Invariance in Human Action Analysis

by

Cen Rao
B.S. Beijing University of Aeronautics and Astronautics, 1995
M.E. Beijing University of Aeronautics and Astronautics, 1998
M.S. University of Central Florida, 2001

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2003

Major Professor:
Mubarak Shah

ABSTRACT

Recognition of human actions from video sequences is an active area of research in computer vision. Possible applications of recognizing human actions include video surveillance and monitoring, human-computer interfaces, model-based compression and augmented reality. The motion of an object can be captured by its trajectory. Analysis of human perception of motion shows that information for representing the motion is obtained from changes in the speed and direction of the trajectory. In this dissertation, we propose a computational representation of human action to capture these changes using spatio-temporal curvature of 2-D trajectories. This representation is compact, view-invariant, and is capable of explaining an action in terms of meaningful action units called "dynamic instants" and "intervals". A dynamic instant is an instantaneous entity that occurs for only one frame, and represents an important change in the motion characteristics of the action agent. An interval represents the time period between two dynamic instants during which the action agent's motion characteristics do not change. Starting without a model, we use this representation for recognition and incremental learning of human actions. The Dynamic Time Warping matching is employed to match trajectories of actions using a view invariant similarity measure. The nearest-neighbor clustering approach is used to learn human actions without any training. The proposed method can discover instances of the same action performed by different people from different viewpoints. Our approach heavily uses the properties of 3D epipolar geometry and employs rank constraints in matching 2-D projections of a 3-D action in order to eliminate the distortion due to this projection, without explicitly constructing the 3-D trajectory. We also propose the use of a rank constraint on the fundamental matrix for spatio-temporal alignment of video sequences. This rank constraint is more robust and does not require actual computation of the fundamental matrix.

Therefore it is easier to compute than the previous fundamental matrix based approaches. We propose a dynamic programming approach using the rank constraint to find the nonlinear time-warping function for videos containing human activities. In this way, videos of different individuals taken at different times and from distinct viewpoints can be synchronized. Moreover, a temporal pyramid of trajectories is applied to improve the accuracy of the view-invariant dynamic time warping approach. We show various applications of this approach, such as video synthesis, human action recognition and computer aided training. Compared to the state-of-the-art techniques, our method shows a great improvement. This dissertation makes two fundamental contributions to view invariant action recognition: (1) A view-invariant representation of action trajectories based on Dynamic Instant detection. (2) View-invariant Dynamic Time Warping to measure the similarity between two trajectories. We have successfully applied the view-invariant spatio-temporal information of the action trajectories for both action recognition and video synchronization, without explicitly reconstructing 3D information.

To my wife Qingyi Grace Yu and my parents, who love me.

ACKNOWLEDGMENTS

First and foremost, I would like to thank Dr. Mubarak Shah, my advisor, for his guidance and encouragement through out the course of my PhD. Moreover, I want to thank Dr. Tanveer Syeda-Mahmood who was my mentor during my summer internship at IBM Almaden Research Center in 2002.

Acknowledgements are due to my colleagues Sohaib Khan, Yaser Ajmal, Omar Javed, Khurram Hassan-Shafique, Alper Yilmaz, Zeeshan Rasheed, Paul Smith, Lisa Spencer, Orkun Alatas, Alexei Gritai, Asaad Hakeem, Feixue Han, Jiangjian Xiao Yun Zhai, Yunjun Zhang, and Niels Haering, for several stimulating discussions. Especially, the discussions with Alper and Khurram on action recognition, with Jiangjian on projections, and Alexei on video synchronization are worth mentioning. I am grateful to Lisa, Yaser, and Paul for helping me in the preparation of this manuscript.

I am also grateful to Drs. Niels da Vitoria Lobo, Michael Moshell and Takis C. Kasparis for their service as the members of my committee. I would also like to express my regards to the other professors at the computer science department for the knowledge they shared with me during my stay here. Moreover, I want to thank Heather Oaks, Sonja Rosignol and Linda Locky for their kind help during my study.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Action Recognition

Video is a preservation of data that is of importance to humans. Video information has become ubiquitous in forms of surveillance videos, web-cams, home videos, major motion pictures, etc. Among all these forms, a sizable percentage is used to observe and record human beings performing various tasks in a diverse set of situations. Thus, to understand and analyze the video data, a reliable recognition of human actions is essential. Researchers have been continuously making progress in this area throughout the last decade.

Human action recognition systems find primary application in the following four areas: Surveillance Systems, Human-Computer Interaction, Interactive Spaces and Content-based Video Analysis.

- In a surveillance system, subjects are tracked and their activities are recorded in real-time. Furthermore, those activities are classified into different categories. For example, in [44, 3], computers can monitor some specific activities in an office, a kitchen, or a car; and in [34], the system detects important changes, events, and activities, flags significant events, and presents a summary in terms of key frames to help police patrol electronically. Other similar systems were presented in [22, 36, 41, 44] to facilitate automation of security systems, monitoring of office environments, automated supervision of children, etc. The unusual events will trigger the alarms, and the usual activities can support the study of psychology, and develop working environment. Robust recognition of human activities is the primal problem in surveillance systems.

- Interpreting human response is literally half the problem in developing effective HCI systems. Representing and understanding human actions are obvious prerequisites in eliciting meaningful responses from software user interfaces, virtual instructors, robots and other HCI-based systems [40, 10, 55, 23, 74, 72]. For example, in distance learning, the emphasis on a topic can be learned from actions made by the presenter [71]; the recognition of actions is also central to the automatic decoding and translation of sign language [65, 80].

- The union of real-time graphics, computer vision, speech processing and synthesized sound is a new medium where people are immersed in a virtual environment - the Interactive Space [7, 11, 64, 1]. In such environments, human actions are the primary input, and the ability to reliably recognize such actions through video data eliminates the need for bulky body-position sensors. Furthermore, the generated media must be consistent with human perception to make the subjects comfortable. Therefore, the requirement of understanding the intention of subjects through sensors data is important.

- With the advent of large digital video libraries, intelligent video retrieval can be automated by representing and matching human actions [83, 15, 53, 2, 32, 58]. For instance, searching for video clips containing playing tennis sequences could perhaps only be realistically achieved through human activity recognition [83, 15, 53]. Furthermore, higher level semantic interpretations like video annotation can also be automated for movies and television shows through the analysis of human actions [32, 58].

With further improvement of computing power, there will be more and more applications, taking video as input, helping computers to understand human activities, improving the interaction between the computers and users, reducing the workload of security stuff and saving resources, and so on.

1.2 Classification of Actions and the Action Recognition Systems

1.2.1 Classification of Actions

Natural actions can be classified into three categories: *events*, *temporal textures* and *activities* [54]. Within this classification, *events* do not exhibit temporal or spatial repetition. It may be described by low-level or high-level descriptions. Low-level descriptions can be a sudden change of direction, a stop, or a pause, which provides important clues to the type of motion: while high level descriptions can be "opening a door", "starting a car", or more abstractly, "pick up", "push", "throw", etc. *Temporal texture* exhibits statistical regularity. Examples of temporal textures are ripples on water, leaves in wind, or a flag waving in wind. *Activities* consist of motion patterns that are temporally periodic and also possess compact spatial structure. Examples of activities are walking, running, jumping, etc.

1.2.2 Classification of Action Recognition Systems

There are two types of approaches for human action recognition: 3-D and 2-D. In 3-D approaches, 3-D models of human body and human motion are used. A projection of the model with a particular pose is then compared with each frame of the input video to recognize the action. The advantage of this type of approach is that since a 3-D model is used it is not ambiguous. However, it is computationally quite expensive [29]. Therefore, 3-D approaches are limited in some specific applications, such as dancing analysis and sign language recognition [10, 20]. Most of the 3-D approaches focus on events and activities, while no work has been done for temporal texture due to the complexity of various 3-D shapes of texture.

In 2-D approaches, no 3-D model is used, but only 2-D motions, e.g. optical flow and trajectory, are employed to compute features in a sequence of frames to recognize actions. The advantage of this approach is that it is quite simple. Great progress has been made to recognize different categories of actions by 2-D approaches. Here are some examples:

Recognition of temporal texture: To recognize the temporal textures, the statistical features of optical flow such as mean flow magnitude, standard deviation, the positive and negative curl and divergence, are used in [54]. A similar approach based on the statistical features of spatio-temporal gradient direction is used for classifying human activities, e.g. walking, running, and jumping [12].

Recognition of events: Events can be represented as "motion verbs" and then the recognition is performed by associating natural language verbs with the motion performed by the subjects. For example, the verbs describing the movement of vehicles are used to characterize the motion trajectories of the vehicles [38]. In order to recognize normal and abnormal behavior of a heart's left ventricular motion, an artificial intelligence system [70] was developed. In this AI system, the natural language semantic components were used to describe motion concepts. Generally, the recognition system needs some pre-knowledge to set up the syntax description for each action being performed.

Recognition of activities: The approaches to recognize human activities include region-based [18, 49, 54, 3], temporal trajectory-based [48, 79, 57, 25], part-based [6, 9, 35] or a combination of these [5, 26]. The approaches work based on either 2-D shape or motion. Usually, the recognition system involves some similarity measurement between the activities and the models. Moreover, since people execute actions with different speed, time-warping process is necessary.

1.2.3 View Invariance

The 2-D approaches discussed above are sensitive to changes in viewpoint. Since an action takes place in 3-D, and is projected on a 2-D image plane, the projected 2-D trajectory may vary depending on the viewpoint of the camera. This causes ambiguity in interpreting trajectories at higher levels. To avoid the ambiguity, these 2-D approaches require explicit models to handle different viewpoints. However, in most current work on action recognition, the issue of view-invariance has been largely ignored, resulting in methods that do not succeed in general situations. Recent attempts have alleviated the impact of viewpoint by explicitly recovering viewpoint transformations using homography [12, 3], or the general perspective model [67]. Seitz and Dyer [61] used view-invariant measurement to find the repeating pose of walking people and the reoccurrence of position of turning points.

In this thesis, we argue that finding a view-invariant representation of actions makes the recognition far more tractable and reliable. In this way, there is no need to explicitly recover the viewpoint transformations, which is vulnerable to noise and is not applicable in general. Furthermore, in order to generalize view invariant recognition of actions, we lay emphasis on the ability of the system to learn unsupervised. In our system, the events are detected and activities are further recognized based on both the events and the motion information of action agents.

1.3 Overview of Our Action Recognition Work

1.3.1 A Motion-Based Framework for Human Action Recognition

In this work, we propose a motion-based action recognition framework, which consists of three modules (Figure 1.1):

- Tracking – the extraction of relevant visual information from a video sequence;
- Representation – representing extracted information in a suitable form;
- Recognition and learning.

Within this framework we have made contribution by bringing forward:

- A view invariant representation, which reveals the real physical meanings of actions and is consistent with human perception.
- A spatio-temporal similarity measurement that compensates for variance of execution speed, which is also view-invariant.
- A clustering approach to group actions into different categories based on the view-invariant representation and the view-invariant measurement.

1.3.2 View Invariant Action Recognition System

In the recognition system that we implemented, a view-invariant representation of action is proposed consisting of *dynamic instants* and *intervals*, which is computed using the spatio-temporal curvature of an action trajectory. (The definition of action trajectory is given in Section 1.3.3) The dynamic instants are atomic units of actions representing changes in the direction and/or the speed. They can be reliably detected by identifying maxima in the spatio-temporal curvature of the action trajectory. We formally demonstrate that the dynamic instants are view-invariant, except in the limited cases of accidental alignment.

The proposed representation is then used to automatically learn and recognize human actions. The view invariant properties of instants are analyzed first. Then in order to match two representations for recognition purposes, we analyze a view-invariant measurement, which uses the eigenvalues of a matrix formed from the dynamic instants of two actions.

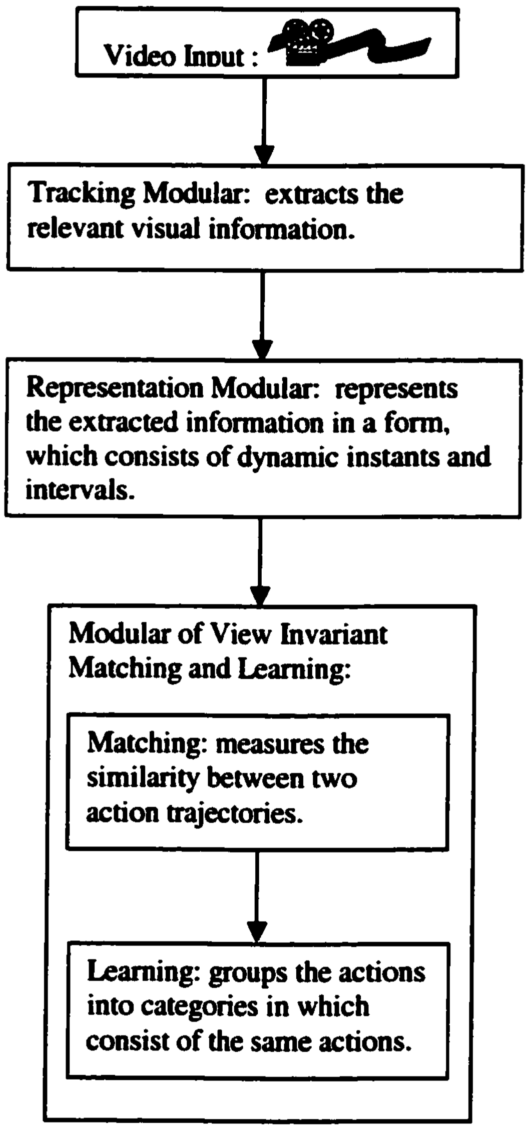

Video Inout :
Tracking Modular: extracts the relevant visual information.
Representation Modular: represents the extracted information in a form, which consists of dynamic instants and intervals.
Modular of View Invariant Matching and Learning:
Matching: measures the similarity between two action trajectories.
Learning: groups the actions into categories in which consist of the same actions.

Figure 1.1: The framework of the action recognition system.

We restate a theorem given in [61] in the context of matching actions. Furthermore, we propose a matching algorithm by dynamic programming to handle the varying execution speed. This matching algorithm uses the view-invariant measurement for computation, so the matching result is not affected by the change of viewpoint. Hence, not only the spatial but also the temporal information in the action trajectories are employed in the matching algorithm. Finally, the matching results are input to an unsupervised learning algorithm, which groups the instances of action based on the matching result. Although we use a straightforward learning algorithm, due to the effectiveness of our view-invariant action representation, the system successfully discovers the same actions performed by different people from different viewpoints.

1.3.3 Action and its Representation

In this dissertation, we focus our attention on human actions such as opening and closing overhead cabinets, picking up and putting down a book, picking up and putting down a phone, and erasing a white-board.

Within the current framework, actions are represented as the trajectory of an *action agent* in the sequence of image frames. The location and the orientation of the action agent are used as motion characteristics, and other features, such as area, eccentricity and solidity, may also be included.

Actions are represented in a multi-dimensional *spatio-temporal space.* Each point in the space is defined as (x, y, θ, t), where x and y are the image coordinates of the action agent, θ is the orientation of the action agent in image, and t is the timestamp (frame index number). Each motion characteristic corresponds to a dimension of the spatio-temporal space; and time is an independent dimension as well. During an action, the path along which the action agent moves through the space is referred to as the *trajectory* of action. Our action recognition system works based on the trajectories in the spatio-temporal space.

1.4 Video Temporal Alignment

For action recognition, it is necessary to measure the similarity between each pair of points along the two trajectories. Therefore, we need to find the temporal correspondence for the action trajectory points. Moreover, many applications, such as video mosaicing, video retrieval, image based modelling and rendering, video synthesis, and multi-sensor surveillance, require a computation of spatio-temporal alignment of video sequences. Some of these methods assume the input video sequences are already synchronized, while the other methods use an built-in expensive hardware that provides synchronization. We generalize the action recognition approach for video synchronization, such that the similarity between the movement of feature points in videos are measured and provides the clue for the temporal alignment for videos. This dissertation presents a novel approach of alignment of video sequences.

