

We begin by describing the traditional dynamic-time warping algorithm (DTW) [25]. The alignment between two time series $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{M}$ and $\mathbf{Y} = \{\mathbf{y}_j\}_{j=1}^{N}$ of lengths M and N, is given by the point-correspondences $W = \{w_k\}_{k=1}^{K}$ (also called warping path), where $\max\{M, N\} \leq K < M + N$. Here, $w_k = (i_k, j_k)$, and i_k and j_k are indices from **X** and **Y**, respectively. The elements of W are chosen to minimize some criterion (e.g., average pairwise point distance). The DTW's warping path is subject to three main constraints: (*i*) every point in both time series is used in the warping path, (*ii*) points in W must be monotonically spaced in time, and (*iii*) the warping path must be continuous. The optimal warping path is the one that minimizes the following warping cost:

$$C(\mathbf{X}, \mathbf{Y}) = \min \left\{ \sqrt{\sum_{k=1}^{K} D_{w_k}} \right\}, \tag{1}$$

where D is a matrix of the pairwise distances between all points in **X** and **Y**. Equation 1 can be solved by dynamic programming using the following recursion:

$$C_{i,j} = D_{i,j} + \min \left\{ C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1} \right\}, \tag{2}$$

where C is a $M \times N$ warping cost matrix, and $C_{i,j}$ is the minimum warping cost between time series of lengths i and j. Once C is computed, the warping path $W = \{w_k\}_{k=1}^{K}$ can be obtained by a greedy search starting from $C_{M,N}$ and ending in $C_{1,1}$. With the point-correspondence assignments at hand, the distance measure between two sequences can be determined by:

$$\mathbf{d}(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{k=1}^{K} D_{w_k}}{K}. \tag{3}$$

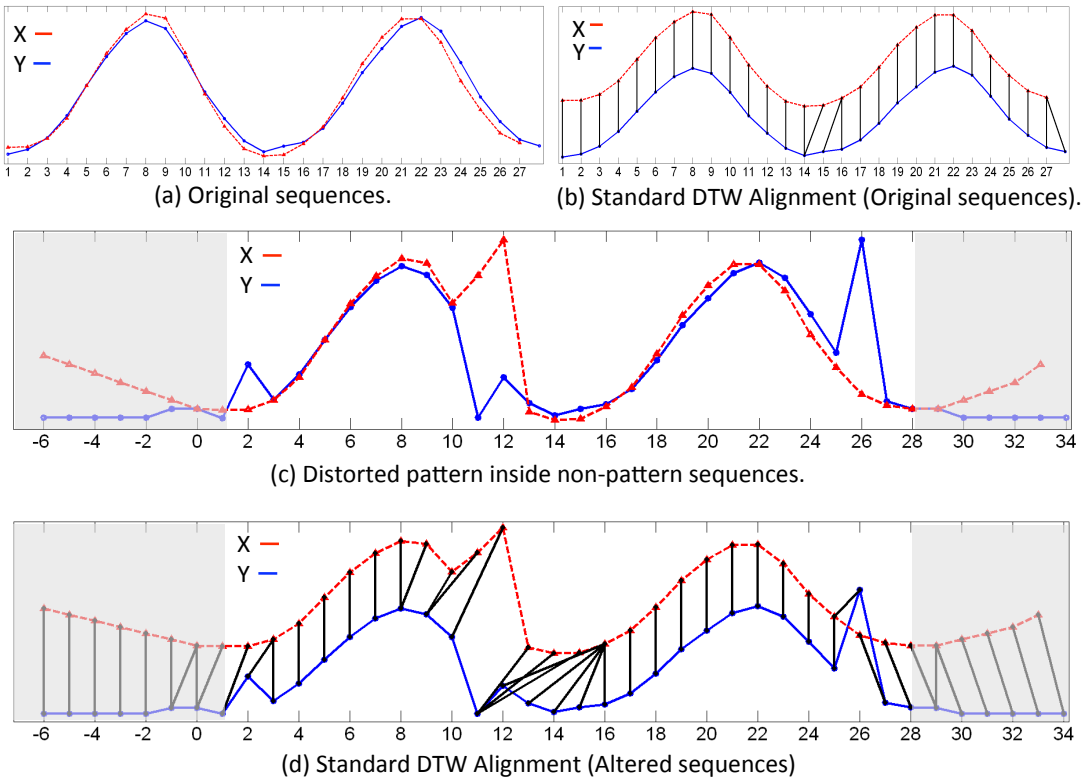


Figure 5: Applying the traditional DTW algorithm to noisy sub-pattern matching. (a) Original pattern subsequences. (b) DTW Correspondence assignment. (c) Pattern subsequences were perturbed with 10% random noise and inserted into non-pattern sequences. (d) DTW correspondence assignment for the sequences in (c). Non-shaded regions correspond to the noisy original pattern subsequences. *(Sequences in (b), (c), and (d) were vertically shifted for display purposes only)*

Here, D_{w_k} is the cost of assigning correspondences w_k, and is usually calculated as the Euclidean distance between points $\mathbf{x}_{i_k}$ and $\mathbf{y}_{j_k}$. Next, we describe the details of our sequence matching algorithm.

4.1.2. Robust Sequence Matching

We propose a Robust Sequence-Matching algorithm (RSM) that produces approximate alignment of sequences. In comparison with the DTW, our algorithm produces fewer matches on the subsequences without actor-object states (i.e., non-pattern subsequences), and is less sensitive to noise. We impose two constraints that improve the alignment robustness. The first of them is the strict temporal monotonicity given by $i_k < i_{k+1}$, $j_k < j_{k+1}$, $\forall\ k$. This constraint ensures the original sequences' temporal order, and prohibits multiple correspondence assignments for a single point. The second constraint in our alignment method helps produce fewer matches on points belonging to the non-actor-object state portions of the sequences by allowing for non-continuous correspondences along the time series.

Our method uses dynamic programming, and consists of two main steps. First, a cost matrix C is built using a recursive accumulation function of the pairwise point distances $D_{i,j}$. Secondly, the optimal warping function is determined by tracing back the minimum cost path in C. In order to produce fewer matches on the non-pattern subsequences, we relax the warping path continuity constraint of the original DTW algorithm.

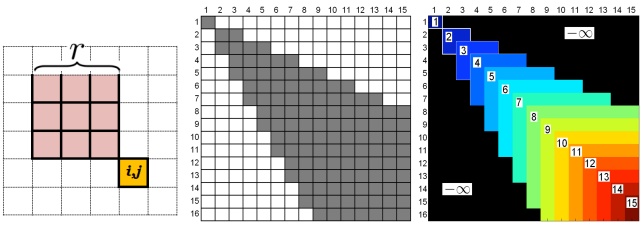


Figure 6: Components of the RSM algorithm. (a) Cost matrix computation strategy for $r = 3$. (b) Form of the cost matrix for $r = 2$ with only dark region elements computed. (c) Example of the slity matrix R computed for $r = 2$.

This is accomplished by the following cost matrix calculation:

$$C_{i,\,j} = D_{i,\,j} + \min_{m,\,n} C_{i-m,\ j-n}, \tag{4}$$

where $m = 1, \ldots, r$ and $n = 1, \ldots, r$, and r is the size of a local constraint window representing the range of points that the algorithm is allowed to involve in the computation. The graphical representation of the scheme for the cost matrix element $C_{i,j}$ is shown in Figure 6(a). The form of Equation 4 does not require computation of all elements in matrix C. Figure 6(b) displays the structure of the cost matrix with the dark region corresponding to the filled elements. Equation 4 prohibits multiple correspondence assignments to a single point while allow-

ing for discontinuities in the warping path. The procedure for computation of matrix C is given in Function 1.