When a feature point moves in a 3D space with respect to time, due to either the camera motion or the actions of subjects,it generates a 3D trajectory: $\{(X_1, Y_1, Z_1), (X_2, Y_2, Z_2), \ldots, (X_t, Y_t, Z_t)\}$, where t is the time stamp. This 3D trajectory is projected as a 2D trajectory in the image plane: $Ptrj_1 = \{(u_1, v_1), (u_2, v_2), \ldots, (u_t, v_t)\}$. The relationship between a point (X_i, Y_i, Z_i) in 3D trajectory and its 2D projection (u_i, v_i) is defined as follows:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = P \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}, i = 1, 2, ..., t, \tag{1.1}$$

where P is the projection matrix (camera model).

Assume that the same motion is performed with a different speed (temporal extent), then we obtain another 3D trajectory: $\{(X'_{T(1)}, Y'_{T(1)}, Z'_{T(1)}), (X'_{T(2)}, Y'_{T(2)}, Z'_{T(2)}), \cdots, (X'_{T(t)}, Y'_{T(t)}, Z'_{T(t)})\}$,where $T(i)$ is a time warping function such that

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} = \begin{bmatrix} X'_{T(i)} \\ Y'_{T(i)} \\ Z'_{T(i)} \\ 1 \end{bmatrix}, i = 1, 2, ..., t$$

Now assume that the viewpoint of the camera has also been changed. Then the projection of this 3D trajectory to a 2D trajectory, $Ptrj_2 = \{(u'_1, v'_1), (u'_2, v'_2), \ldots, (u'_t, v'_t)\}$, is defined in the similar way:

$$\begin{bmatrix} u'_{T(i)} \\ v'_{T(i)} \\ 1 \end{bmatrix} = P' \begin{bmatrix} X'_{T(i)} \\ Y'_{T(i)} \\ Z'_{T(i)} \\ 1 \end{bmatrix}, i = 1, 2, ..., t.$$

Therefore, the problem of aligning video sequences is to discover the time-warping function, $T(i)$, for $i = 1, 2, \ldots, t$, using the information in two 2D trajectories, $Ptrj_1$ and $Ptrj_2$.

There are two crucial aspects of exploring correspondences between video sequences. First, the 2D trajectory is highly dependent on the viewpoint, that is the same 3D trajectory may look different in videos shot from the different viewpoints. Second, the same motion may have different speeds (temporal extents). The second problem becomes more complicated when the motion changes dynamically, such that the indices of corresponding frames are non-linearly related. This is very common in videos depicting human activities, since even the same person may perform the same activity with different speeds.

In this study, we propose a novel approach for aligning and matching of videos, which is based on the epipolar geometry and can discover the non-linear time-warping function, $T(t)$. Moreover, some applications are proposed for using this approach.

1.5 Organization of this Thesis

The organization of the rest of the thesis is as follows. In Chapter 2, the tracking of an action agent is discussed (Section 2.1), which includes a description on tracking the centroid of action agents (Section 2.1.1) and how to segment the region of an action agent in the image(Section 2.1.2). Section 2.1.3 discusses a smoothing approach that not only removes noise from the tracks but also preserves meaningful changes within the tracks. Next, the representation of action is discussed (Section 2.2). Section 2.2.1 discusses the related work of action representation. Then in Section 2.2.2, the psychological and theoretical aspects of motion and actions are analyzed. In Section 2.2.3, we discuss the spatio-temporal representation of action trajectories. In Section 2.2.4, we propose a mathematical model to overcome problems existing in previous approaches, which is followed by analysis of the proposed method's ability to find "instants". In section 2.2.5, a comparison between our method and previously proposed approaches is given. Section 2.2.6 discusses the view invariance of instant detection. In Section 2.2.7, we generalize the spatio-temporal curvature to multi-dimension characteristics-temporal space, so that the generalized curvature can capture the changes of any motion characteristics. Next, a view-invariant representation of action based on the *dynamic instants* is presented in Section 2.2.8.

In Chapter 3, we discuss how to recognize and learn the actions. Section 3.1 discusses previous literature on recognition approaches. In Section 3.2, we demonstrate the view-invariance property of instants, which is a very useful characteristic for action recognition, and how to recognize "picking up" and "putting down" actions. Section 3.3 discusses how the representations can be matched using the eigenvalues of a matrix formed from the dynamic instants of two actions. Section 3.4 discusses the view-invariant dynamic programming method to match two action trajectories from different viewpoints. Section 3.5 deals with learning human actions, and how the system can learn actions incrementally without any model.

In Chapter 4, we discuss the approach to find the temporal alignment between videos. In Section 4.1, we first discuss the related work and their limitations. Then in Section 4.2, a rank constraint of computation of fundamental matrix to measure the similarity between points from two trajectories is discussed for measuring the similarity between points of two trajectories. In Section 4.3, we discuss the view-invariant dynamic time warping. Finally, in Section 4.4, a coarse-to-fine refinement approach is proposed to improve the accuracy and robustness of alignment.

Finally, we present experimental results of the proposed action recognition system and the temporal alignment approach in Chapter 5, and summarize and propose future work in Chapter 6.

CHAPTER 2

TRACKING AND REPRESENTATION OF ACTION

In this chapter, we discuss the tracking and representation modules of our action recognition framework (Figure 1.1). The tracking module takes video frames as input, and tracks the motion characteristics of the action agents. The motion characteristics include orientation, and x and y position of the action agent. The representation module takes the output of the tracking component, and transforms the data into a form that is not only easy for processing but also emphasizes the actual physical events of the actions, so that the recognition component can perform its task reliably.

It goes without saying that effective representation of any data simplifies the task of recognition greatly. Because the actions take place in 3-D, and their 2-D projections may vary depending on the viewing directions, the same action may have very different trajectories, and trajectories of different actions may look the same. This may create a problem in interpretation of trajectories at a higher level of abstraction. However, if the representation of actions captures only characteristics, which are view-invariant, then the higher level interpretation can proceed without ambiguity. In this study, we demonstrate a view-invariant action representation approach, which is both consistent with human perception and also able to emphasize physical events during actions.

The tracking module will be discussed first, followed by details of the representation module.

2.1 Tracking

Before we proceed to the discussion of representation, we first discuss some well-known trackers, which are employed in our system to extract motion information from video data. Although much research effort has been expended towards robust tracking, we carefully chose the following trackers because they suit our purposes better than the others and have been found to be reliable. Since tracking is not our contribution, we just evaluate their merits and demerits and explain some implementation issues. Interested readers can refer to the original papers for more detailed theoretical analysis.

We extract relevant information by visual tracking. Tracking involves detection of regions of interest in image sequences, which are changing with respect to time, and also finding frame to frame correspondence of each region. If the volume of action agent is small compared with the distance to the camera, the region of action agent can be represented by its centroid. In most of surveillance system the centroid is sufficient for the requirement of the recognition system. The approach of tracking centroid is discussed in Section 2.1.1. However, for recognizing some complicated actions, the system may need more information to analyze the actions, such as the area and the orientation of the action agent. We discuss methods to segment action agents based on the color information in Section 2.1.2.

2.1.1 Mean-Shift Tracker

The trajectory of the action agent is generated by marking the agent's position in consecutive frames. We use the mean-shift tracker [16] to obtain the spatio-temporal trajectory of the action agent. The Mean-Shift Tracker is based on maximizing the likelihood of the model (the region of the agent in the first frame) color intensity distribution and the candidate (the regions in the new frames) color intensity distribution using

$$\rho(\mathbf{m}) = \sum_{u=1}^{n} \sqrt{q_u p_u(\mathbf{m})}, \tag{2.1}$$

where $\mathbf{m}$ is the centroid of the region, n is the number of bins in the distribution, and q_u and p_u are the weighted histograms of the model and the candidate respectively. The weights for the histograms are obtained using the Epanechnikov Kernel given by

$$K(\mathbf{x}) = \frac{1}{2} c_d^{-1}(d+2)\left(1 - \|\mathbf{x}\|^2\right), \tag{2.2}$$

where $\mathbf{x}$ is a d-dimensional vector, c_d is the volume of a d-dimensional sphere and $\|.\|$ is the magnitude operator. The kernel assigns smaller weights to pixels farther from the center. This increases the robustness of the distribution estimation since peripheral pixels are the least reliable. The centroid of the region of the action agent in the next frame is found using

$$\mathbf{m}_{new} = \frac{\sum_{\mathbf{x}_i \in S} w_i(\mathbf{m} - \mathbf{x}_i)}{\sum w_i} + \mathbf{m}_{old} \tag{2.3}$$

where S is the image patch and w_i are the weights computed using

$$w_i = \sum_{u=1}^{n} \delta\left(S(\mathbf{x}_i - u)\right) \sqrt{\frac{q_u}{p_u(\mathbf{m})}}, \tag{2.4}$$

where δ is the *Kronecker delta* function. In this way, the position of the centroid in the next frame is found efficiently.

The result of the mean-shift tracker is a trajectory, which is a spatio-temporal curve defined by: $(x[1]$,$y[1]$,$t[1])$, $(x[2]$,$y[2]$, $t[2])$, $\cdots$, $(x[n], y[n], t[n])$, where x and y are the image coordinates, and t is the time-stamp or the frame number. Figure 2.1 shows some tracking results (the body of a walking person) using the mean-shift tracker.

We use the mean-shift tracker, rather than background subtraction-based tracking approaches, since we allow camera panning, tilting or zooming in between actions (at the moment that there is no action in the field of view). The mean-shift tracker performs quite favorably when compared to other trackers, such as correlation-based tracking, skin detection, and optical flow. Correlation-based tracking typically suffers from slow speed, and

Figure 2.1: The results of the mean-shift tracker, the '+' shows the results calculated by the algorithm (frame 174, 176, 178, 180).

the ambiguity of local maxima. Skin detection based approaches detect the action agent (hand) region based on color in each frame, but do not prove to be robust to background interference. Optical flow based algorithms demonstrate frequent errors at boundaries of tracked objects. Thus, after empirical evaluation, the Mean-Shift Tracker was selected as our tracker of choice.

2.1.2 Region Detection

Since we are tracking regions that may rotate or scale, the centroid information may not be an adequate description of the action agent's motion. By segmenting out image *regions* of the action agent and computing their characteristics, such as orientation, eccentricity, or size, a fuller description of the the agent's motion can be obtained.

The region detection of the action agent works as follows: The Mean-Shift Tracker generates x and y coordinates of the centroid of the agent in the current frame. Then a patch of the frame, centered at the centroid, is taken. Within this patch, the region of the action agent is extracted by the skin detection method discussed in [37]. Figure 2.2 shows the flow chart of this approach.

Skin has unique color in most of images, especially in an office environment. This gives us a good chance to detect skin region from an image just by using pixel color values. The

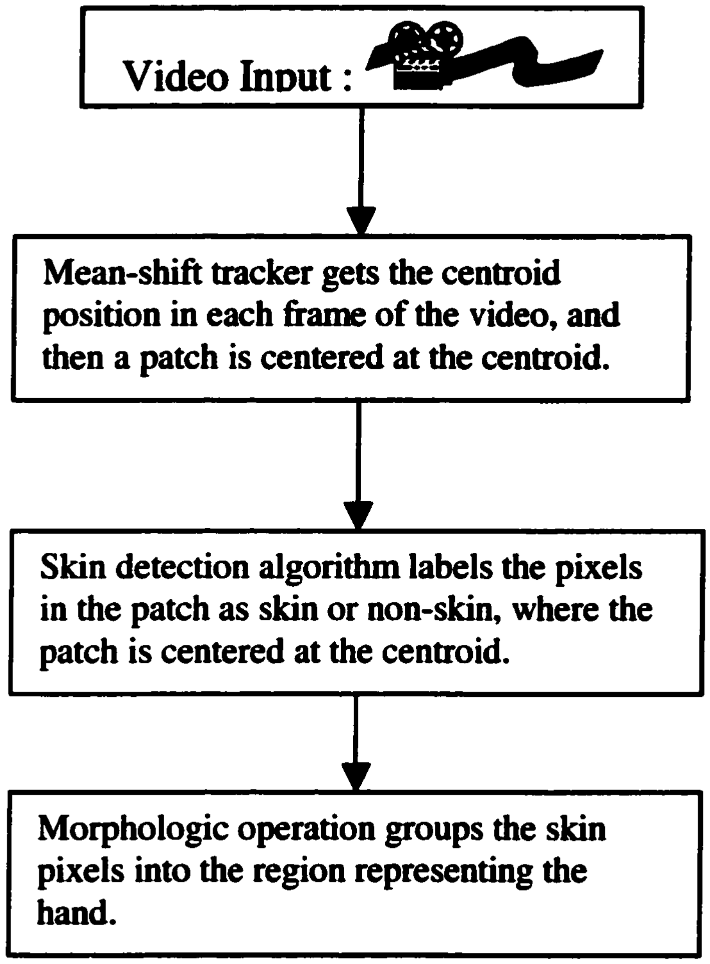

Video Inout :
Mean-shift tracker gets the centroid position in each frame of the video, and then a patch is centered at the centroid.
Skin detection algorithm labels the pixels in the patch as skin or non-skin, where the patch is centered at the centroid.
Morphologic operation groups the skin pixels into the region representing the hand.

Figure 2.2: The flow chart of region detection operation.

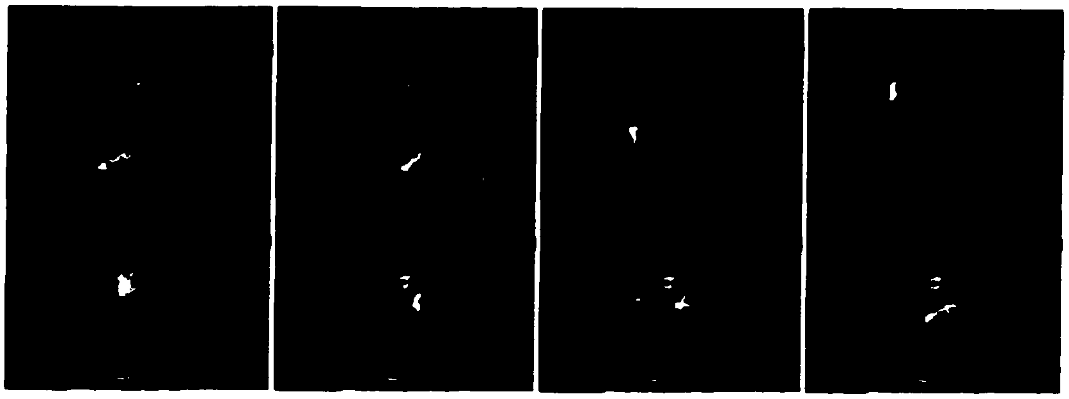
output of skin detection is a binary image, which has "1" indicating skin pixel and "0" indicating non-skin pixel. This skin detection method is based on the color predicates of the skin. The color predicates are computed to generate a look-up table from a training set of skin and non-skin regions. The incoming pixels are then labelled as skin or non-skin by searching the look-up table, allowing very high run-time speeds. Morphological operations are then applied to remove noise and to find the connected-component that best represents the region of the action agent. It is then straight-forward to recover the measurements of the region, such as the orientation and size. Figure 2.3 shows examples of a training image set, mask, and detection result.

Compared to using skin detection tracks directly, this method is far more robust for the following reasons: (1) Mean-shift is a robust tracker that can handle scale, partial occlusion, clutter, and rotations. (2) The detected region of the action agent is based on the tracking result from the mean-shift tracker and limited within the patch. Therefore, the pixels with skin-like color in the background are affected less by this method than by normal skin detection operation.

We do not choose contour-based region tracking, such as CONDENSATION [31], snakes [73] or level-set trackers [42], even though such trackers have been shown to follow the boundary of objects with high accuracy, because they need initialization of the entire contour first, which may not be feasible in video processing applications.

We also generalize the skin-detection algorithm to extract other action agents, such as the feet of walking people. Instead of using images containing skin regions for training, we can use images containing shoes to generate the color predicate, such that the incoming pixel is labelled as shoe or non-shoe. The mean-shift tracker fails to track one specific foot for a long walking sequence, since the feet occlude each other in every cycle and the two feet are almost identical. Therefore, we track the body of subject, and the patch is located with some offset to the mean-shift tracking result. The foot regions are detected using color predicates. The correspondence is solved using the R-S algorithm [56], which

(a) (b)



(c)

Figure 2.3: Skin detection. (a) A training image, (b) the corresponding mask, (c) skin detection result.

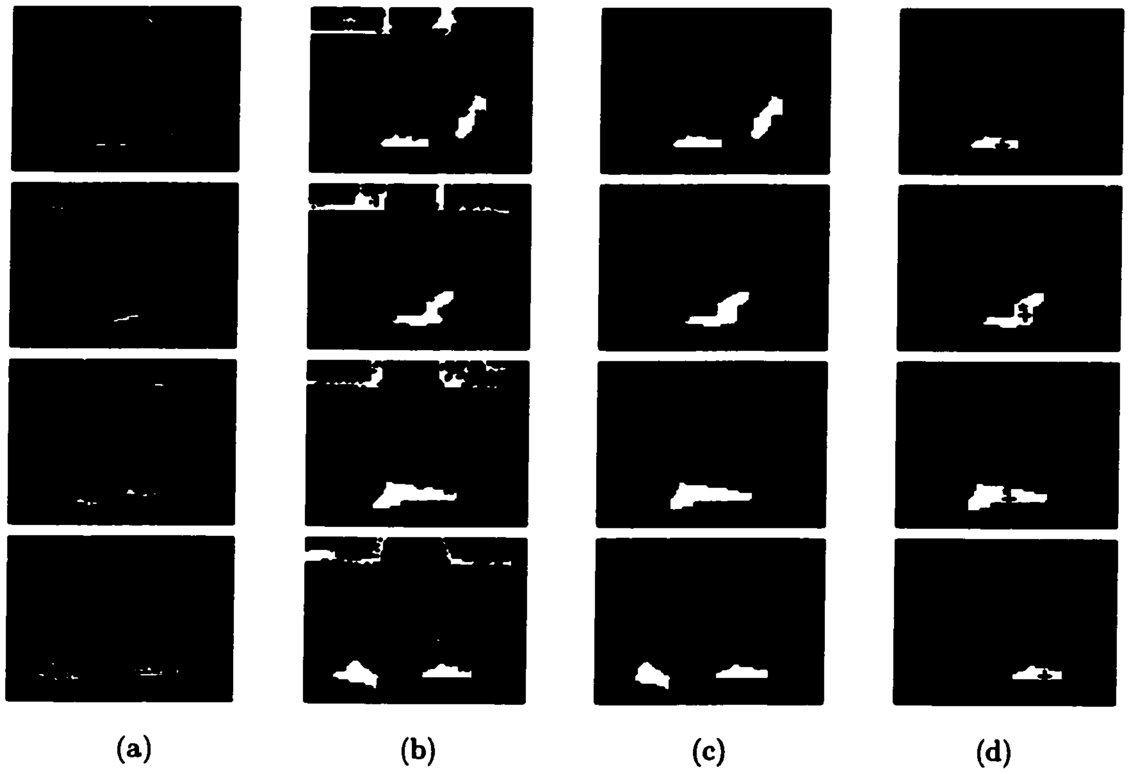


Figure 2.4: Shoes detection. Each row is for one frame (174, 176, 178, 180). The first column is the shoe's color probability image (brighter = higher probability), the second column is the thresholding results, the third column is the foot detection results, and the last column is labelling results of the left foot and the right foot.

considers both speed and direction of moving objects. Figure 2.4 shows some of the result of shoe detection and labelling results of the left foot and the right foot.

2.1.3 Action Trajectory Smoothing

Since it is unlikely that perfect tracks will be obtained from the tracking module, we have to perform smoothing operations to minimize the effect of noisy data. Although a lot of filter designs are available in related literature to reduce noise, such as low pass and mean filters, they are not suitable here, since they smooth out all the peaks, which may suppress meaningful changes in action. Instead, we use anisotropic diffusion to smooth the $x(t)$ and $y(t)$ coordinates of the trajectory.

Anisotropic diffusion was proposed in the context of scale-space [4]. This method iteratively smoothes the data (I) with a Gaussian kernel, and adaptively changes the variance of the Gaussian based on the gradient of a signal at a current point, as follows:

$$I_i^{t+1} = I_i^t + \lambda[c_N \bullet \nabla_N I + c_S \bullet \nabla_S I]_i^t \tag{2.5}$$

where $0 \leq \lambda \leq \frac{1}{4}$, (we choose 0.2 in our experiments), t represents the iteration number, and

$$\begin{aligned} \nabla_N I_i &\equiv I_{i-1} - I_i \\ \nabla_S I_i &\equiv I_{i+1} - I_i \end{aligned} \, .$$

The conduction coefficients are updated at every iteration as a function of the gradient:

$$\begin{aligned} c_N^t &= g(|\nabla_N I_i^t|) \\ c_S^t &= g(|\nabla_S I_i^t|) \end{aligned}$$

where $g(\nabla I) = e^{-(\|\nabla I\|/k)^2}$.

The constant k can be fixed either manually, or can be estimated from the "noise estimator" [32]. We choose $k = 10$ in our experiments.

The original diffusion algorithm proposed by Perona and Malik only applies to functions that have a 1D co-domain, such that F: $R^n \rightarrow R^1$ (grayscale image), rather than trajectory functions: T: $R^1 \rightarrow R^3$, which has 3 co-domains (x coordinates, y coordinates, and orientation θ). If we apply the smoothing method on each component of the trajectory separately, then the correlation between different component is lost. So we need an algorithm that works on the vector data $(x[t_i], y[t_i], \theta[t_i])$.

The KL (Karhunen-Lo'eve) transform is a mathematical way of determining that linear transformation of a set of data in N-dimensional space which exhibits the properties of the data most clearly along the coordinate axes. Along the new axes the data variances are extremes (maxima and minima), and uncorrelated. Therefore, we can apply the P-M smoothing method on each transformed dimension of data separately. Furthermore, for later processing, we use the same KL transform matrix to get the smoothed trajectory data back. By this way. the correlation between different dimension is kept.

The steps of the empirical method we use are: (1) perform the KL (Karhunen-Lo'eve) transform on the raw data so that the correlations between different dimensions are minimized; (2) perform Perona-Malik smoothing on each dimension of the transformed data, (3) transform the smoothed data back to original data coordinates.

Figure 2.5 shows a raw trajectory (a) and the result of anisotropic diffusion of x and y coordinates (b). Notice that now the trajectory is much smoother and important changes during the action are preserved.

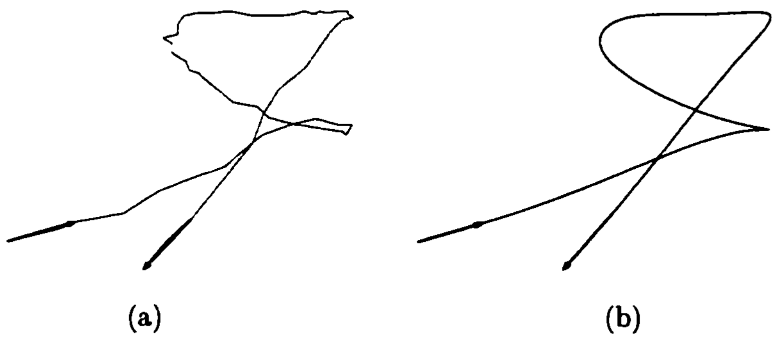


Figure 2.5: (a) "Opening overhead cabinet" trajectory (b) smoothed version of the trajectory.

2.2 Action Representation

In review, techniques have been described to extract motion and region characteristics of the action agent from raw video data. In this section, we describe a compact representation for the motion and region characteristics of the action agent, which is not only computationally feasible but also emphasizes the important physical characteristics of the action. In our representation, an action is represented as a sequence of *dynamic instants* and *intervals*. A *dynamic instant* is an instantaneous entity that occurs for only a single frame, and represents an important change in motion characteristics. *Interval* is defined as the time period from one instant to the next. We will discuss some related work in Section 2.2.1, followed by relevant research in psychology about action representation and recognition in Section 2.2.2, and finally, a novel method of detecting important changes during the action will be proposed in Section 2.2.3.

2.2.1 Related Work

The earliest research used simple position, velocity and acceleration data to characterize a complete action sequence. Although these data representations captured the characteristics of an action at each moment, they did not emphasize important physical events during the action. Therefore, recognition of actions was a non-trivial task that required a complex training process for the recognition algorithm. For example, Hidden Markov Model (HMM) based action recognition systems may need hundreds of examples for the training process. This is a significant reason why HMMs are not suitable for video surveillance applications, since it is difficult to collect many examples for some suspicious or dangerous actions. It is desirable to detect unusual actions which are (by definition) actions that have not been encountered often or ever. Researchers have been developing action representations for over a decade and some subsequent progress has been achieved.

Davis and Bobick [2] proposed a view-based approach for action representation. For each action sequence, a binary motion-energy image (MEI) is generated to represent where motion has occurred in an action sequence. Furthermore, a motion history image (MHI), which is a scalar-valued image where intensity is a function of recency of motion, is generated. These two images are combined together to form a temporal template. Rosales and Sclaroff [29] extended Davis' work further. Instead of just considering 2-D spatial information, an extended Kalman filter was used to generate trajectories from the video sequence. Although the depth information was estimated up to a scale factor, it still helped in locating the person and solving occlusion problems.

One obvious drawback of this approach is that only 2-D information is employed. The MEI and MHI will change dramatically when the viewing direction changes, i.e. it is view-variant, and as a result this approach cannot succeed for the general case.

Rangarajan et al. [25] proposed a scale-space based action trajectory representation method. The salient changes of action motion characteristics in different scales are recorded and named as Trajectory Primary Sketch (TPS) for actions. Each type of motion has its own unique TPS. Therefore, the recognition is based on the shape of the TPS. Davis et al. [19] proposed another motion representation method by fitting sinusoidal models to the motion characteristics. The sinusoidal model contains amplitude, frequency, phase, and translation parameters. Based on the sinusoidal model coefficients, the motion can be classified into different categories, because each family of motion has its unique pattern of coefficients. For example, when drawing a figure eight, the frequency of y is double the frequency of x. This approach can recognize up-and-down, side-to-side, undulate, circle, spiral, and loop motion patterns.

The problem with this approach is that only some types of actions can be represented well. The TPS research studies only 4 types of action motion trajectories: translation, rotation, projectile, and cycloid. Sinusoidal model research studied eight types of trajectories: up-and-down, side-to-side, circle, spiral, undulate, loop, figure-8, and u-shuttle. For a new action, the systems require setting up a model manually, so it is fairly hard to use these type of approach without extensive preparation. Moreover, both of these approaches are still working on 2-D information, therefore again, they fail when the viewing direction changes.

Assuming an action can be represented by a set of bases, Yacoob and Black [30] proposed a method for modelling and recognizing activities. In their paper, they claim that the motion characteristics of body parts, for example, horizontal translation of arm, vertical translation of torso, and rotation of thigh, can be represented by a weighted summation of bases using Principle component analysis (PCA) methods, so the recognition can be based on the coefficients of PCA. Their second contribution is a new method for computing PCA – using a robust regression method. However, their view-invariant capability is limited to scaling, and temporal warping, and as a result, their method is not view-invariant in general cases.

Tanveer Syeda-Mahmood [67] proposed a representation of action with a generalized cylinder. For example, in an action with the hand, the hand moves through 3-D space, and the space volume swept by the palm can be represented as a cylinder. Therefore, the same actions should have the same cylinder, while different actions have different cylinders. The drawback of this work is that the 'action-cylinder' requires very heavy computation. Furthermore, the 'action-cylinder' method has to provide the corresponding features in two cylinders manually for the recognition module at the next level. On the contrary, we analyze the motion characteristics to segment a long action sequence into consistent atomic parts, and this segmentation result contains implicit correspondence between two action trajectories. Moreover, the action cylinder needs much more accurate tracking, which is not yet available in normal surveillance systems.

In [51], Parameswaran et al. presented an approach to detect human action, in which canonical body poses are modelled in a view-invariant manner to enable detection from a general viewpoint. They employ 2-D invariants to recognize canonical poses of the human body. This approach requires the complete tracks of multiple features on human body (left hand, right hand, feet, head), so it is not applicable for general surveillance systems, because occlusion of human body parts is common. Even the authors admitted that the experiments are done by using motion capture data instead of the real video sequences.

2.2.2 Psychology Research about Actions

Contemporary psychology has provided an instructive analysis of the atomic units of actions that are of substantial value to perception. These atomic units of actions are defined as motion events due to the significant changes in motion trajectories [60]. The changes are *start*, *pause*, or *stop* of motion and a sudden change in the direction or the speed of the motion. *Start* is the boundary (time instant) at which the object changes from the stationary state to the moving state. Similarly, *stop* is the boundary from the moving state to the stationary state. The results due to a change in the force applied to the object during the activity, which is named a *Dynamic instant,* causes a boundary (change) in the direction and/or speed. Since *pause* is a combination of stop and start, we will not treat it as an additional class of motion boundary.

In [82], Zacks showed that people tend to divide activities at locations that correspond to changes in the physical features (speed and direction), and this division of activities constitutes basic actions that are primitive actions. This conclusion is strengthened by a set of studies on the role of events in action comprehension [52, 47]. In [52], Parish et al. described American Sign Language sequences in terms of the activity index, which is obtained from the changes in position of the hands. They selected the frames corresponding to local minima of the activity index as critical event boundaries of the sequences. Newtson et al. [47] conducted experiments on human perception and organization of events. In their experiments, they selected representative frames (shot boundaries) from movies and analyzed the descriptions about the actions from these representative frames by the movie observers. When the representative frames were presented to the observers in sequence,

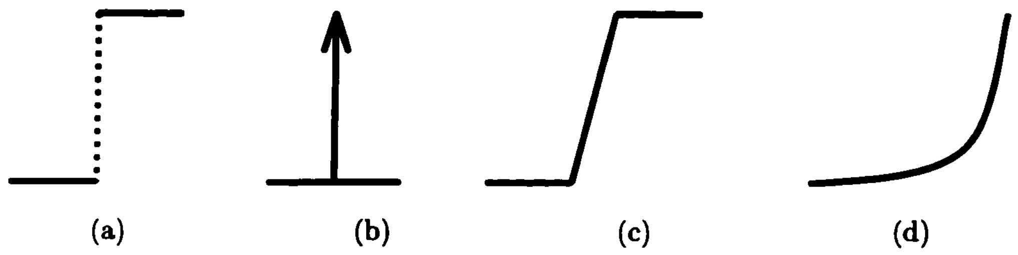


Figure 2.6: Possible functions for force and speed; (a) Non-smooth-discontinuous (step function), (b) Non-smooth-discontinuous (impulse function), (c) Non-smooth-continuous (ramp function), (d) Smooth-continuous.

they had more accuracy and confidence in their description compared to the presentation of these frames out of order.

There can be two types of forces applied to the object: continuous and discontinuous. A discontinuous force (force being a function of time) can be either a step (Figure 2.6a) or an impulse (Figure 2.6b) function; whereas a continuous force can be a non-smooth (ramp, Figure 2.6c) or a smooth function (Figure 2.6d). According to Newton's second law of motion

$$\mathbf{F} = m\mathbf{a} \tag{2.6}$$

where F is force, m is mass and a is acceleration. Assuming mass remains constant, any type of change in force results in a proportional change in acceleration. Since force is not a measurable quantity in images, we focus on acceleration and speed in our discussion.