Function 1: `ComputeCostMatrix(D, r)`

Input: Distance matrix D for sequences $\mathbf{X}$ and $\mathbf{Y}$
Size r of the local window
Output: Cost matrix C

Declare $C = \infty$ of size $M \times N$
$C_{1,1} = D_{1,1}$
for $i = 2$ **to** M **do**
  for $j = 2$ **to** N **do**
    $C_{i,j} = D_{i,j} + \min_{m,n} C_{i-m,j-n}$, where $m = 1, \ldots, r$ and $n = 1, \ldots, r$
  end
end

Intuitively, an element $C_{i,j}$ of the cost matrix C represents the cost of matching subsequences up to the pair of points (i, j) such that the temporal distance between two consecutive matches does not exceed r. Stated more formally, we have $w_k = (i, j)$ and $w_{k+1} = (i', j')$, where $i < i' \leq i + r$ and $j < j' \leq j + r$. The procedure in (4) is biased towards short but good matches. Overly sparse matchings may not provide optimal performance and bias toward longer matches must be introduced. This bias is enforced at the stage of computing the path through the cost matrix C, and is described in detail in Function 2. The approach is depicted in Figure 7. Figure 7(a) represents two sequences distorted by noise. The figure assumes that the correspondences were traced from the end of the sequences up to the pair of points (Z, z). The submatrices in Figure 7(b) correspond to the subsequences (A, B, C, D, E, Z) and (a, b, c, d, e, z), and are as follows: pairwise distance submatrix D_s, dynamic programming cost submatrix C_s, and the reachability submatrix R_s. The $M \times N$ reachability matrix R establishes the notion of proximity of pairs of matches, and for any given pair, $R_{i,j}$ encodes the direction of motion in the cost matrix toward the origin of the matrix. In practice, $R_{i,j}$ represents the maximum number of point-correspondence pairs that is possible to assign for a subsequence starting with the pair $(1, 1)$ and ending with (i, j). The reachability matrix R is given by the following recursive equation:

$$R_{i,j} = 1 + \max_{m,n} R_{i-m,j-n}, \tag{5}$$

where $m = 1, \ldots, r$ and $n = 1, \ldots, r$. An example of the reachability matrix computed for $r = 2$ is shown in Figure 6(c). The regions marked with $-\infty$ correspond to the locations in the cost matrix C that cannot be reached from the matrix origin. The reachability matrix can be computed efficiently using the correspondences computation procedure given in Function 2. Note that, in Figure 7(b), both C_s and R_s are computed for $r = 2$.

For a non-square matrix, R, there will be several elements that have the maximum value. We begin computation of the warping path starting with the element (i, j) that corresponds to the largest value in the reachability matrix while having the smallest associated cost. Formally, let $\Omega = \{(\widehat{k, l}) : (\widehat{k, l}) = \arg\max_{k,l} R_{k,l}\}$ be a set of matrix elements corresponding to the largest values in R. For non-square matrix R there will be more than one element in Ω (e.g., Figure 6(c)). Then, the starting

Function 2: `TraceCorrespondences(D, C, r, σ)`

Input: Distance matrix D for sequences $\mathbf{X}$ and $\mathbf{Y}$
Cost matrix C
Size r of the local window
Threshold σ on the maximum tolerable distance between two correspondence points
Output: Point-correspondences W

// Initialize Reachability Matrix R
Declare $R = -\infty$ of size $M \times N$
$R_{1,1} = 1$
for $k = 2$ **to** $\min\{M, N\}$ **do**
  $R_{i,k} = k$, where $i \in [k, ..., k + (k - 1) * r]$
  $R_{k,j} = k$, where $j \in [k + 1, ..., k + (k - 1) * r]$
end

// Calculate correspondences W
Declare empty lists W of point-correspondences
$(i, j) = \arg\min_{k,l} C_{k,l}$, **s.t.** $R_{i,j} = \max_{k,l} R_{k,l}$
Add pair (i, j) to the head of list W
while $i > 1$ **and** $j > 1$ **do**
  Obtain a set of candidate coordinates (m, n), where $m \in [i - r, ..., i - 1]$ and $n \in [j - r, ..., j - 1]$
  if $D_{m,n} \geq \sigma, \forall m, \forall n$ **then**
    // Distance values too high. Choose the direction of smallest cost.
    $(i, j) = \arg\min_{m,n} C_{m,n}$
  else
    // Some distance values are acceptably small. Select the pair that locally maximizes the number of matches.
    $(i, j) = \arg\max_{m,n} R_{m,n}$, **s.t.** $D_{i,j} < \sigma$
  end
  Add pair (i, j) to the head of the list W
end

element is found using the following equation:

$$(i, j) = \arg\min_{(k,l)\in\Omega} C_{k,l}. \tag{6}$$

If the two sequences under consideration are of the same length, then there will be a single maximum element in the square matrix R, and, the solution of Equation 6 will correspond to the end-points of the sequences. For two sequences of different lengths, the starting element (i, j) will correspond to the end-point of the shorter sequence, and a point from the longer sequence that may not necessarily be the end-point. For example, consider two sequences whose matrix R is shown in Figure 6(c). There are two elements in the matrix R that correspond to the largest value in R (i.e., element $(16, 15)$ and element $(15, 15)$). Depending on the associated costs, the starting element can be either $(16, 15)$, which corresponds to the end-points of the two sequences, or $(15, 15)$, which corresponds to the end-point of the shorter sequence, and to the next to end-point of the longer sequence.

When tracing point-correspondences using the cost matrix C, we introduce the parameter σ to represent the maximum tolerable distance between pairs of points in the final correspondences set. In this case, when tracing the sequence of matches for a given correspondence pair (i, j), we obtain candidate

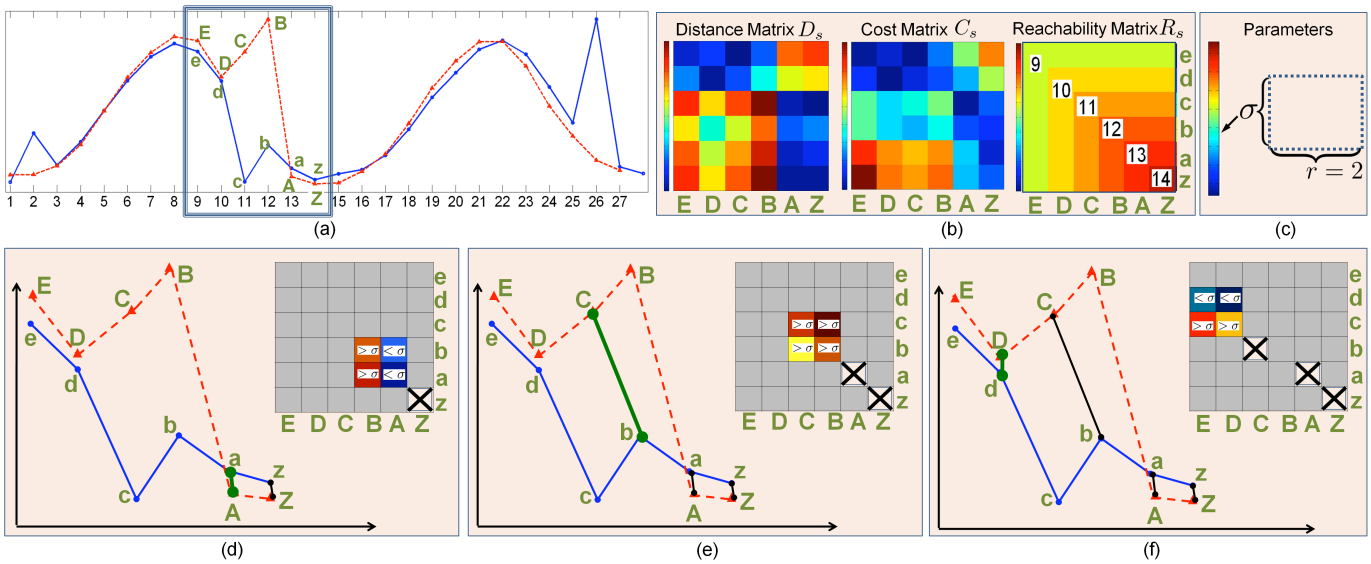


Figure 7: Point-correspondence assignments using RSM. (a) Two perturbed sequences. (b) Data structures in our algorithm: distance submatrix D_s, cost submatrix C_s, and reachability submatrix R_s. (c) Example values of the algorithm parameters. The magnitudes of parameters σ and r in (c) are at the same scale as the figures (d,e,f). In addition, the color value of σ is also provided. (d,e,f) Example of obtaining the next three correspondence pairs given the most recent correspondence. The tables in (d,e,f) show the next candidate correspondence pairs. The cells for the correspondence pairs assigned so far are marked with the cross. *(The figure is best viewed in color.)*

matches (m, n) where $m \in [i-r, ..., i-1]$ and $n \in [j-r, ..., j-1]$. The selection of candidate correspondence pairs is performed in the same way as in Function 1. Among the values $D_{m,n}$ such that $D_{m,n} \leq \sigma$, we select the pair (m, n) that satisfies the following equation:

$$(i, j) = \arg\max_{m,n} R_{m,n}\,, \quad \textbf{s.t. } D_{i,j} < \sigma. \tag{7}$$