Analysis of human perception shows that humans successfully perceive starting and stopping instants emerging from any type of acceleration (continuous or discontinuous) applied to the object [60]. Similarly, humans are also able to perceive dynamic instants resulting from a discontinuous (step or impulse) change in acceleration. However, chnages that result from a continuous change in acceleration are not observed by humans [60]. We summarize this in Table 2.1, where the first and second columns show the possible speed

Table 2.1: Classes of motion boundaries, based on the types of speed functions, and whether they are detectable or undetectable by human observers.

	Continuous speed	**Discontinuous speed**
Start instant	Detectable (Fig. 2.10a, b)	Detectable (Fig. 2.10c)
Stop instant	Detectable	Detectable
Dynamic instant	Undetectable	Detectable (Fig. 2.12)

functions: continuous and discontinuous respectively, and the rows show the instants: start, stop, and dynamic. Among the six categories of motion tabulated in Table 2.1, in Figures 2.10 and 2.12 we show the five categories of motion detected by human observers (stop instants can be categorized as the inverse of start instants).

In the next few sections, we propose a mathematical model to detect the instants described in this section.

2.2.3 Spatio-temporal Action Trajectory

Human actions take place in three dimensions as the action agent moves in space with respect to time. Figure 2.7 illustrates a typical trajectory in which an individual opens an overhead cabinet, in which centroids are displayed as circles; the red centroid representing the final centroid, and the black circle representing the earliest one.

We shall begin with a study of motion in 1D space. Figure 2.8 shows 1D particle motion, in which a point moves along a line. Its position with respect to time can be represented as $\{x_0, x_1, x_2, \ldots, x_t\}$, where t is the time index. At the position x_1 and x_2, the speed values are v_{x1} and v_{x2} respectively (Figure 2.8) . In x and t space, the slope is computed as: *slope* $= \Delta x/\Delta t$, and as a result the slope is equivalent to the particle's speed. Note, the trajectory is a straight line *iff* speed $v_{x1} = v_{x2}$. Moreover, the bigger the difference between

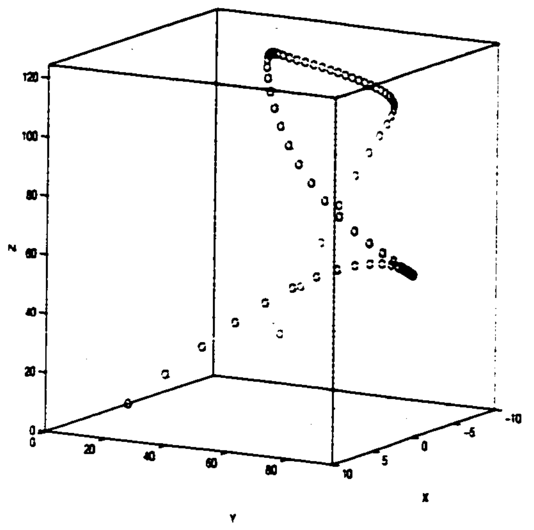


Figure 2.7: An opening overhead cabinet action trajectory.

speed v_{x1} and v_{x2} (acceleration). the larger the turning at the position of x_1 will be. This results in a larger curvature value in spatio-temporal space. We can therefore conclude that the discontinuity of speed is detected by *spatio-temporal curvature*. From Table 2.1. readers can observe that this corresponds to dynamic discontinuous instants. The *starts* and *stops* are discussed in Section 2.2.4.

The benefit of using *spatio-temporal curvature* to detect changes become more apparent in 2-D motion. in which not only can the speed change, but the direction can change as well. Figure 2.9 shows a motion that contains both speed and direction changes. Both changes correspond with turnings in spatio-temporal space. Therefore, both changes can be captured using spatio-temporal curvature at the trajectory points.

Sometimes, in addition to the position, we need to measure other characteristics of a moving object, which can bring us more information about the movement, such as orientation and eccentricity. Moreover, these characteristics can be put into the generalized

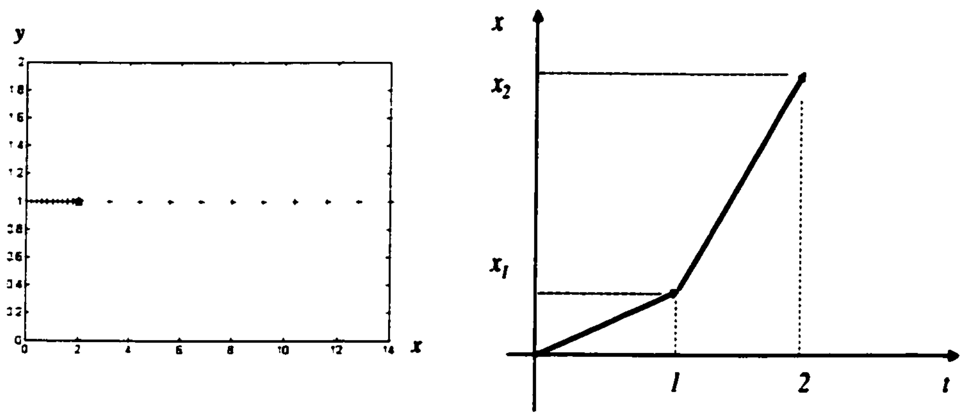


Figure 2.8: A 1D motion. a) The motion trajectory. Each '+' represents the position at a moment. '★ ' is the position the speed changes. b) The trajectory in spatio-temporal space. The turning point represents the speed change.

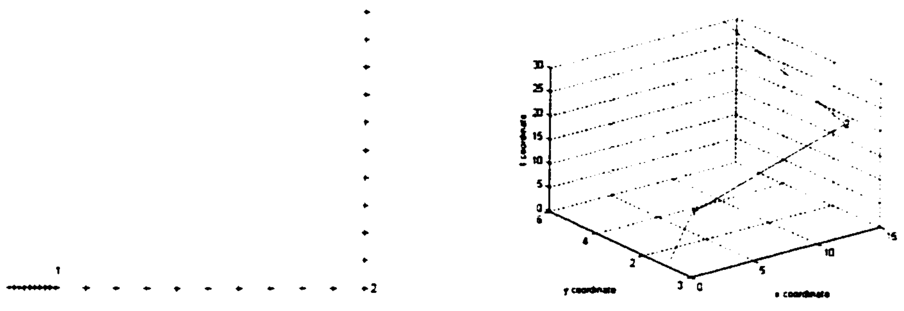


Figure 2.9: A 2-D motion. a) a motion trajectory in 2-D. Each '+' represents the position at a moment. 1 is the position the speed changes and 2 is the position the direction changes. b) The trajectory in spatio-temporal space. The first turning point represents a speed change in a) and the second turning represents a direction change in a).

spatio-temporal curvature computation, by treating each characteristic as an individual dimension of spatio-temporal space. This allows us to capture all the important changes of different characteristics with one curvature quantity. In Section 2.2.7 we will demonstrate examples of walking sequences, in which the subject's feet were tracked, and the centroid position (x,y) of the feet and orientation of the feet region are used to detect the important changes during walking. The results validate our claim that the spatio-temporal curvature gives a uniform framework for detecting changes in motion characteristics.

2.2.4 Computing Spatio-temporal Curvature

We have discussed the motivation behind using spatio-temporal curvature to detect important changes during actions. The spatio-temporal curvature of a trajectory is computed by a method described by Besl and Jain [4]. In this case, a 1D version of the quadratic surface fitting procedure is used. The spatio-temporal curvature, k is given as follows:

$$k = \frac{\sqrt{A^2 + B^2 + C^2}}{\left((x')^2 + (y')^2 + (t')^2\right)^{3/2}}. \tag{2.7}$$

where

$$A = \begin{vmatrix} y' & t' \\ y'' & t'' \end{vmatrix}, B = \begin{vmatrix} t' & x' \\ t'' & x'' \end{vmatrix}, C = \begin{vmatrix} x' & y' \\ x'' & y'' \end{vmatrix}$$

The notation $|\cdot|$ denotes the determinant, and

$$\begin{aligned} x'(t) &= x(t) - x(t-1), \\ x''(t) &= x'(t) - x'(t-1). \end{aligned} \tag{2.8}$$

Since the time interval is constant in video sequences, i.e. t=1, 2, 3,..., so t'=1, and t''=0. Equation 2.7 becomes:

$$k = \frac{\sqrt{y''^2 + x''^2 + (x'y'' - x''y')^2}}{\left((x')^2 + (y')^2 + 1\right)^{3/2}}. \tag{2.9}$$

The spatio-temporal curvature of the "opening overhead cabinet" trajectory is shown in Figure 2.13a. The local maximums, which correspond to changes (instants), are detected by zero-crossings of first derivative of spatio-temporal curvature values, and the changes are marked as '*' on the trajectory in Figure 2.13b.

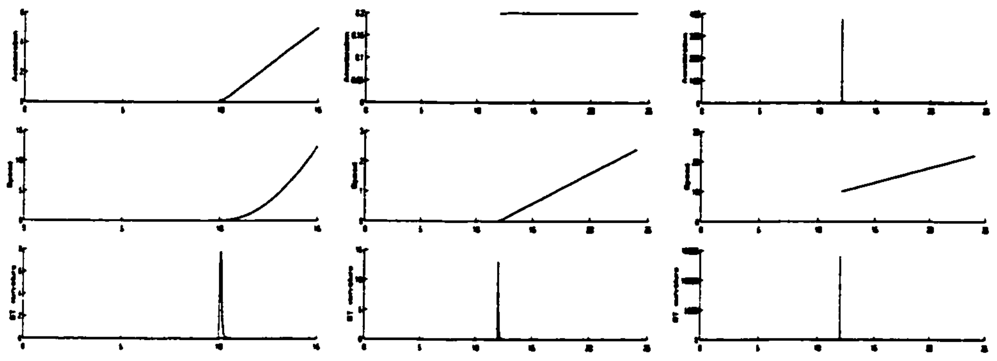
Instants, which are elementary components of motion, segment the motion trajectory into *intervals*. In Section 2.2.2, it was discussed that human observers are able to perceive *start*, *stop* and *dynamic instants*.

For simplicity, we continue the analysis of spatio-temporal curvature in the one dimensional case. One-dimensional temporal curvature, using Equation (2.7), is given by

$$\kappa_{1D} = \frac{|x''(t)|}{(x'(t)^2 + 1)^{\frac{3}{2}}}, \tag{2.10}$$

where $y(t)$ is set to a constant value, ie. $y'(t) = y''(t) = 0$. A quick analysis of equation (2.10) can be done by looking at the effect of the speed vector, $\mathbf{x}'(t)$, on the curvature with respect to the acceleration. Due to the higher exponent of speed ($\frac{3}{2} > 1$), and acceleration being the derivative of speed, an increase in speed will lower the value of curvature exponentially. To see the effect of speed and acceleration on detecting the motion boundaries (*instants*), we analyze five possible motion classes, which are observed by humans as motion boundaries (*instants*) listed in Table 2.1. These boundaries are shown in Figures 2.10 and 2.12.

In Figure 2.10, we show examples of *start instants* due to continuous and discontinuous speed changes. In Figure 2.10 (a) and (b), before the object starts its motion the spatio-temporal curvature given in Equation (2.7) is $\kappa_{1D} = 0$. At the moment when the object starts moving, the curvature becomes κ_{1D} >0. Since the effect of an increase in the speed is exponential in Equation (2.7), the curvature reduces to κ ≈0, which results in a peak



(a) (b) (c)

Figure 2.10: Three examples of *start instants* due to smooth continuous (a), non-smooth continuous (b), and discontinuous (c) speed, which changes the state of object from rest to active. Corresponding accelerations and spatio-temporal curvatures (κ_{1D}) are also shown.

in κ_{1D}. A similar effect also holds for Figure 2.10c, where the motion starts due to a discontinuous force on the object. Peaks in curvature for all start instants relate to the motion boundaries (*instants*) that humans are also able to perceive.

So far we have shown how spatio-temporal curvature captures the psychological motion boundaries that occur due to *start* or *stop instants*. The third category of motion boundaries, which is independent of starts and stops, is the *dynamic instant* that occurs due to the change in force applied to a moving object (active state of the motion). Humans, however, are only able to perceive one type of *dynamic instant*, as was discussed in Section 2.1. The diagrams in Figure 2.12 show the types of *dynamic instants* that humans are able to perceive, which are also captured by the spatio-temporal curvature k_{1D}. In Figure 2.12, we show a complete set of speed discontinuities of an object that result in *dynamic instants*, rather than showing a complete set of the infinite number of ways that force changes can be applied to an object.

The construction of this complete set of speed discontinuities is obtained as follows: let s_a and s_b be the speed discontinuity values as shown in Figure 2.11. The speed function

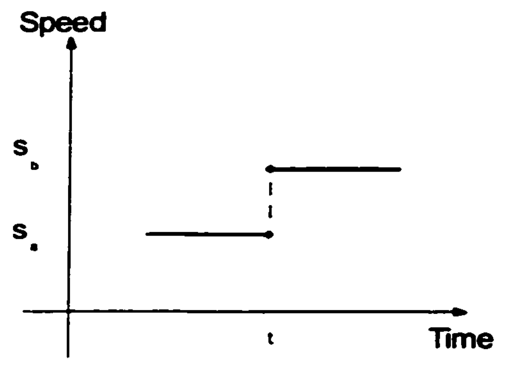


Figure 2.11: Discontinuity values in the speed function.

before or after the discontinuity can be either increasing or decreasing. We represent increasing speed by $s^{\uparrow}$ and decreasing speed by $s^{\downarrow}$. Thus one of the speed discontinuities can be given by: $(s^{\uparrow}s^{\uparrow}; s_a < s_b)$, which is interpreted as an increase in the speed before the discontinuity, an increase after the discontinuity, and at the discontinuity $s_a < s_b$. Other discontinuities are: $(s^{\uparrow}s^{\uparrow}; s_a > s_b)$, $(s^{\downarrow}s^{\uparrow}; s_a > s_b)$, $(s^{\downarrow}s^{\uparrow}; s_a < s_b)$, $(s^{\downarrow}s^{\downarrow}; s_a > s_b)$ $(s^{\downarrow}s^{\downarrow}; s_a < s_b)$, $(s^{\uparrow}s^{\downarrow}; s_a < s_b)$, and $(s^{\uparrow}s^{\downarrow}; s_a > s_b)$.

2.2.5 Previous Approaches

Spatial curvature is commonly used in 2-D shape analysis. When the time information is ignored in the spatio-temporal curvature, we simply get spatial curvature. In this case, the time interval is 0, therefore $t' = t'' = 0$, and Equation (2.7) reduces to spatial curvature:

$$c = \frac{\sqrt{(x''y' + y''x')^2}}{(x'^2 + y'^2)^{\frac{3}{2}}} \tag{2.11}$$

Comparing Equation 2.9 and 2.11, we see that the denominators only differ by a constant, which is 1. But the numerators are very different; the spatio-temporal curvature has $\sqrt{y''^2 + x''^2 + (y''x' + y'x'')^2}$, while the spatial curvature has only $\sqrt{(y''x' + y'x'')^2}$. There-

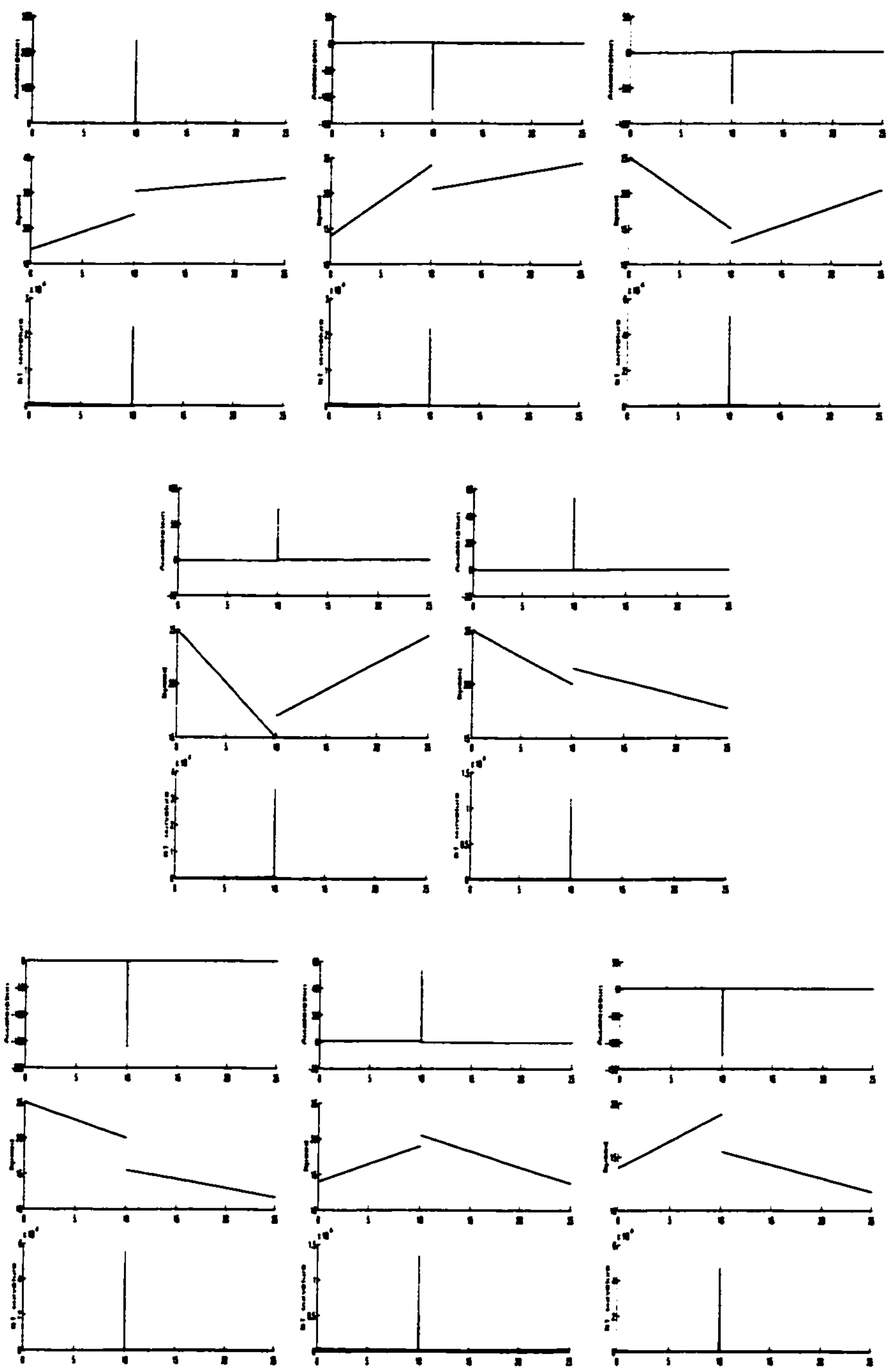


Figure 2.12: Eight possible *dynamic* boundaries that occur due to the non-smooth step change in speed. Corresponding accelerations and spatio-temporal curvatures are also shown.

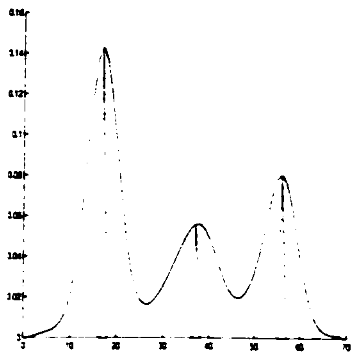
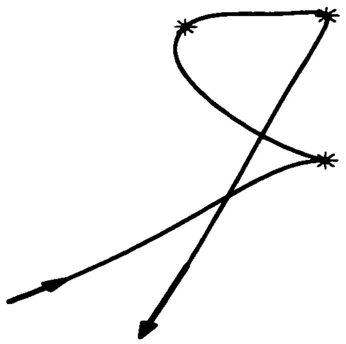


(a) (b)

Figure 2.13: Spatiotemporal curvature and detected maxima (dynamic instants) in the "opening overhead cabinet" trajectory. The actually sequence is in the experiment section.

fore, when the speed is slow, for example when the action agent stops to touch an object, the normal spatial curvature will not be able to reflect acceleration change since x' is small. On the contrary, the spatio-temporal curvature will generate a large value because of the x'' components in the computation. According to the psychology research, the start and stop are very important events for the human perception of action [60], thus the spatio-temporal curvature is much more suitable for an action recognition system.

For extracting the instants from a 2-D projected motion trajectory, Rubin and Richards [60] considered the change of velocity in polar coordinates, where the magnitude of the velocity vector is the speed, $s(t)$, and the angle is the direction, $\mu(t)$, of motion. In their approach for obtaining the motion boundaries (which we call instants), they compute the zero-crossings of the second derivatives of both $s(t)$ and $\mu(t)$. Since the changes in velocity and speed are not always temporally aligned, the primary problem with this approach is combining these two pieces of information in a meaningful manner. For example, the union of speed and direction instants results in too many instants, while the intersection results in too few instants. This issue was not addressed by Rubin and Richards.

Detection of instants was also addressed by Gould and Shah [25]. Instead of using polar coordinates, they used the velocity vector $v(t) = [v_x, v_y]^T$ for instant detection. They also introduced the Trajectory Primal Sketch (TPS), where significant changes are identified by the strength of zero-crossings of v_x and v_y computed at various scales. This process results in a set of TPS contours, where each contour corresponds to an instant. Their recognition module takes the instant detection results from both v_x and v_y and determines the type of object motion. However, this approach still suffers from the temporal alignment problem.

2.2.6 View Invariance of Instants

Since an action takes place in 3-D, and is projected onto a 2-D image, depending on the viewpoint of the camera, the projected 2-D trajectory may vary. Thus, the same action may have different trajectories, and trajectories of different actions may look the identical. This can create a problem in interpretation of trajectories at higher levels. Therefore, we need to study the attributes of detection of instants. From Section 2.2.4, the spatio-temporal curvature detects start, stop and dynamic instants, which correspond with the speed increasing from zero to nonzero, speed decreasing from nonzero to zero, and speed discontinuities. We propose that start, stop and discontinuity are view-invariant characteristics of a 2-D projection of a 3-D trajectory.

Theorem 1 The continuities and discontinuities in velocity and acceleration in the 3-D trajectory of a moving object are preserved in 2-D image trajectories under a continuous projection function.

The proof is as in [60]:

Notation: $\vec{p}$ is the velocity vector of a 3-D motion, and I is any projection function that maps space to an image plane such that every image point is a one-manifold in space (line of sight).

Claim: Let $\vec{p}(t) : R^1 \rightarrow R^3$ and $I : R^3 \rightarrow R^2$ be continuous functions such that $I^{-1}(x, y)$is a one-manifold in R^3. Then $\vec{p}(t)$ is continuous $\Rightarrow$ $I(\vec{p}(t))$ is continuous, and $I^{-1}(\vec{p}(t))$ is continuous $\Rightarrow$ (almost always) that $\vec{p}(t)$ is continuous.

Proof: Since the composition of continuous functions are continuous, we have that the continuity of $\vec{p}$ implies the continuity of $I(\vec{p})$. Suppose that at some t_0, $I(\vec{p}(t_0))$ is continuous, but that contrary to the claim we are to prove, $\vec{p}$ is discontinuous at t_0. We consider only step discontinuities. Let $\vec{p}^{+}(t_0) = \lim_{t \rightarrow t_0+} \vec{p}(t)$, and $\vec{p}^{-}(t_0) = \lim_{t \rightarrow t_0-} \vec{p}(t)$. A step discontinuity at t_0 implies $\vec{p}^{+}(t_0) \neq \vec{p}^{-}(t_0)$. But I assigns the same R^2 value only to points in R^3 lying on a particular one-manifold. There is zero probability that the two points $\vec{p}^{+}(t_0)$ and $\vec{p}^{-}(t_0)$ lie on one of those special one-manifolds.

Corollary: Let $\vec{p}(t) : R^1 \rightarrow R^3$ be a vector-value function, and let I be a reasonable and continuous imaging function as before. Then, almost always, $I(\vec{p}(t))$is discontinuous at t_0 iff $\vec{p}(t)$ is discontinuous at t_0.

Proof: Contrapositive of claim above.

Theorem 2 *The zeros in velocity in the 3-D trajectory of a moving object are preserved in 2-D image trajectories under a continuous projection function*

Claim: Let $\vec{p}(t) = 0$ and $I : R^3 \rightarrow R^2$ be continuous functions such that $I^{-1}(x, y)$is a one-manifold in R^3. Then $\vec{p}(t)$ is $0 \Rightarrow I(\vec{p}(t))$ is 0, and $I^{-1}(\vec{p}(t)) = 0 \quad \Rightarrow$ (almost always) that $\vec{p}(t) = 0$.

Proof: Similar to last claim.

Corollary: Let $\vec{p}(t) \neq 0$ and $I : R^3 \rightarrow R^2$ be continuous functions such that $I^{-1}(x, y)$is a one-manifold in R^3. Then $\vec{p}(t) \neq 0 \quad \Rightarrow I(\vec{p}(t)) \neq 0$, and $I^{-1}(\vec{p}(t)) \neq 0 \quad \Rightarrow$ (almost always) that $\vec{p}(t) \neq 0$.

Proof: Contrapositive of claim above.

From this discussion, we reach the **conclusion** that *instants*, which are the maxima in the spatio-temporal curvature of a trajectory, are view-invariant, except in the limited cases of accidental alignment. By accidental alignment, we mean a view direction, which is coincident to the plane where the action is being performed. In this case, the centroids of the action agent in consecutive frames are projected at the same position in the image plane, resulting in a 2-D trajectory, which is essentially a single point.

Figure 2.14 shows some projections of 3-D synthetic data, with different viewing directions in (a), (c), and (e). The centroids of later motions are the denoted by a redder circles while the centroids of earlier motions by blacker circles. The corresponding spatio-temporal curvature values are shown in (b), (d) and (f). The spatio-temporal curvature value for the current trajectory is in red, while the previous ones are in blue. Note that although the value of the spatio-temporal curvature at each point changes when the viewing direction changes, the positions of local maxima of spatio-temporal curvature are consistent.

Examples of *instants* in the trajectories of opening and closing the overhead cabinet are given in Figure 2.15 for different views. Even though these trajectories look quite different, three *dynamic instants* for every viewpoint are correctly detected by the proposed method.

It is important to handle outliers that may arise. There are two principle sources of outliers during this representation phase. The first source of outliers is due to the discrete nature of video sequences. Under ideal conditions, if there is a discontinuity, the spatio-temporal curvature will be a Dirac delta function since the numerator of the Equation (2.7) will be infinite. However, for video sequences, the impulse degenerates to a peak in the spatio-temporal curvature values. In addition, the spatio-temporal curvature is not constant; it fluctuates when the motion is changing smoothly. Therefore, it is hard to distinguish the peaks generated by discontinuities and those by smooth motion when the video frame rate is not high enough. The second source of outliers is caused by the projection of the 3-D trajectory onto the 2-D image plane. The projection may change the property of a smooth 3-D curve, such that the spatio-temporal curvature may represent a peak even when the object is under smooth motion. This too may generate a false detection. Fortunately, the viewing direction only affects the intervals that have continuous second

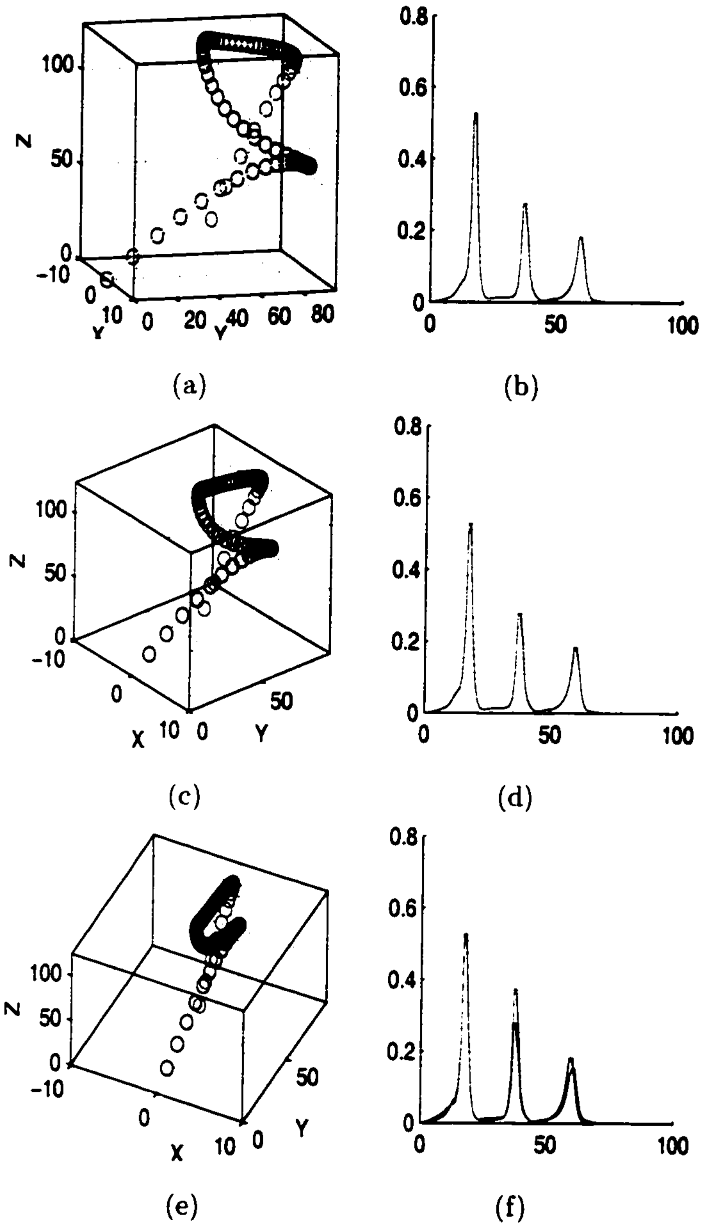


Figure 2.14: A synthetic trajectory in 3-D space, projected with different viewing directions in (a), (c), and (e). The latest centroid is the most red circle while the oldest centroid is the most black circle. The corresponding spatio-temporal curvature values are (b), (d) and (f). The spatio-temporal curvature value for the current trajectory is in red, while the previous ones are in blue.

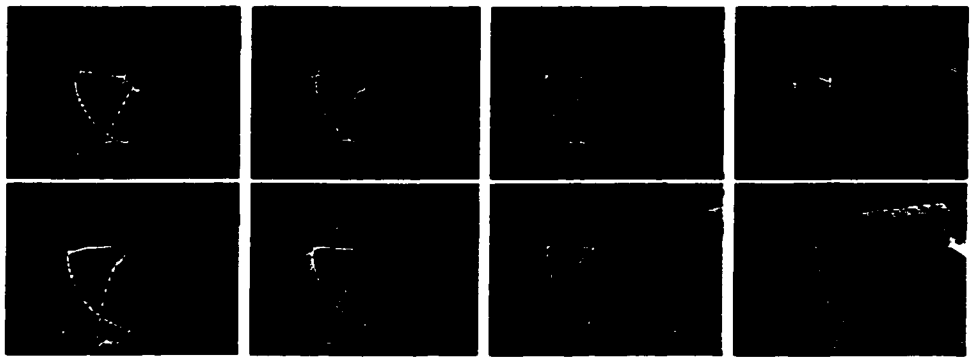


Figure 2.15: Trajectories from different viewpoints for opening (top) and closing (bottom) overhead cabinet action. Both the opening and closing actions in the same column are taken at the same viewpoint.

derivatives, and does not affect the intervals along straight lines. Experiments show that human beings always choose the straight path during daily life, since straight paths save time and energy. A simple example is that when a person wants to pick up an object, the hand approaches the object along a straight line. It is counter-intuitive that an individual would move his hand along a circle to approach the object. Therefore, outliers caused by projection are rarely *gross* errors. To handle these outliers we propose the use of dynamic time warping (DTW), which provides an efficient and reliable basis of suppressing outliers and finding correspondences between instants from different action trajectories. We will discuss the DTW algorithm in detail in Chapter 3.