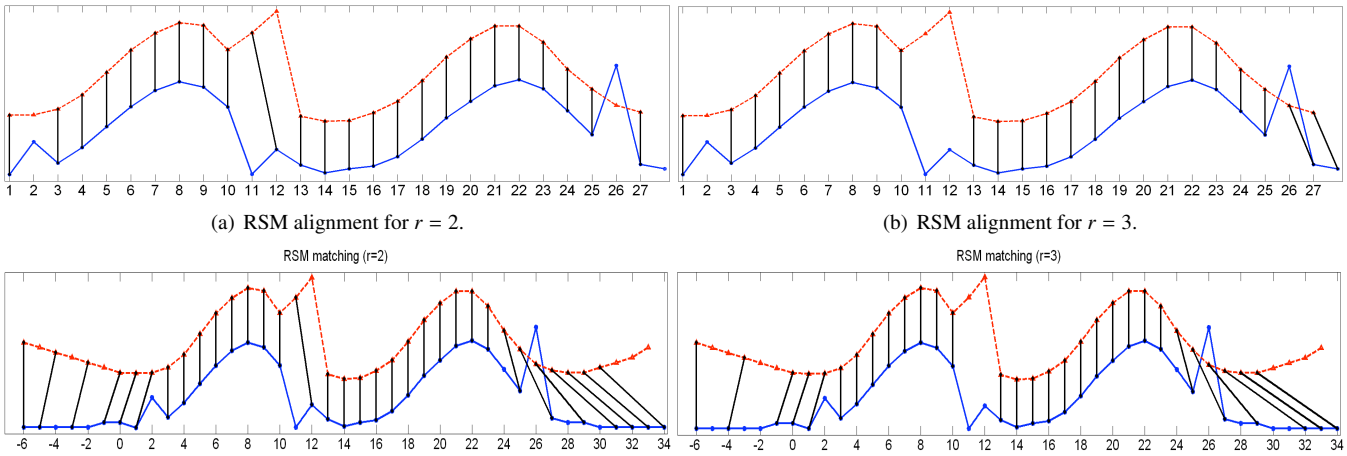
Equation 7 can be interpreted as selection of the "nearest" correspondence that satisfies $D_{i,j} < \sigma$, where the proximity between points is encoded in matrix R.

The maximization in (7) results in the selection of the next acceptable pair of matches that is *closest* to the last assigned correspondence pair (i, j). This introduces the necessary bias towards denser matchings. Consider the example in Figure 7. Assuming that the most recent point-correspondence assignment was (Z, z), Figures 7(d), (e) and (f) provide an example of obtaining the next three correspondence pairs given the parameters in Figure 7(c). Figure 7(d) shows the case for which more than one candidate matches have acceptable distances. In the example, the pairs (A, a) and (A, b) have corresponding distance values $D_{A,a} < \sigma$ and $D_{A,b} < \sigma$. However, the pair (A, a) is preferable as it maximizes the local matching density. The fact that pair (A, a) is *closer* to the pair (Z, z) is reflected in the corresponding values of the slity matrix (i.e., $R_{A,a} = 13$ and $R_{A,b} = 12$). In the case when all values $D_{m,n}$ are greater that σ, we select the pair corresponding to the smallest cost $C_{m,n}$. The use of the cost matrix C (rather than the distance matrix D) when no candidate pairs have acceptable distances is key to our algorithm. Intuitively, the cost matrix C encodes the smallest cost of matching highly noisy subsequences. Thus, when none of the values $D_{m,n}$ are acceptable, the algorithm selects the direction of the smallest cost. In this way, the matching algorithm is able to balance dense and low-cost matchings. An example of the above scenario is depicted in Figure 7(e). Here, none of the pairs (B, b), (B, c), (C, b) and (C, c) have the acceptable distance. Therefore, the candidate pair with the smallest cost is selected (i.e., the pair (D, c)). Here, the pair (D, c) also represents the smallest distance $D_{D,c}$. Finally, the example in Figure 7(f) displays another case in which more than one candidate pair have acceptable distance values. Again, the pair locally maximizing the matching density is chosen.

Our sequence alignment algorithm is effective in discarding noisy points. As an example, consider the two noisy sequences from Figure 8(a). The figure shows alignment results obtained for values $r = 2$ and $r = 3$. In both plots, discontinuities in the matches represent noisy data points. When $r = 2$, a discontinuity cannot span over more than one point (i.e., frame). Thus, one of the noisy points at $t = 11$ was assigned a correspondence (Figure 8(a)). Increasing the value of r allows us to filter out noisy points (Figure 8(b)). The results also depend on the choice of the parameter σ. An empirical strategy for selecting a good σ value will be discussed in the end of this section. With point-correspondence assignments at hand, we can calculate the distance measure between two sequences using Equation 3.

4.1.3. Empirical Selection of Parameter σ

For noise-free well-aligned sequences $\mathbf{X}$ and $\mathbf{Y}$, and for all $M \times N$ corresponding pairwise distance values, we expect at most $\min\{M, N\}$ pairs to have close-to-zero distance values. Consequently, the value of σ is selected such that at least $\min\{M, N\}$ elements in matrix D are below σ. Following this strategy, σ will be different for different pairs of sequences, and will depend on the level of correlation between the data



(a) RSM alignment for $r = 2$. (b) RSM alignment for $r = 3$.

(c) RSM alignment for $r = 2$. (d) RSM alignment for $r = 3$.

Figure 8: Example performance of the RSM algorithm for the sequences in Figure 5. Point-correspondences obtained for $r = 2$ (a) and $r = 3$ (b).

points. We acknowledge that this empirical approach to selecting σ may be not optimal for a particular dataset. Moreover, from our experience, one can achieve a slightly better recognition performance on leave-one-out cross-validation experiments by performing a sweep over possible σ. Nevertheless, the improvement in recognition performance using optimal σ in our experiments was not significant.

4.2. Step 2 – Extracting Actor-Object States

The sequence matching procedure described above provides low-density correspondence assignments for highly dissimilar subsequences. We use this property to help discard highly dissimilar portions of sequences. More specifically, we propose to iteratively remove subsequences that produce low-matching densities. For this, we define the following measure of matching density at position i in sequence $\mathbf{X}$:

$$\rho_i^{\mathbf{X}} = \frac{1}{s+1} \sum_{i=0}^{s-1} \tilde{\delta}(i - s, W_{\mathbf{X}}), \tag{8}$$

where $s \in \mathbb{N}$ is the moving window size, and $W_{\mathbf{X}}$ is the set of indices in $\mathbf{X}$ that were assigned correspondence pairs. The function $\tilde{\delta}$ is a presence function that tests whether a given integer is an element of a specific integer-valued vector, and is defined as:

$$\tilde{\delta}(i, W_{\mathbf{X}}) = \begin{cases} 1, & i \in W_{\mathbf{X}} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The matching density $\rho_j^{\mathbf{Y}}$ at position j of sequence $\mathbf{Y}$ is defined similarly in terms of $W_{\mathbf{Y}}$ (i.e., the set of $\mathbf{Y}$ indices that were assigned correspondence pairs). We use the above density measures to guide a subsequence-removal process based on a minimum density threshold, λ. We remove low-density matching subsequences from the beginning and end of both sequences $\mathbf{X}$ and $\mathbf{Y}$. In practice, we perform the sequence matching backward and forward while removing the low-density regions at the end of both sequences. The iterative pattern discovery procedure is described in Function 3.

Figure 9 shows four iterative steps of the pattern-discovery algorithm for two bell-shaped patterns. Here, the bell-shaped patterns correspond to the actor-object states of interest. While there is a distinct bell-shaped pattern inside the sequences, the subsequences outside the pattern of interest are considerably different. The figure shows iterations for $r = 3$ and matching density threshold $\lambda = 0.8$. For each iteration, the plots display the point-correspondence assignments, distribution of matching densities calculated for $s = 1$, and, finally, the sequences with low-matching density regions removed. During Iterations 1 and 3, matching is performed in a *forward* manner, while matching in Iterations 2 and 4 takes place *backward*. Notice that, in the matching density plot of Iteration 1, there are regions both at the beginning and the end of the sequences that have matching densities below the acceptable value σ. Nevertheless, since Iteration 1 is performed in a forward manner, only the low-density regions at the end of sequences are removed. As shown in Figure 9, the pattern subsequence was correctly recovered during the fourth iteration.

Function 3: `DiscoverPattern(`D, λ`)`

Input: Distance matrix D for sequences $\mathbf{X}$ and $\mathbf{Y}$
Threshold λ on the acceptable matching density
Output: Pattern-based sequence distance $\mathbf{d}$

repeat
  // Match current sequences
  W = MatchSequences(D)
  // Extract pattern subsequence
  Calculate matching densities $\rho^{\mathbf{X}}$ and $\rho^{\mathbf{Y}}$
  Assign $e_{\mathbf{X}} = \max k, \quad \mathbf{s.t.}\ \rho_k^{\mathbf{X}} \geq \lambda$
  Assign $e_{\mathbf{Y}} = \max k, \quad \mathbf{s.t.}\ \rho_k^{\mathbf{Y}} \geq \lambda$
  $D = D_{e_{\mathbf{X}}\ldots 1, e_{\mathbf{Y}}\ldots 1}$
until *W does not change* ;
Calculate sequences distance $\mathbf{d} = \sqrt{\frac{\sum_{k=1}^{K} D_{w_k}}{K}}$

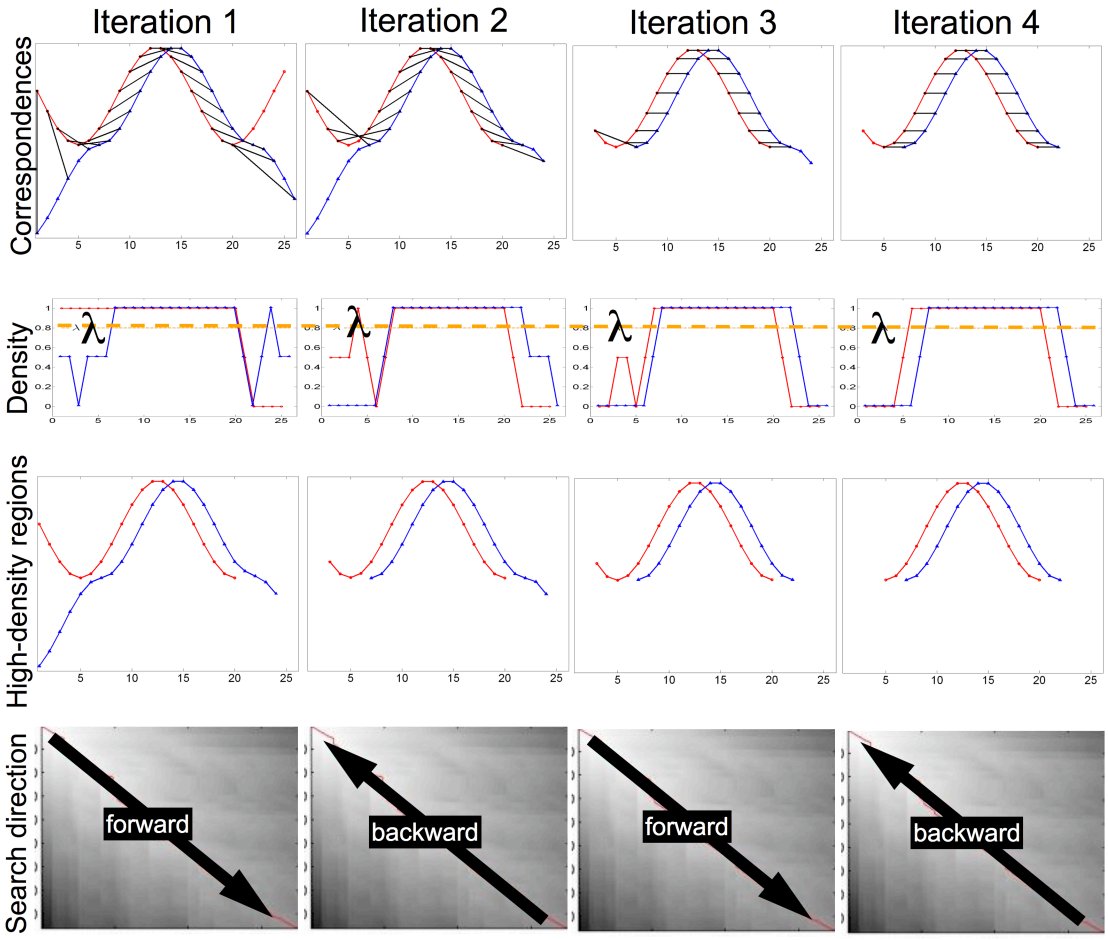


Figure 9: Example of the iterative pattern discovery.

4.3. Actor-Object Interaction Classification

Once a similarity measure between trajectories is established, a nearest-neighbor classification score can be obtained for every trajectory and every information type (i.e., motion and appearance). Given a set of L training video sequences $(\mathcal{V}^1, ..., \mathcal{V}^L)$ and a test sequence $\widehat{\mathcal{V}}$, the final classification score is obtained as the weighted linear combination of individual scores:

$$\gamma = \sum_{i=1}^{K} \alpha_i^{\mathbf{A}} \mathbf{d}_i^{\mathbf{A}} + \sum_{i=1}^{K} \alpha_i^{\mathbf{M}} \mathbf{d}_i^{\mathbf{M}}, \tag{10}$$

where $\mathbf{d}_i^{\mathbf{A}}$ is the nearest-neighbor classification score obtained for the appearance data ($\mathbf{A}$) extracted for the i-th subtrajectory $\mathcal{T}_i$, and is given by:

$$\mathbf{d}_i^{\mathbf{A}} = \min\{\mathbf{d}(\widehat{\mathbf{a}}_i, \mathbf{a}_i^1), \ldots, \mathbf{d}(\widehat{\mathbf{a}}_i, \mathbf{a}_i^L)\}. \tag{11}$$

Similarly, $\mathbf{d}_i^{\mathbf{M}}$ is the nearest-neighbor classification score obtained for the motion data ($\mathbf{M}$) extracted for the i-th subtrajectory $\mathcal{T}_i$, and is given by:

$$\mathbf{d}_i^{\mathbf{M}} = \min\{\mathbf{d}(\widehat{\mathbf{v}}_i, \mathbf{v}_i^1), \ldots, \mathbf{d}(\widehat{\mathbf{v}}^j, \mathbf{v}_i^L)\}. \tag{12}$$

The coefficients $\alpha_i^{\mathbf{A}}$ and $\alpha_i^{\mathbf{M}}$ are calculated as the inverse of the average distance between the test sequence and the training sequences:

$$\alpha_i^{\mathbf{A}} = \frac{1}{\frac{1}{L}\sum_{j=1}^{L} \mathbf{d}(\widehat{\mathbf{a}}_i, \mathbf{a}_i^j)} \quad \text{and} \quad \alpha_i^{\mathbf{M}} = \frac{1}{\frac{1}{L}\sum_{j=1}^{L} \mathbf{d}(\widehat{\mathbf{v}}_i, \mathbf{v}_i^j)}. \tag{13}$$

Equations in (13) are the normalizaters for classification scores obtained for the two information types.

5. Experimental Results

Our experiments are divided into three main parts. In the first part of our experiments, we test our method on a set of synthetic sequences as well as on sequences of human motion silhouettes disturbed with synthetically generated occlusion effects. Then, we demonstrate the effectiveness of our pattern discovery method for actor-object interaction recognition.

5.1. Synthetic Sequences

While our proposed RSM algorithm is designed to be a part of the pattern-subsequence discovery process, it is worthwhile to evaluate its performance for the alignment of noisy

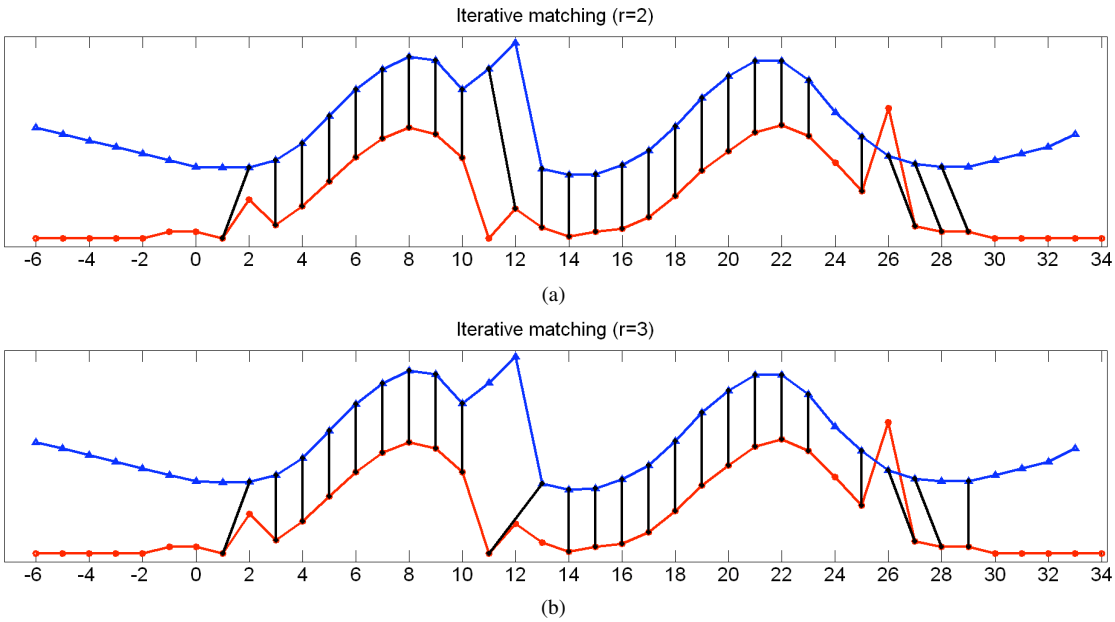


Figure 10: Pattern discovery on synthetic sequences. Final point-correspondence assignments obtained for $r = 2$ (a) and $r = 3$ (b).

sequences. We began by testing our pattern extraction method on two synthetic patterns disturbed by noise. The first pattern is shown in Figure 10. In this experiment, we show the effect of the choice of r in the pattern discovery process. For both $r = 2$ and $r = 3$ the inside pattern was correctly discovered. Additionally, most of the noisy points were automatically excluded from the final matching. However, due to small value of r in Figure 10(a) a correspondence pair with large cost was included into the final matching.