2.2.7 Instant Detection Based on Multiple-motion Characteristics – Generalized Spatio-temporal Curvature

In previous sections, the motion was limited to a moving point, which means that at every moment in time the moving object is treated as a point. This assumption is valid when the size of object is relatively small compared with the distance between the object and the camera, and when the camera and the object are not rotating. However, this is not valid when the subject is performing some complicated actions. Therefore, we need a method that can analyze more complicated motion patterns.

Based on the discussion in Section 2.2.4, we can apply spatio-temporal curvature to the multi-dimensional motion characteristic curve $\{x_i, y_i, \theta_i, t_i\}$, $i = 0, 1, 2, \ldots$, such that changes of speed, direction and rotation will be captured by turning points in the spatio-temporal domain. The spatio-temporal curvature k is given as follows:

$$k = \frac{\sqrt{A^2 + B^2 + C^2 + D^2 + E^2 + F^2}}{\left((x')^2 + (y')^2 + (\theta')^2 + (t')^2\right)^{\frac{3}{2}}} \tag{2.12}$$

where

$$A = \begin{vmatrix} y' & t' \\ y'' & t'' \end{vmatrix}, B = \begin{vmatrix} t' & x' \\ t'' & x'' \end{vmatrix}, C = \begin{vmatrix} x' & y' \\ x'' & y'' \end{vmatrix},$$
$$D = \begin{vmatrix} \theta' & t' \\ \theta'' & t'' \end{vmatrix}, E = \begin{vmatrix} \theta' & x' \\ \theta'' & x'' \end{vmatrix}, F = \begin{vmatrix} \theta' & y' \\ \theta'' & y'' \end{vmatrix}.$$

The notation $|\cdot|$ denotes the determinant. Furthermore, we can generalize this formula to handle other motion characteristics that change with respect to time. In this way, we can detect the change in any motion characteristic using a single quantity. Moreover, the detection results are not affected by a change in viewpoint. This is a significant discovery, since we now have a uniform framework to detect important changes in different motion characteristics.

We use this approach to analyze human gait. Figure 2.16 shows the trajectories of left and right feet respectively in three walking sequences. The short line segments represent the foot orientations at the centroid. The detected instants correspond to three important changes during a walking cycle: "foot touching the ground", "leaving the ground", and then "moving forward". The intervals are "the foot remains on the ground", "the foot rises above the ground", and "the foot moves forward". We compared the results with the detection using only x, y, and t information, from which we can only get two instants consistently (Figure 2.17). Therefore, we conclude that the generalized spatio-temporal curvature can successfully detect the important changes of various motion characteristics during an action.

2.2.8 View-invariant Action Representation

Representation, which is an abstraction of the sensory data that reflects a real world situation, is an important and sometimes difficult aspect of an intelligent system. The representation of data should not only be view-invariant and compact but also be reliable for later processing.

For high-level data abstraction, we propose a new representation scheme based on the spatio-temporal curvature of a motion trajectory. Our representation of trajectory includes a sequence of *dynamic instants* and *intervals*, and assigns physical meanings to them. A *dynamic instant* is an instantaneous entity that occurs for only one frame, and represents an important change in the motion characteristics (speed, direction and acceleration). These changes are captured by the spatio-temporal curvature. We detect *dynamic instants* by identifying maxima in the spatio-temporal curvature of an action trajectory. Examples of *dynamic instants* include: touching, twisting and releasing during an action. Moreover, from Section 2.2.6 we conclude that the detection results are invariant with respect to viewing direction.

In the proposed representation, a *dynamic instant* is characterized by its frame number, the image location. Among these characteristics, the "frame number" represents the time at which the *dynamic instant* occurs and the "image location" provides the spatial position of the action agent in the frame when the *dynamic instant* occurs.

Similarly, an *interval* represents the time period between any two *dynamic instants*, during which the motion characteristics do not change drastically. Examples for *intervals*

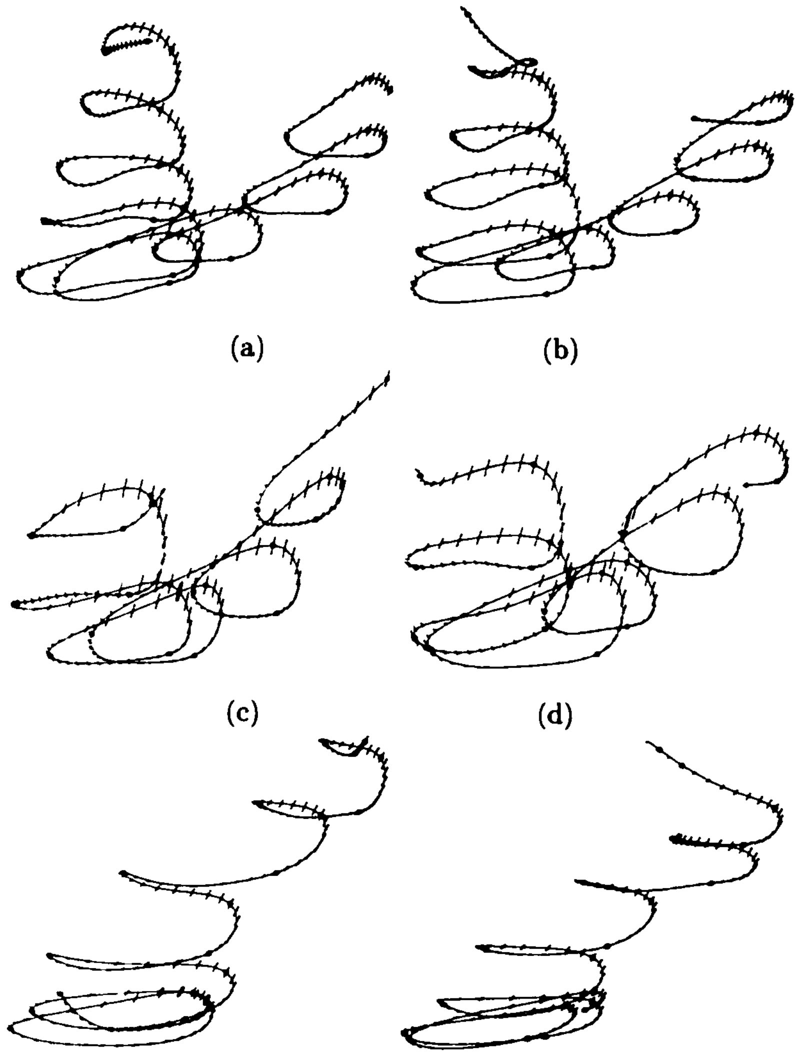

(a) (b) (c) (d) (e) (f)

Figure 2.16: The trajectories from three walking sequences, (a),(c) and (e) are the trajectories of the left foot; (b),(d) and (f) are the trajectories of the right foot (b,d,e). The small lines display the orientation value, and the '*' are the instants detected by spatio-temporal curvature.

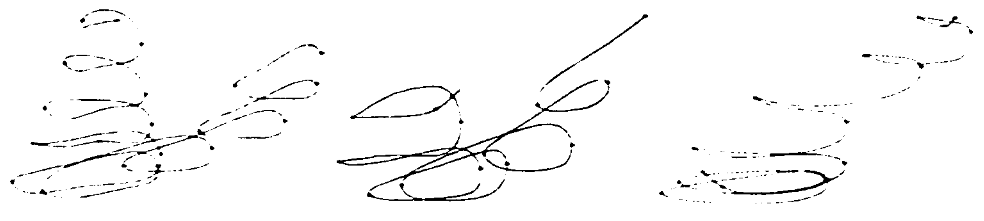
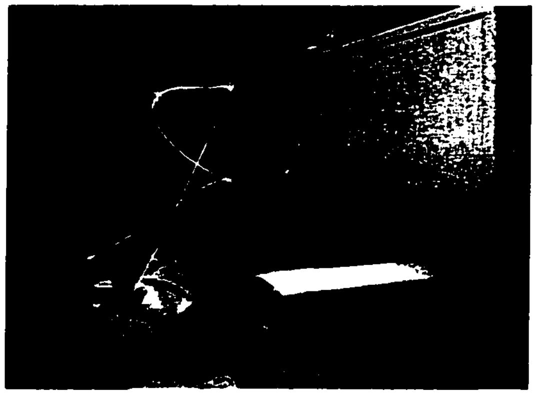


Figure 2.17: The left foot trajectories and the instant detection results (the red '*')using only x,y,t information.

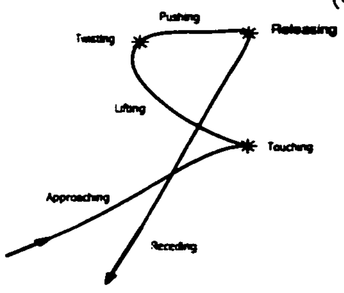
include approaching, lifting, pushing, and receding. The temporal information of interval is employed for recognition in Chapter 3.

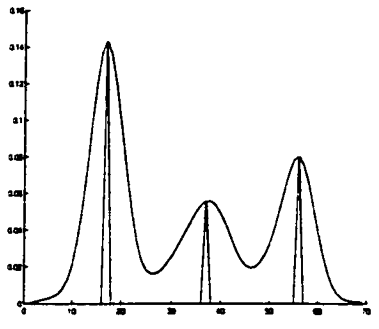
A remarkable feature of our representation is that it is able to explain an action in natural language in terms of meaningful atomic units, which not only can be mathematically modelled but can also be detected in real images. In Figure 2.18, we show the *dynamic instants* by $*$ on the motion trajectory of the opening overhead cabinet action. This action can be described as: the hand approaches the cabinet (approaching interval), the hand makes contact with the cabinet (touching instant), the hand lifts the cabinet door (lifting interval), the hand twists (twisting instant) the wrist, the hand pushes (pushing interval) the cabinet door in, the hand releases the cabinet door (releasing instant), and finally the hand recedes (receding interval) from the cabinet.

Figure 2.19 displays the trajectory of the "erasing whiteboard" action. This action can be described as: the hand approaches the eraser (approaching interval), the hand makes contact with the eraser (touching instant), the hand picks up the eraser (lifting interval), the hand turns (turning 1 instant), the hand wipes the board (wiping interval), the hand turns (turning 2 instant), the hand wipes (wiping interval), the hand turns (turning 3 instant), the hand wipes (wiping interval), the hand turns (turning 4 instant), the hand puts the eraser back (putting down interval), the hand releases the eraser (releasing instant), and finally the hand recedes (receding interval) from the board.



(a)





(b) (c)

Figure 2.18: (a) The "opening a cabinet" action. The hand trajectory shown in white is superimposed on the first image; (b) a representation of the action trajectory in terms of *instants* and *intervals*; (c) corresponding spatio-temporal curvature values and detected maximums (dynamic instants).

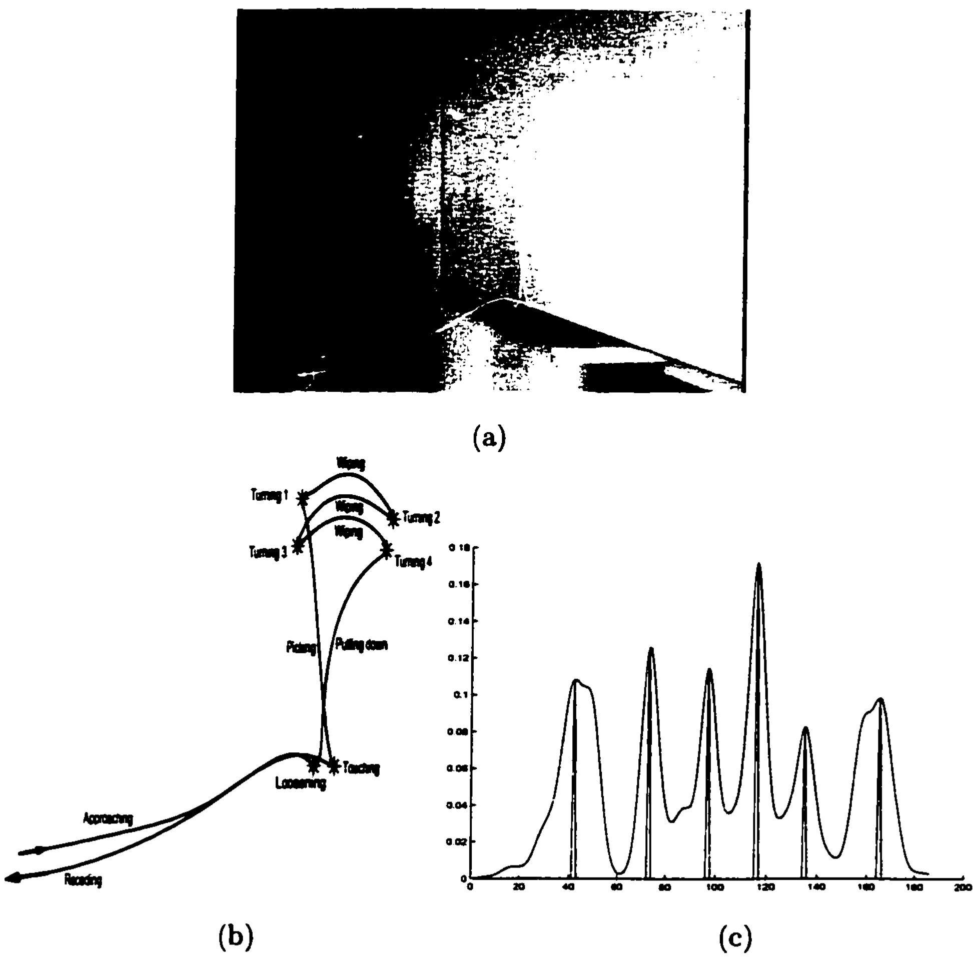


Figure 2.19: (a) The "erasing whiteboard" action. The hand trajectory shown in white is superimposed on the first image; (b) a representation of the action trajectory in terms of *instants* and *intervals*; (c) corresponding spatio-temporal curvature values and detected maximums (dynamic instants).

Figure 2.20 shows the trajectory of "picking up an object from the floor and then putting it down on the desk". The action can be described as: the hand approaches the object (approaching interval), makes contact with the object (touching instant), picks it up (lifting interval), breaks the contact (releasing instant), and then recedes (receding interval).

In previous approaches, such as TPS and PCA decomposed bases, the action was represented as a simple trajectory, and human perception is simply not considered. Moreover, such approaches treat each tracking data point equally, so the real physical events are not revealed. Therefore, their recognition modules require complicated processes to discover these events, such as using manually defined segmentation rules or extensive training examples. Otherwise, those methods can only detect some simple action instances. Furthermore, they cannot automatically generate new action models by using previous recorded exemplars. The systems do not succeed for general cases, because there are a huge number of human actions in our daily life and with a large variety among execution styles. Our representation is based on not only physical characteristics but also human perception, thus the recognition system obtains results, which are consistent with psychology research. What is more, our action representation has view-invariant characteristics, which allows for a higher-level interpretation of the information without any ambiguity. We will discuss issues involved in view- invariant recognition and learning based on the action representation in Chapter 3.

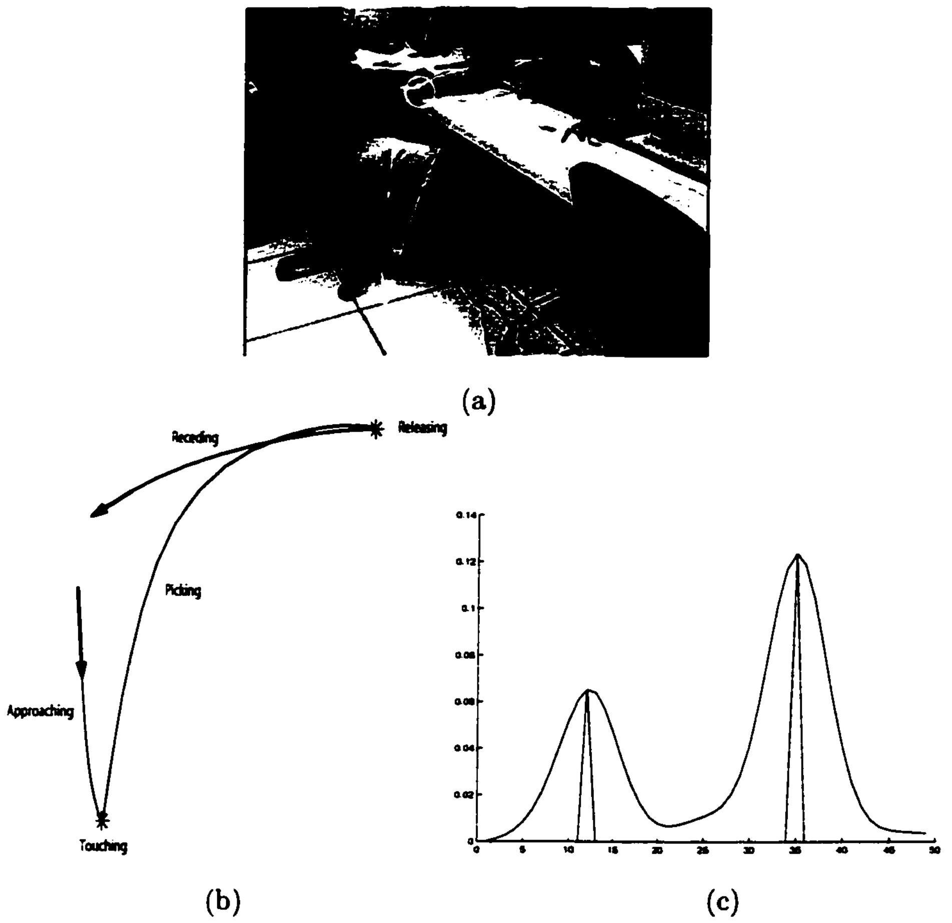


Figure 2.20: (a) The "picking up and putting down object" action. The hand trajectory shown in white is superimposed on the last image; (b) a representation of the action trajectory in terms of *instants* and *intervals*; (c) corresponding spatio-temporal curvature values and detected maximums (dynamic instants).

CHAPTER 3

ACTION RECOGNITION AND LEARNING

Once the representation has been defined, the next step is to use this representation to recognize and learn human actions. In this chapter, we will discuss the properties of instants, how to determine two action trajectories are from the same action, how to suppress the outliers of instant detection, and how to group the actions into categories. Because of the effectiveness of the representation, our recognition module achieves recognizing similar actions by different people and from different viewing directions, and categorizes actions into groups without any training process.

We assume that the camera is fixed. However, people can perform actions with any orientation. The system continuously analyzes a video stream captured by the camera. The system detects the action agent using skin detection, determines the trajectory of the action agent, and computes a view-invariant representation of each action.

For an action recognition system, there are two aspects that need to be considered: 1) the action trajectories are from different viewing directions, thus the same action may look very different; 2) a person executes an action with large temporal variation, i.e. the same person does the same action faster or slower each time. The recognition system must be able to handle both factors.

For each action, the recognition module determines sets of similar actions based on a match score. For example, different cases of the "opening overhead cabinet" action can be automatically determined to be similar. For each set, only one prototype representation is maintained, since all other instances convey the same information. For each prototype we associate a confidence, which is proportional to the cardinality of the set represented by this prototype. When more evidence is gathered, the confidence of some actions is increased, while the confidence of others remains the same. The prototypes with small confidence can ultimately be eliminated.

3.1 Related Work

The recognition approach is closely related with the representation of actions. Furthermore, a simple representation form requires a very smart recognition module to achieve a reasonable recognition rate, whereas a meaningful representation form can ease the complexity of the recognition module. Several recognition approaches will be described next.

The simplest representation is just the raw action trajectory or the motion characteristics. Due to the large variation of execution style, many training examples are needed for making a successful recognition module. Section 3.1.2 discusses this type of approach. Section 3.1.1 discusses the nearest distance measure, which is based on some typical representations. Section 3.1.3 discusses the recognition methods based on syntax, and Section 3.1.4 discusses the approaches based on the representations that fit parameters for temporal functions.

3.1.1 Nearest Neighbor Distance Based Recognition

For actions represented by motion-energy image (MEI) and motion history image (MHI) [18] (Section 2.2.1), the Hu-moments are calculated as descriptors of the templates. To classify an action, the Mahalanobis distance is calculated between the descriptor of the input and each known action. The input action will be grouped with the model that has the smallest distance from the input.

Yacoob and Black [76] proposed representing actions as the weighted summation of a set of bases. Their recognition module takes the coefficients of an action upon each basis and compares the distance with the known actions. The action is grouped with the action that gives the nearest distance.

Although, these two typical representations need relatively simple recognition modules, they require that the actions be performed in exactly the same viewing direction, because only 2-D information is used. Moreover, a manual preparation to set up a model for each action is necessary. Therefore, they are not successful in the general cases.

3.1.2 Recognition using HMMs

One standard approach for human action recognition is to extract a set of features from each frame of a sequence and use those features as input of classifiers. The features can be an image location of a particular point on the object, a centroid of an image region, moments of an image region, gray levels in a region, optical flow in a region (used as magnitude of optical flow, or concatenating u and v in a vector), the sum of all changed pixels in each column (XT trace), 3-D locations of a particular point on the object, joint angles; movement of the parts of the body with respect to time, muscle actuations, properties of optical flow in a region like curl and divergence, or coefficients used in the eigen decomposition of the above features. Although some classic classifier can be used (K-means, isodata), HMMs have received the most attention in action recognition research. An HMM consists of a set of states, a set of output symbols, state transition probabilities, output symbol probabilities, and initial state probabilities. The model works as follows: The features extracted from video sequences are used to train the HMMs; matching of an unknown sequence with a model is done through the calculation of the probability that an HMM could generate the particular unknown sequence; the HMM giving the highest probability is the one that most likely generated that sequence [78].

Discrimination between different tennis strokes was investigated by Yamato et al. [77] using HMM. They can be seen simply as symbol generating machines. An image sequence is processed in three steps. In the first, a mesh feature is extracted, and then associated to a symbol by a clustering technique. From this process, a sequence of output symbols is derived, such that there is one symbol for each frame of the sequence. In the second step, sequences are used to train the HMMs. There are as many HMMs as there are motions to be recognized. During this phase, the parameters describing an HMM are optimized such that it has a high probability of generating the sequence of output symbols derived from a particular motion. Finally, given a sequence of symbols from an unknown motion, we want to find the HMM that is most likely to generate the unknown sequence. The likelihood is computed using a probabilistic approach.

Siskind and Morris proposed a system that consists of two sub-systems [63]. The lower-level module performs object tracking. Position, orientation, shape, and size of each participant object are used. Ellipses are fitted to the characteristics of each object. The higher-level sub-system takes the 2-D pose stream produced by lower-level processing, and classifies it as an instance of a given action type. The system uses HMMs to perform the recognition task. The system was tested for six action types: pick up, put down, push, pull, drop and throw. 35 out of 36 sequences were correctly recognized.

Campbell et al. used 3-D measurements, which were obtained from a stereo camera, as features [10]. Gesture recognition was performed on a set of 18 T'ai Chi gestures (an ancient Chinese martial art), and the performance of ten different feature vectors derived from 3-D hand and head tracking data was compared. Their conclusion is worth noting that choosing the right set of features can be crucial to the recognition.

Essa et al. [44, 43] use HMMs to perform action recognition in an office, a kitchen, or a car. The areas occupied by the objects in the field of view are marked manually. As the hand moves to interact with the objects, it passes through certain areas that correspond to the HMM's states. Hand transitions from area to area generate a sequence of states, which characterize an action. In [44], the system captured 3 minutes of video of a user interacting with different objects in the office scene, and got a 92% correct recognition rate. More experiments were presented in [43], which consisted of three categories of a total of 450 labeled action examples for training and recognition. The authors reported an increase in recognition rate. This approach is an improvement on Siskind's, because the structure of HMMs is more meaningful.

Hoey and Little [28] proposed another framework. Instead of using trackers to get the motion trajectories, they use optical flow to capture the changes in the field of view. Zernike Polynomials are used to represent the flow. Then the HMMs use ZP coefficients to do recognition. This approach is very suitable for expression recognition and lipreading, because the ZP is defined on the unit disk so that it is very convenient to represent the spatial change of faces.

Many interesting actions are composed of multiple interacting processes. Even with the correct number of states and vast amounts of data, large HMMs generally train poorly, because the data is partitioned among states early (and incorrectly) during training, although HMMs can model any system in principle. In order to model these interactions, Oliver et al. proposed a more complex architecture, Coupled Hidden Markov Models (CHHM). In this architecture, state chains are coupled via matrices of conditional probabilities modeling causal (temporal) influences between their hidden state variables. For each chain, the state at time t depends on the state at time $t-1$ in both chains. The influence of one chain on the other is through a causal link. Moreover, in order to improve the efficiency of the training process, Oliver et al. [50] developed synthetic behavioral agents for the ability to generate synthetic data, which allows the system to determine which Markov model architecture will be best for recognizing a new behavior (since it is difficult to collect real examples of rare behaviors). However, the examples are very simple, such as one person following another person, a person approaching another person and talking to each other then separating, a person approaching another person then talking to each other and leaving together.

In the HMM based research, more emphasis has been put on discovering appropriate features. Therefore, not much work has been done on HMMs; they have been treated as black boxes. There are several important issues related to HMMs. First, since HMMs rely on probabilities, they require extensive training, so one needs to have a large number of training sequences for each action to be recognized. Some of the results reported are tricky. For example, researchers use 80% of the data to do training and 20% of the data for testing. The recognition module may be overtrained, and not doing real classifying work. Second, for each action to be recognized, a separate HMM needs to be built. Therefore, this approach can only recognize some predefined set of actions. Third, since the HMM is treated as a black box, it does not explain or describe an action. It just outputs the probability an unknown action is recognized as a modelled action. Regarding features, the issue of representation of features has mainly been ignored. Furthermore, in most approaches, only view-based features have been used, so the proposed systems do not have the ability to recognize the same action from different viewing directions.

3.1.3 Parsing

Instead of using HMMs, Ayers and Shah [3] use state models to perform recognition. Their recognition module uses a predefined environment layout, and a state machine based on prior knowledge. Tracking, skin detection, and scene change detection modules are used to provide the necessary information to the action recognition module. The actions are recognized based on the state machine results. The states represent the transitions of actions so that they exhibit real physics and psychology meaning.

Kuniyoshi et al. [39] proposed an action recognition approach by using a symbolic parser. From the hand tracker, the system gets the motion status, such as moving up, moving down, or moving fast. Furthermore, the relationship between the hand and the object are described as approaching, leaving, touching, or separate. Based on the motion status and relationship, the atomic components of actions are represented as symbols. The symbols are stored in a queue, and fed into a parser. The parser classifies an unknown action as one of the predefined actions based on the syntax rules of the parser. The recognition module requires a lot of manual preparation, such as predefined rules to recognize events and status.

Bobick and Ivanov [8] proposed a very interesting approach for action recognition, which uses Context Free Stochastic Grammar. A parser is used to analyze the HMM output (the probability of input compared to the model), and HMMs are trained to recognize primitive components of actions. The action sequence is fed into the HMMs in parallel. The parser takes the HMM output values, and recognition is based on a predefined grammar. This approach has some advantages: 1) there are only a limited number of action components, but those components can represent a huge set of actions; 2) HMMs can take care of uncertainty factors in human action; 3) the parser provides longer range temporal constraints, disambiguates uncertain low-level processing, and allows the inclusion of a priori knowledge about the structures of temporal events in a given domain. However, the problem with this approach is obvious, this approach needs not only a huge set of training data but also manually defined syntax rules.

These systems use predefined models, which limit the number of actions that can be recognized. The grammars rules and state machine are manually designed, so it will take a lot of set up time to recognize a new action. Moreover, the systems ignored the view invariance problem. so they can only be used in very constrained domains. Furthermore, these systems try to recognize atomic action components from action sequences, so each component needs to be distinguished from the other ones at a very early stage of recognition. Therefore, once there is a mistake at the low-level stage, the final results will be greatly affected. For this reason, these systems always have a lot of heuristic constraints to improve their robustness. However, these constrains also restrict the range of actions that can be recognized.

3.1.4 Parameter Fitting

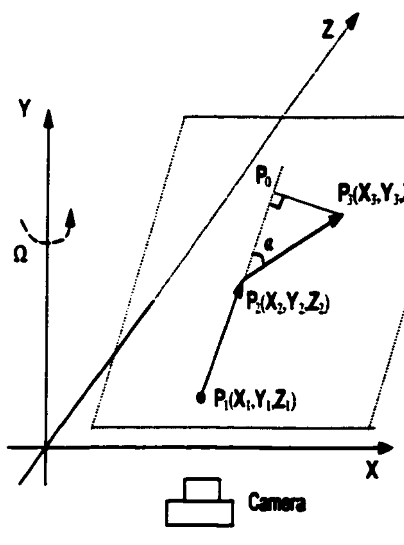
TPS [25] and sinusoidal model [19] display the action category by the representations, thus the recognition modules are very simple by just analyzing the coefficients of functions representing actions. However, the systems ignored unmodelled actions. Therefore, the types of actions that can be recognized are limited. In addition, only 2-D motion characteristics are considered in these approaches, and the representations may vary widely for the same action from different viewing directions. These approaches have restricted uses.

3.1.5 Conclusion

All the recognition approaches reviewed cannot achieve recognizing similar actions by different people and from different viewing directions, and categorize actions into groups without any training process. They require either manually setting up recognition syntaxes, or an extensive training process, and most of them ignored the fact that actions are in 3-D space by only using 2-D motion characteristics. In this chapter, we propose a spatio-temporal matching algorithm to overcome these shortcomings.

3.2 The Properties of Instants

An action is represented as a sequence of instants and intervals. Instants represent the physical interaction between the action agent and the environment, while the intervals represent how the action agent moves from one event to the next. Therefore, the instants display clues for action recognition.



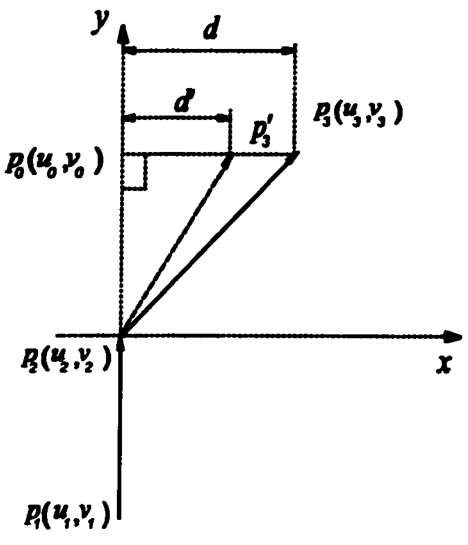


Figure 3.1: (a) Three points in 3-D, (b) the 2-D projections on the image plane.

3.2.1 The Sign of an Instant

Assume that the location of an action agent in 3-D space at times t_1, t_2, t_3 is given by P_1, P_2, P_3. In this case, we have two vectors $\overrightarrow{P_1P_2}$ and$\overrightarrow{P_2P_3}$ (see Figure 3.1a). The projection of these three points in image plane is shown in Figure 3.1b. It is clear that there is a dynamic instant at t_2, due to the significant change in the direction. Assume that the angle between the two vectors is α. The sign of this angle can be determined by computing the sign of the cross product of the projections of the two vectors in the image plane. We will use the sign of this angle as the sign of the instant. In this research, we define the sign of turning right positive, and the sign of turning left negative. We claim that the sign of an instant is view-invariant when the camera viewpoint remains in the upper hemisphere of the viewing sphere.