Another experiment on synthetic sequences was performed to evaluate the method for matching noisy and misaligned sequences. Here, 15% of the points in the original sequences from Figure 9 were perturbed with random noise (Figure 11(a)). Figure 11(b) displays the final correspondence assignments produced by our pattern-discovery algorithm. Figure 11(c) displays the interpolation of the matches over missing points inside the pattern. The figure suggests that the target pattern can be well discovered even in the presence of noise.

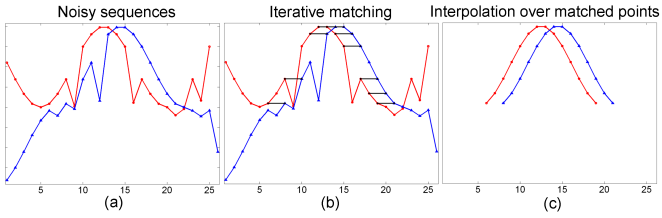


Figure 11: Pattern discovery for sequences in Figure 9 distorted by noise: (a) Noisy input sequences, (b) correspondence assignment, and (c) missing points interpolation.

5.2. *Human Action Classification Under Occlusion*

We tested the robustness of our algorithm on a human action classification task. Here, we were especially interested in observing the effect of the presence of occlusion and prolonged loss-of-track. To accomplish this goal, we provide a comparison between our sequence-matching algorithm with existing alternative approaches (i.e., DTW and longest common subsequence (LCSS) algorithms) using motion patterns acquired by a single static camera. Motion sequences from the Weizmann's action dataset [35] were used for evaluation. We want to emphasize that in this specific set of experiments we assess the performance of of the sequence alignment component (i.e., RSM) of our pattern-discovery approach.

5.2.1. *Video Data Preparation*

In this experiment, pose registration was performed by isolating the foreground silhouette using a simple background-subtraction operation. A more robust foreground-segmentation method can be used [36] whenever a static background is not available. A bounding box enclosing the foreground pixels was constructed for each frame, and the largest frame size was chosen to represent the standard frame size for the entire sequence. Finally, all remaining frames were aligned (w.r.t. the center pixel) to the center of the standard selected frame. Once registration was completed, all silhouettes were converted into a smooth gray-level gradient using a distance transform [31]. In our method, the highest distance transform values are assigned to the silhouettes' most medial-axis points. The values in all resulting images were then re-scaled to a predefined maximum value (e.g., 255). The result of this preprocessing step is illustrated in Figure 13(bottom row).

5.2.2. Classification

The medoid [37] of a set of training sequences was obtained to serve as the model for a specific motion. We adopted a leave-one-out evaluation scheme by taking videos of one subject as testing data, and using sequences of the remaining subjects for training. Additionally, only the best match for each model was considered when making the labeling decision. Short-term loss-

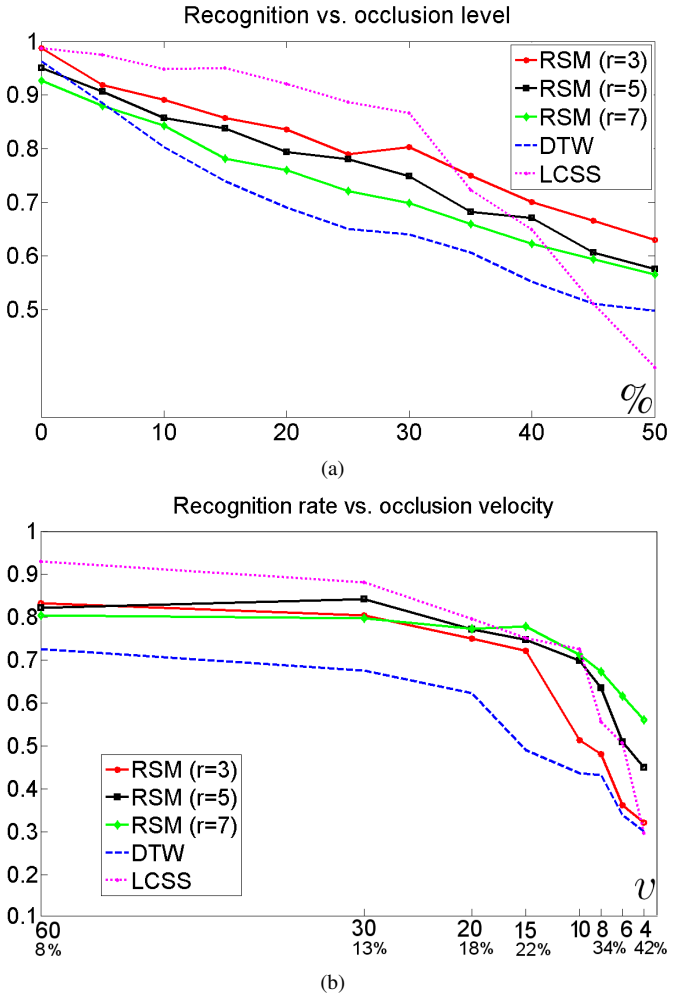


Figure 12: (a) Evolution of recognition performance with the occlusion level for randomly occluded individual silhouettes; (b) Simulation of prolonged loss-of-track due to moving rigid occlusion; The percentages indicate the amount of consecutively perturbed frames.

of-track was simulated by adding vertical bars to some video frames (Figure 13).

In order to assess the robustness of our method, we gradually increased the percentage of perturbed silhouettes in the sequences (i.e., silhouettes with vertical bars at random positions). The distance-transformed silhouettes were resized to 40×40 pixels. The dimensionality of the data was further reduced to 40 using principal component analysis. For a given pair of sequences, the threshold value σ on the maximum tolerable distance was selected following the empirical selection procedure described in Section 4.1.3. Matching results for different values of r are presented in Figure 12(a).

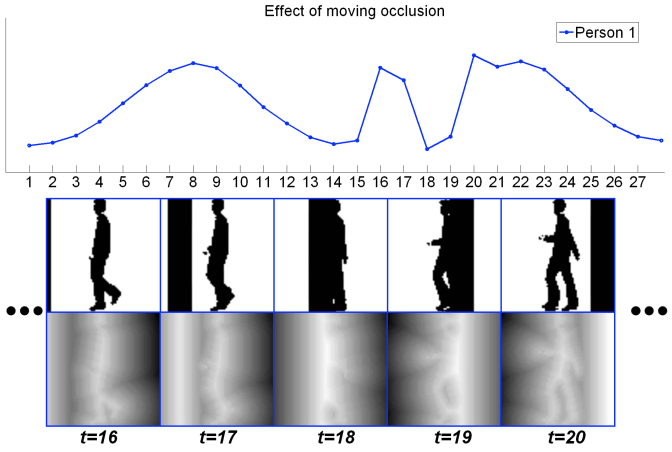


Figure 13: Simulated loss-of-track and its effect on the major eigenvector PCA projection. Occluded frames: 16 through 20.

This experiment was repeated 25 times. The curves show the evolution of the average recognition rates. Our method achieved 98.8% accuracy for the noise-free silhouettes. This was higher than 96.3% accuracy obtained using the standard DTW approach. As the amount of occlusion increases, our robust matching approach significantly outperforms the classical DTW. For example, our method was able to achieve 63.7% recognition rate even when 50% of the silhouettes were occluded, while the DTW could only achieve a 50.6% recognition performance. Additionally, Figure 12 shows a comparison with the LCSS algorithm. LCSS's performance is affected by the choice of its parameters (i.e., threshold on the maximum tolerable inter-point distance, and the width of a lookup window [28]). We performed a parameters sweep to find LCSS parameters such that the average recognition accuracy using LCSS is the highest. The plot in Figure 12(a) suggests that for smaller noise amounts, the LCSS algorithm performed better than our proposed RMS algorithm. At the same time, when the noise level exceeded 35%, our sequence-alignment algorithm outperformed both DTW and LCSS.

In our second experiment, we simulated prolonged loss-of-track caused by a slow moving rigid object. In this case, the occluding object was assumed to be moving with horizontal velocity v relative to the person. Figure 13 shows an example of the effect of the moving occlusion with relative velocity of 10 pixels/frame in the direction opposite to the person's motion. It can be seen from the figure that the occlusion significantly distorts the input signal. However, unlike the previous experiment, the occlusion spans over a non-unit period of time and is equivalent to the prolonged loss-of-track. Loss-of-track duration was proportional to the percentage of occluded silhouettes. Figure 12(b) shows the evolution of the recognition rate as the percentage of occluded frames increases. Results are shown for the classical DTW, LCSS, and for the RSM with local window $r = 3$, $r = 5$, and $r = 7$. Here, our RSM consistently outperformed the classical DTW algorithm by a large margin. Also, for short-duration loss-of-track that corre-

sponds to the low noise levels, LCSS outperformed our RSM algorithm. However, as the number of consecutively occluded silhouettes increases (i.e., prolonged loss-of-track), our method was able to outperform both classical approaches. Notably, Figure 12 also suggests that both DTW and LCSS algorithms performed worse when perturbed (i.e., occluded) silhouettes appeared in consecutive frames. As the result, the performance of the classical algorithms is significantly worse in Figure 12(b) for the same occlusion noise levels as in Figure 12(a).

5.3. Recognizing Human-Object Interactions

5.3.1. Dataset

The goal of our next set of experiments is twofold. First, we focus on assessing the validity of our proposed interaction representation concept. Secondly, we demonstrate the effectiveness of our pattern discovery method for classifying basic human-object interactions. Vision-based human-object interaction recognition is a novel problem with no widely available datasets. We acquired our own dataset of human-object interaction with complex scenarios deliberately chosen to motivate future improvements of human-object interaction methods. Example frames from our dataset are shown in Figures 14 and 15. It consists of eight different actor-object interaction types performed by ten individuals in two different scenarios.

The interactions are "grasp a cup", "grasp a fork", "touch a fork", "grasp a spoon", "touch a spoon", "grasp a toy car", "touch a toy car", and "push a toy car". Every individual interacted with a unique set of objects (i.e., different cups were used by different individuals in a "grasp a cup" interaction). The two scenarios had clean and cluttered background, respectively. In the "cluttered background" scenario, the background was changed for every individual and every interaction type. Any two interaction types in the dataset differ in one of three aspects: (1) different objects and different motions (i.e., "grasp a cup" vs. "touch a fork"); (2) similar objects and different motions (i.e., "grasp a fork" vs. "touch a fork"); and (3) different objects and similar motions (i.e., "grasp a fork" vs. "grasp a spoon"). The choice of interactions was inspired by experiments using functional neuro-imaging in humans [38, 39] to investigate human perception of hand-object interactions. These experiments revealed the presence of specialized neuronal regions for visuomotor actions such as reaching, grasping, and touching. Sequences in our dataset were acquired with a CCD camera at thirty frames per second rate, and video frames were downsized to 144×180 pixels. The average number of frames in the sequences is 24.6, with the videos of "grasp a fork" interaction containing the smallest number of frames (i.e., 20 frames on average), and videos of "push a car" interaction being the longest (i.e., 29.4 frames on average). The dataset is publicly available from the authors' website [1].

5.3.2. Video Pre-Procesing and Feature Extraction

For each video from our dataset, we manually obtained point-finger and thumb trajectories. Velocities were then extracted to represent motion dynamics. Object appearance (i.e., fork and spoon objects) was captured by means of subregions of size 40× 100 pixels. For each subregion, we extracted edges using the Canny edge detector. We applied Gaussian filter smoothing to the original edge maps. Finally, distance transform was applied to obtain a final representation of the subregions.

5.3.3. Classification Results

Again, a leave-one-out evaluation scheme was used. In our experiments, only the best match for each model was considered when making the labeling decision. First, we assessed our method's classification performance using only a single type of information. Figure 16(a) and Figure 16(b) provide the confusion matrices obtained using only either static or motion information, respectively. The values of the algorithm parameters were chosen to be $r = 2$ and $\lambda = 0.6$. Notice that, when only motion information was used, the methods mostly misclassified interactions having similar motion.

Finally, we combined the static and motion information results using the classifier in Equation 10. The algorithm achieved 72.5% and 58.8% recognition accuracy on the clean and clutered backgrounds, respectively. In comparison, the best performance of DTW on the clean scenario in the interactions dataset was only 62.5%, and the best performance of the LCSS was 66.3%. We believe that the stronger performance of our method in the interactions recognition experiment is largerly attributed to the ability to discard irrelevant subsequences corresponding to the beginning and to the end of interaction seqeunces. These irrelevant subsequences do not contain information that is descriptive of the specific interaction, and are misleading for traditional sequence-matching algorithms.

The method does not require a prolonged training stage and the classifier is very simple. A more sophisticated classification algorithm may be used to improve the recognition performance.

For two interactions of the same type performed by two different individuals, the lengths of the pattern subsequences (i.e., the number of actor-object states) extracted by our pattern-extraction algorithm on average constituted 32.1% of the lengths of the video sequences. In contrast, for any two sequences (whether or not of the same interaction type), the length of the extracted pattern subsequence was on average only 28.3% of the original video lengths.

5.3.4. Effect of Method Parameters Selection

Next, we assesed the effect of our method's parameters on the recognition performance. Since the classification score in Equation 10 is obtained as a linear combination of classification scores obtained for static and motion information, the parameters of the pattern discovery algorithm may be different for different information types. For these experiments, we obtained classification results using only static appearance information on the clean scenario of the interactions dataset.

Figure 18(a) shows evolution of the recognition performance for different values of r. While the performance degrades with the increase of r, it is worth pointing out that both for $r = 2$ and $r = 3$ the recognition rates are acceptable. Decreased recognition rates for large values of r may be due to the small du-

[1] `http://www.cs.fit.edu/~eribeiro/actor_object_interactions/`

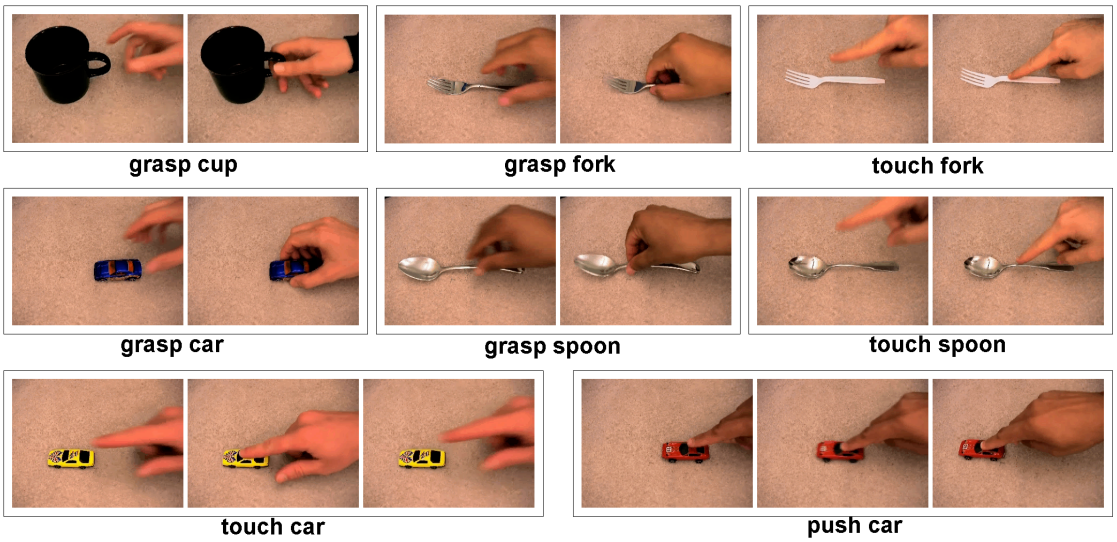


Figure 14: Example frames from video sequences in our interaction dataset performed in clean background scenario.

Table 1: Performance of our interaction recognition approach for appearance subregions of different sizes (clean scenario).

subregion size	recognition rate
40×40	55.0%
40×80	67.5%
40×100	**72.5%**
40×120	68.8%
100×100	62.5%

ration of the constrained motion in primitive actions (i.e., the point finger is only in as few as six frames in the "touch car" interaction). Figure 18(b) shows evolution of the recognition performance for different values of the threshold λ on the minimum matching density. Again, one can observe several values of λ yielding high recognition rates. For large values of λ (i.e., $\lambda = 1$), it was not always possible to achieve satisfactory matching density. If the matching threshold is not met, the algorithm will not produce a matching distance for two interaction sequences. To avoid this, we prohibited the algorithm from discarding more then 75% of the original sequences.

5.3.5. *Effect of Subregion Size Selection*

When performing static-subregion extraction required by our method, the size of the subregions describing the actor-object appearance needs to be defined. Due to the nature of interactions in our dataset, the optimal subregion size in the above experiments was found to be 40×100 pixels. Table 1 shows recognition results for the clean scenario using different subregion parameters.

The results in the table suggest that performance significantly deteriorates when the size of the subregion is too small. The main reason for this behavior is the fact that small subregions cannot effectively capture the object's and actor's appearance. Also, when using small subregions size the method learns interactions with only a part of the object, rather than with the entire object (i.e., learning "touch handle" interaction instead of "touch fork" interaction). On the other hand, there are several reasons why recognition performance drops for very large subregions. First, due to small frame size in the videos it is sometimes impossible to extract requested subregion at the tracked location. When a subregion is partially outside the frame's boundary, we extracted the closest valid subregion. As a result, when a tracked point is close to the frame's boundary, the corresponding extracted subregion may not correctly represent the static information around the point. While we did not encounter this problem for smaller subregions, the classification performance will necessarily degrade for subregions of large sizes. Additionally, large subregions capture significant amount of background information that can further reduce the recognition accuracy.

5.4. *Detecting Interactions in Long Sequences*

We also performed an experiment to access the potential of our approach to detect specific video subsequences in longer video sequences. We obtained a set of YouTube video sequences of a stirring interaction performed during cooking. We extracted a short (i.e., 20 frames) stirring interaction in which person performs counterclockwise stirring motion with a spoon. Example frames along with the tracked finger trajectory are shown in the top row of Figure 19. We then obtained a longer sequence of stirring activity (middle rows in Figure 19) featuring the same person stirring a different ingredient.

We then performed sequence matching using a sliding window of width equal to 25 frames. Figure 19 (bottom row) shows example frames from a subsequence that yielded the closest distance measure to the counterclockwise stirring motion. Additionally, Figure 20 shows a plot of matching costs obtained for some of the sliding windows. It can be seen from Figures 19

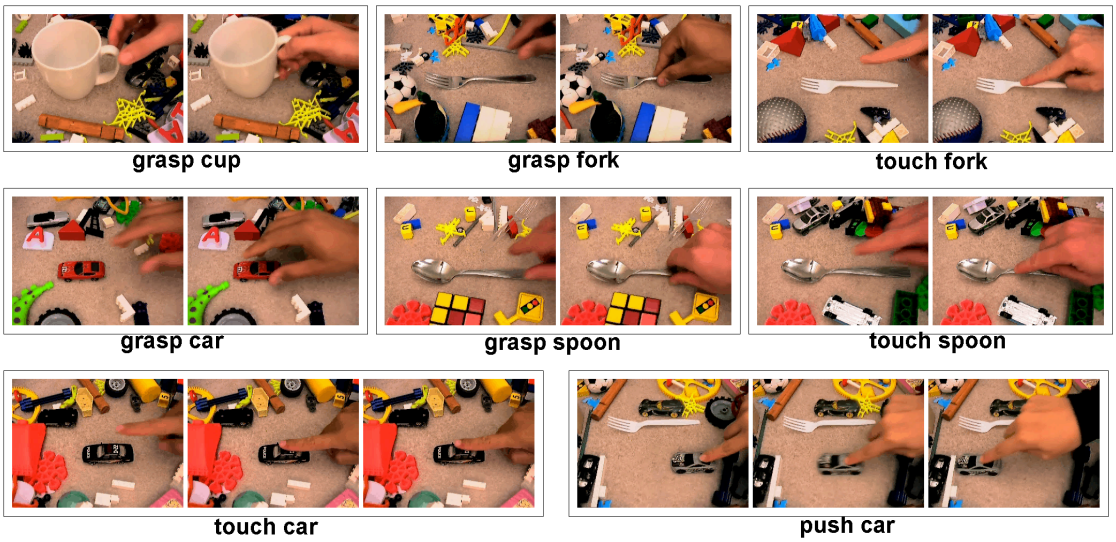


Figure 15: Example frames from video sequences in our interaction dataset performed in cluttered background scenario.

and 20 that the stirring motion subsequence that corresponds to the lowest matching cost is visually similar to the original counterclockwise stirring motion.

6. Conclusions

In this paper, we presented the concept of "actor-object" states for recognition of atomic human-object interactions from videos. We validated the concept by proposing a data-driven approach for extracting actor-object states at the level of detail of individual body parts. Interactions were represented by a collection of image regions combined with motion trajectories velocities. Our method discovered these short-term constrained pattern subsequences by matching pairs of misaligned sequences of features. We showed promising results obtained on an acquired human-object interactions dataset. The experiments suggest that the use of actor-object states allows for satisfactory recognition performance.

Our pattern-discovery approach presents a deterministic similarity measure for pairs of interaction videos and can be used in the case of small training datasets. Although our proposed approach is view dependent, it can be applied to many realistic scenarios (e.g., static-camera surveillance systems). Moreover, complex activities often consist of sequences of primitive actions. Obviously, allowing such system to recognize human-object interactions significantly extends the number of target activities that can be analyzed.

One of the promising directions for future research would be an extension of the proposed algorithm to learning primitive interactions from long video-sequences of activities, each of which may contain many different interactions types. This task is conceptually similar to the problem of extracting video signemes from sign language sentences [40].

Acknowledgements

This research was supported in part by the U.S. Office of Naval Research under contract: N00014-05-1-0764.

References

[1] Roman Filipovych and Eraldo Ribeiro. Recognizing primitive interactions by exploring actor-object states. In *Intl. Conf. on Computer Vision and Pattern Recognition*, Anchorage, Alaska, 2008.

[2] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. Rehg. A scalable approach to activity recognition based on object use. In *International Conference on Computer Vision*, 2007.

[3] M. S. Ryoo and J. K. Aggarwal. Recognition of composite human activities through context-free grammar based representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1709–1718, 2006.

[4] Masanobu Yamamoto, Humikazu Mitomi, Fuyuki Fujiwara, and Taisuke Sato. Bayesian classification of task-oriented actions based on stochastic context-free grammar. In *International Conference on Automatic Face and Gesture Recognition*, pages 317–323, 2006.

[5] N.T. Nguyen, S. Venkatesh, and H.H. Bui. Recognising behaviours of multiple people with hierarchical probabilistic model and statistical data association. In *British Machine Vision Conference*, page III:1239, 2006.

[6] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

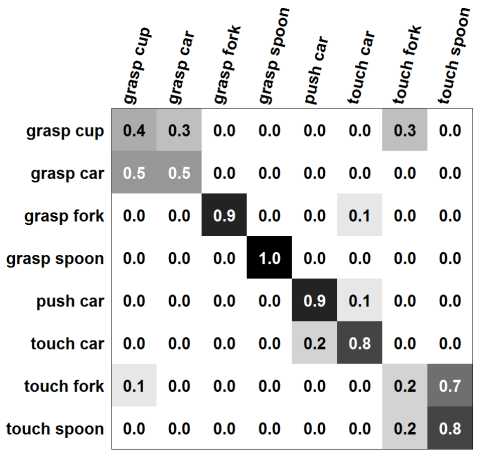
[7] Sangho Park and J. K. Aggarwal. Semantic-level understanding of human actions and interactions using event hierarchy. In *CVPRW*, volume 1, page 12, 2004.

[8] Ram Chelappa, Amit K. Roy-Chowdhury, and Shaohua K. Zhou. *Recognition of Humans and Their Activities Using Video*. Morgan & Claypool Publishers, 2005.

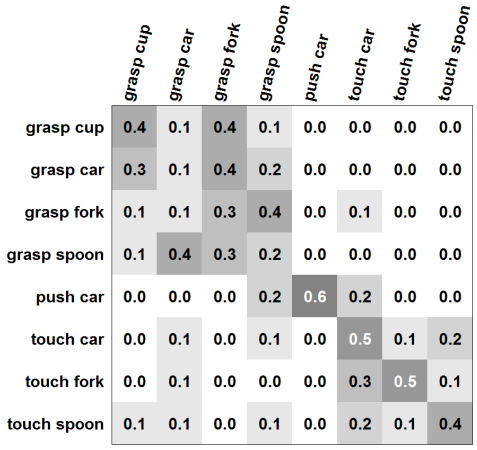
[9] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:2929–2936, 2009.

[10] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, October 2005.

[11] M. Leo, T. D'Orazio, I. Gnoni, P. Spagnolo, and A. Distante. Complex human activity recognition for monitoring wide outdoor environments. In *Intl. Conference on Pattern Recognition, Vol. 4*, pages 913–916, 2004.

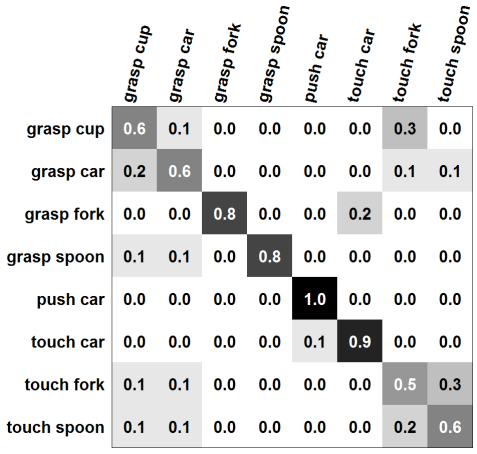


(a)

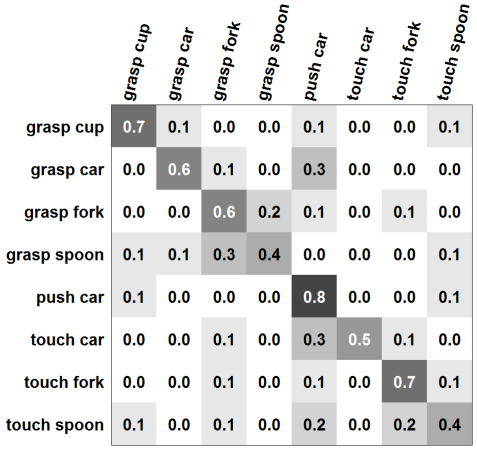


(b)

Figure 16: Clean scenario recognition results (single information type). (a) Only static shape information (68.8% recognition accuracy). (b) Only motion information (37.5% recognition accuracy).



(a)



(b)

Figure 17: Recognition results obtained using combination of two information types. (a) clean background (72.5% recognition accuracy). (b) cluttered background (58.8% recognition accuracy).

[12] O. Boiman and M. Irani. Detecting irregularities in images and in video. In *Conference on Computer Vision and Pattern Recognition*, pages I: 462–469, 2005.

[13] J. C. Niebles and Li Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *Intl. Conf. on Computer Vision and Pattern Recognition*, Minneapolis, USA, June 2007.

[14] Abhinav Gupta, Aniruddha Kembhavi, and Larry S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1775–1789, 2009.

[15] Hedvig Kjellström, Javier Romero, David Martínez Mercado, and Danica Kragic. Simultaneous visual recognition of manipulation actions and manipulated objects. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *ECCV (2)*, volume 5303 of *Lecture Notes in Computer Science*, pages 336–349. Springer, 2008.

[16] Patrick Peursum, Geoff West, and Svetha Venkatesh. Combining image regions and human activity for indirect object recognition in indoor wide-angle views. In *International Conference on Computer Vision*, volume 1, pages 82–89, 2005.

[17] Darnell J. Moore, Irfan A. Essa, and Monson H. Hayes Iii. Exploiting human actions and object context for recognition tasks. *Computer Vision, IEEE International Conference on*, 1:80, 1999.

[18] Ehud Rivlin, Sven J. Dickinson, and Azriel Rosenfeld. Recognition by functional parts. *Comput. Vis. Image Underst.*, 62(2):164–176, 1995.

[19] Louise Stark and Kevin Bowyer. Function-based generic recognition for multiple object categories. *CVGIP: Image Understanding*, 59(1):1–21, 1994.

[20] Michael Pechuk, Octavian Soldea, and Ehud Rivlin. Learning function-based object classification from 3d imagery. *Computer Vision and Image Understanding*, 110(2):173–191, 2008.

[21] Lars Reng, Thomas B. Moeslund, and Erik Granum. Finding motion primitives in human body gestures. In Sylvie Gibet, Nicolas Courty, and Jean-François Kamp, editors, *Gesture Workshop*, volume 3881 of *Lecture Notes in Computer Science*, pages 133–144. Springer, 2005.

[22] Cen Rao, Alper Yilmaz, and Mubarak Shah. View-invariant representation and recognition of actions. *International Journal of Computer Vision*, 50(2):203–226, 2002.

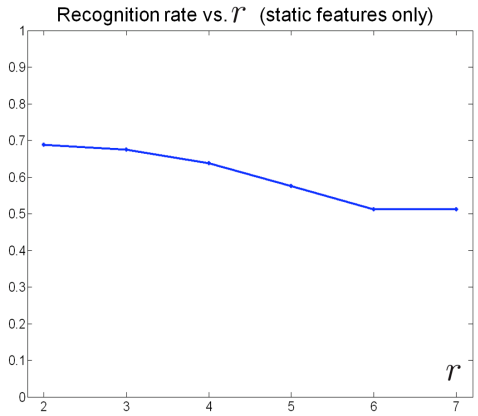
[23] Christoph Bregler. Learning and recognizing human dynamics in video sequences. In *International Conference on Computer Vision and Pattern Recognition*, page 568. IEEE Computer Society, 1997.

[24] Andrew D. Wilson and Aaron F. Bobick. Parametric hidden markov models for gesture recognition. *Trans. Patt. Anal. Mach. Intell.*, 21(9):884–900, 1999.

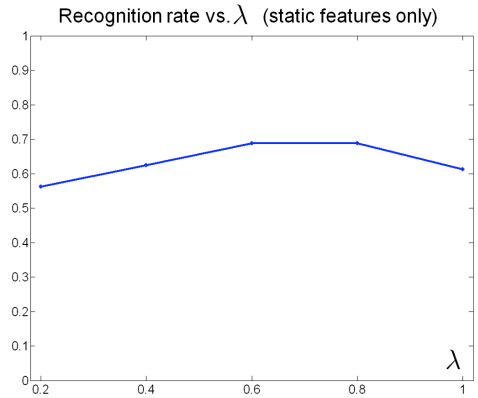
[25] T. K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics and Systems Analysis*, 4(1):52–57, 1968.

[26] Donald J. Berndt and James Clifford. Finding patterns in time series: a dynamic programming approach. *Advances in Knowledge Discovery and Data Mining*, pages 229–248, 1996.

[27] Ashok Veeraraghavan, Amit K. Roy-Chowdhury, and Rama Chellappa. Matching shape sequences in video with applications in human movement



(a)



(b)

Figure 18: Recognition results for the clean scenario obtained for various values of r (a) and λ (b). (Obtained using static information only.)

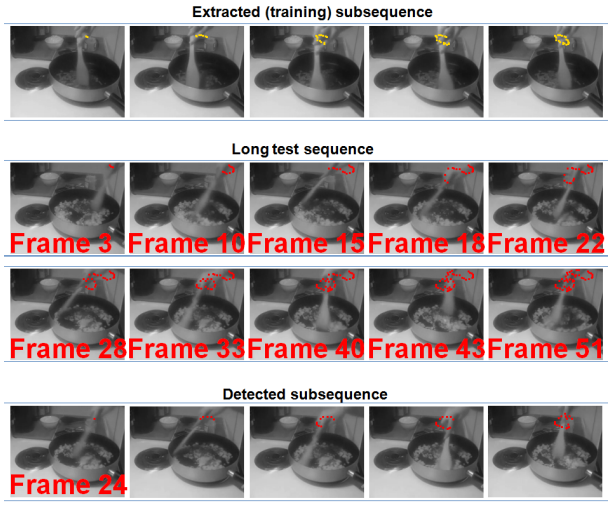


Figure 19: Detecting stirring interaction in longer sequence. (Top) Frames of a counterclockwise stirring motion; (Middle rows) Frames of a prolonged stirring activity; (Bottom row) Detected counterclockwise stirring in the longer activity. The subsequence was detected inside a sliding window that starts at frame 24 of the longer sequence.

Figure 20: Sliding window detection of a stirring interaction. Matching costs obtained inside sliding windows that start at the frames indicated by x-axis. The minimum matching cost was obtained for a subsequence from the sliding window that started at frame 24.

analysis. *Trans. Patt. Anal. Mach. Intell.*, 27(12):1896–1909, 2005.

[28] Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos, and Eamonn Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *ACM KDD*, pages 216–225, 2003.

[29] Thad Starner, Alex Pentland, and Joshua Weaver. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1371–1375, 1998.

[30] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[31] A. Elgammal and C.-S. Lee. Inferring 3d body pose from silhouettes using activity manifold learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 681–688, 2004.

[32] Jaron Blackburn and Eraldo Ribeiro. Human motion recognition using isomap and dynamic time warping. In *Workshop on Human Motion*, pages 285–298, 2007.

[33] S. Cherla, K. Kulkarni, A. Kale, and V. Ramasubramanian. Towards fast, view-invariant human action recognition. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–8, 2008.

[34] Nazli Ikizler and Pinar Duygulu. Human action recognition using distribution of oriented rectangular patches. In *Workshop on Human Motion*, pages 271–284, 2007.

[35] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *Int. Conference on Computer Vision*, pages 1395–1402, 2005.

[36] Y.-L. Tian, M. Lu, and A. Hampapur. Robust and efficient foreground analysis for real-time video surveillance. In *In IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 11821187, 2005.

[37] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, New York, 1990.

[38] J. C. Culham and K. F. Valyear. Human parietal cortex in action. *Current Opinion in Neurobiology*, 16:205–212, 2006.

[39] Scott H. Johnson-Frey, Farah R. Maloof, Roger Newman-Norlund, Chloe Farrer, Souheil Inati, and Scott T. Grafton. Actions or hand-object interactions? human inferior frontal cortex and action observation. *Neuron*, 39:1053–1058, 2003.

[40] S. Nayak, S. Sarkar, and B. Loeding. Automated extraction of signs from continuous sign language sentences using iterated conditional modes. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:2583–2590, 2009.