The camera translation will not affect the angle α. Therefore, we will only consider the situation when the camera rotates. Let us assume, for simplicity, that the camera axis passes through P$_2$ and is perpendicular to the X-Y plane. The distance from the camera to P_2 is D, and $\overrightarrow{P_1P_2}$ is always vertical. It is obvious that camera rotation around the Z-axis does not change α. Therefore, the situations that need to be considered are camera rotations around the X-axis (*tilt*) and the Y-axis (*pan*). While the camera pans (Figure 3.1), the only part that changes is the projection of $P_3(X_3, Y_3, Z_3)$, which becomes $p'_3(u'_3, v'_3)$ in Figure 3.1. Note that P_0 is the projection of P_3 on the line P_1P_2 and its image coordinates are (u_0, v_0). When the camera pans by angle θ, the X-coordinate of any point is changed to X' as follows,

$$X' = X\cos\theta - Z\sin\theta \tag{3.1}$$

The image coordinates under the affine camera model are given by:

$$u' = f\frac{X'}{D} \tag{3.2}$$

Where f is the focal length, and the distance from the camera to P_2 is D. The distance, d_1, between projection of points P_3 and P_0 in the image plane is given by:

$$\begin{aligned} d_1 &= u'_3 - u'_0 \\ &= f\frac{[(X_3\cos\theta - Z\sin\theta) - (X_0\cos\theta - Z\sin\theta)]}{D} \\ &= f\frac{(X_3 - X_0)\cos\theta}{D} \end{aligned} \tag{3.3}$$

In Equation 3.3, if $\theta \in (-90^o, +90^o)$, then $cos(\theta)$ is positive, and the rest of elements on the right hand side of equation are constant, therefore, d_1 retains its sign, so α retains its sign. This means that the sign of α is view-invariant when the camera is panning within the semicircle.

For the situation when the camera tilts around the X-axis and the rotation angle is ϕ, the similar argument still holds, so that the sign of d_2 is view-invariant when the camera is tilting within the semicircle ($\phi \in (-90^o, +90^o)$). Therefore, when the camera tilts within the semicircle the sign of ϕ remains the same. Moreover, the pan and tilt can be combined together to make the camera rotate around an arbitrary axis in the X-Y plane.

The above discussion is for the situations when all the instants are located in one plane, which is restricted. However, we can extend the proof for more general situations as follows:

Assume that there are four instants (P_1, P_2, P_3, P_4). Among these instants, P_1, P_2, P_3 are in one plane with an angle α, and P_2,P_3,P_4 are in another plane with an angle β. Then the signs of α and β are invariant when the camera rotates within within the space of a sphere defined by the two planes intersecting the sphere. For the situations when more non-planar instants are involved, the invariance property of the method is limited to the region where the camera can move without crossing these planes. However, we believe this representation is adequate, since there are not many cases in human actions which generate a lot of non-planar instants.

The sign characteristic of an instant is very useful, because the sequence of signs of instants help us to distinguish between different actions. When the camera movement is within the view-invariant range we discussed above, for any two action trajectories, having the same permutation of instant signs is a necessary condition for the actions to be the same.

For example, the opening cabinet action (Figure 3.2a) has five instants, the signs for second, third and fourth instants are (-,+,+). On the other hand, the closing cabinet action (Figure 3.2b) also has five instants, but the signs of the middle three instants are (-,-,+). In general, for a trajectory with n instants, the number of permutations of signs is $2^{(n-2)}$; here we are not considering the signs of the first and the last instants.

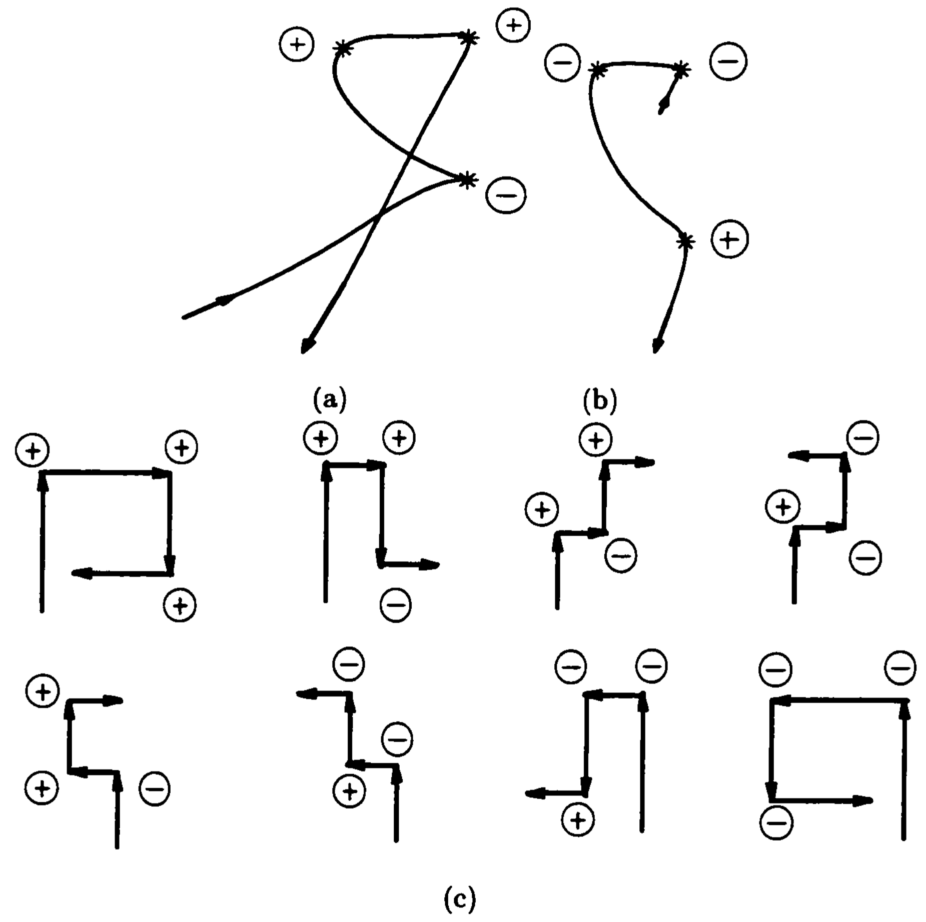


Figure 3.2: (a) Trajectory of the "opening cabinet" action and the signs of the instants. (b) Trajectory of the "closing cabinet" action. (c) Possible permutations of 5-instant actions.

3.2.2 Pick up and Put down Actions Recognition based on instant

The smallest action consists of three instants (start, picking up/putting down object, and end), and those actions are either "pick up an object", or "put down an object". Previously, there were mainly two types of methods to distinguish between picking up and putting down actions. One is based on tracking and the other one is based on context information. The tracking method needs to identify both the hand and the object, because during the approaching stage of picking up an object there is no object in the hand, and when the hand recedes after the picking up event, there is an object in the hand [39]. The system needs to handle occlusion, which will happen when the hand touches the object. Therefore, this approach is not easy to implement. The predefined context approach does recognition based on the change of a specific image area, which corresponds to a specific object. If there is enough change, then the system concludes that the object is moved by a predefined action [3, 44]. This is a very expensive approach, because it requires manually marking the position of every object in the field of view. Moreover, it is view dependent. Once the camera parameters are changed, the system needs another round of manual preparation. Furthermore, it is hard to distinguish between picking up and putting down, because both of them change the background significantly.

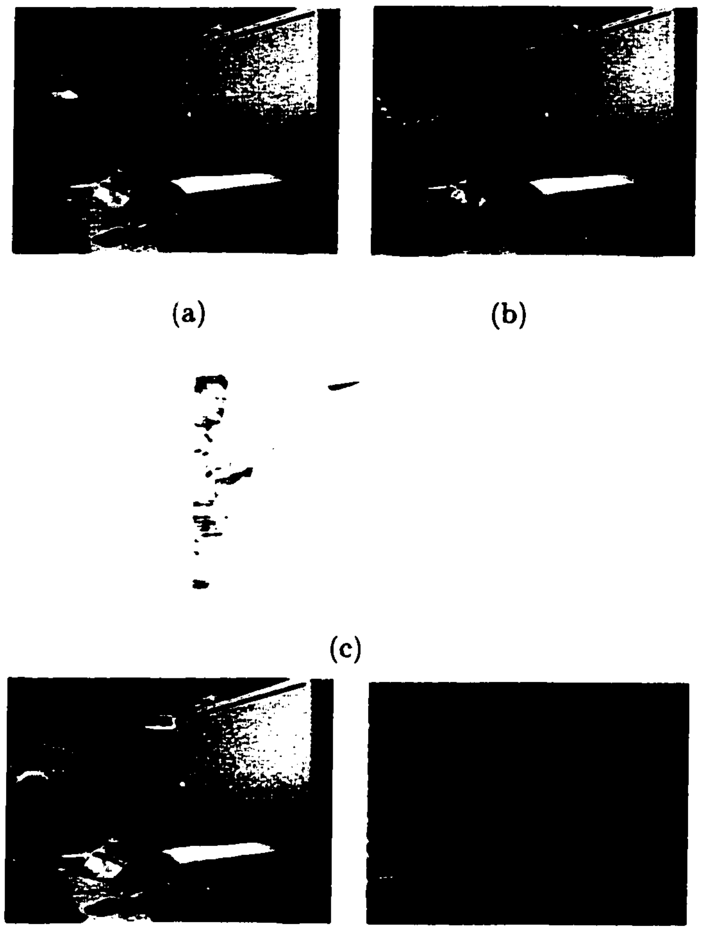
We interpret "pick up" and "put down" actions as follows. We subtract the frame corresponding to the first instant from the frame corresponding to the last instant, and compute the difference for each pixel. Then we apply a Gaussian mask centered at the location of the second instant of the action trajectory. This step emphasizes the pixels in the neighborhood of the location where "pick up" or "put down" happens. Therefore, other interferences, caused by lighting changes or human body movement, are removed. Then a threshold is applied to the weighted difference picture to identify relevant pixels in the middle frame, and a bounding box is calculated for the region. Figures 3.3(c) and 3.4(c) show the difference images, which contain changes caused by human body occlusion, shadows, and the actions. However, with our trajectory-based Gaussian mask, we can accurately find the change area (object), which is caused by the action only. Figures 3.3(d) and 3.4(d) show the detection of objects, which are marked out nicely.

Next, we need to determine if an action is a "pick up" or "put down" action. An edge detector is applied to pixels in the region inside the bounding box of the first frame and the last frame, and the difference of the two edge images is computed. Note that edge images are binary images, thus the difference picture will consists of pixels with values, 0, -1 or 1. We sum the difference of edge images. If the summation is positive then the action is "pick up", if it is negative then it is "put down", and if it is zero then nothing was picked up or put down.

Here, we assume that if the object is in the field of view, there should be more edges detected than the situation in which no object is in the field of view. This is true for most office environments, where the desks are even and uniform colored, and the objects are polyhedrons. A similar idea was applied in classifying whether a region belongs to foreground object or background by Javed et al. in [33].

The representative results for "put down" and "pick up" actions are shown in Figures 3.3 and 3.4. We need only two frames and the action trajectory, require no training, and have achieved a high recognition rate for actions from any viewing direction.

This method is applied to only the simplest actions, in which the hand only interacts with the object only once and the beginning frame and ending frame do not have much area occupied by the person's body. The image areas of instants from other more complicated actions are contaminated, because the subject may stay a long time during the action and occlude the image area where instants are detected. Therefore, the property of picking up or putting down is not reliable.



(a) (b)

(c)

(d) (e)

Figure 3.3: A putting down action. The person put down a bunch of paper on top of the cabinet. (a) The beginning frame, (b) the ending frame, (c) the difference image, (d) the bounding box of the changing area superimposed on the beginning frame, (e) the difference of the edge images.

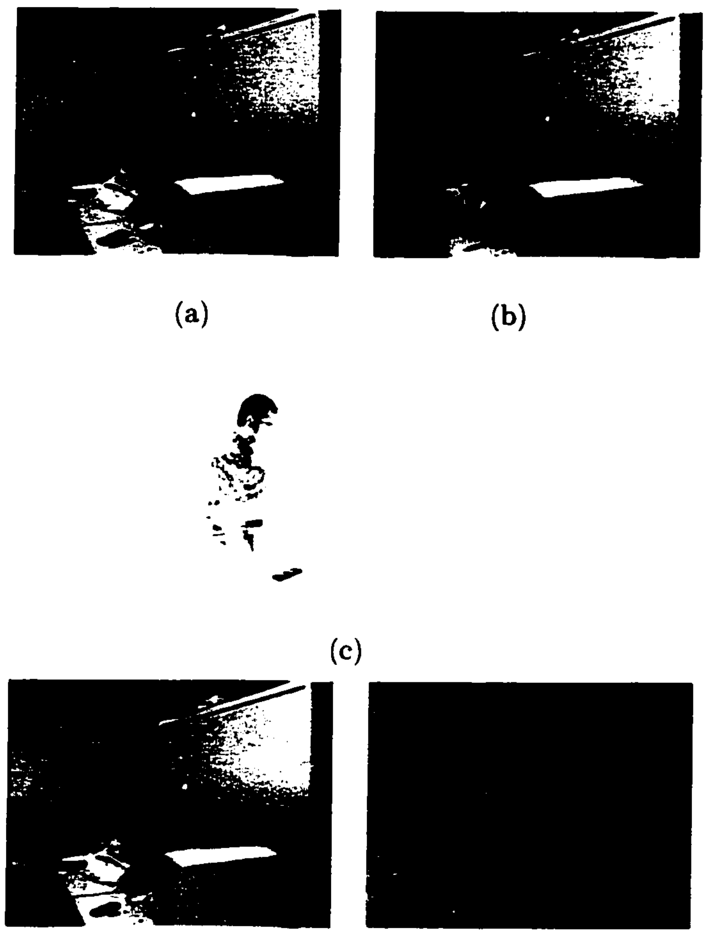

(a) (b) (c) (d) (e)

Figure 3.4: A picking up action. The person picked up a remote control from the desk. (a) The beginning frame, (b) the ending frame, (c) the difference image, (d) the bounding box of the changing area superimposed on the beginning frame, (d) the difference of the edge images.

3.2.3 Limitations of Instants

From the previous discussion, we can conclude that the detection of instants and the signs of instants in an action are view-invariant. However, these two characteristics of instants are not sufficient to uniquely define any action, since two different actions may have the same number of instants with the same signs. Moreover, an instant is an instantaneous entity that occurs for only a single frame, while the information of intervals, which represents how the action agent moves from one instant to the next, has not been used until now. Therefore, we propose to use a view-invariant method to measure the similarity between two action trajectories in Section 3.4. The trajectories of the same action should give us a high matching score as compared to the trajectories of different actions. Moreover, the matching score should not be affected by the camera viewpoint or the length of execution time of the actions.

3.3 View-invariant Similarity Measurement

Given two viewpoint invariant representations of some actions, how can we determine if these are the same actions? We use a view-invariant matching function that equates a set of images if and only if they represent views of an object in the same configuration, as proposed by Seitz and Dyer in [61]. In the recognition system, we assume the camera model is an affine model.

The affine camera model is a special case of the projective camera and was proposed by Mundy and Zisserman [45]. The projection matrix (homogeneous coordinates) can be represented as:

$$P_{aff} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ 0 & 0 & 0 & P_{34} \end{bmatrix} \tag{3.4}$$

An image coordinate $\mathbf{u} = (u, v)^T$ can be represented as a projection of a 3-D point $\mathbf{X} = (X, Y, Z)^T$:

$$u = NX + t \tag{3.5}$$

where $\mathbf{N}$ is a 2×3 matrix (with elements $N_{ij} = P_{ij}/P_{34}$) and $\mathbf{t} = (P_{14}/P_{34}, P_{24}/P_{34})^T$ is a vector.

A property of this affine camera is that it retains its form when the scene undergoes a 3-D affine transformation. Consider a 3-D point $\mathbf{X}$ that moves to a new position $\mathbf{X}'$ as $\mathbf{X}' = A\mathbf{X} + T$, where A is a 3×3 matrix and T is a 3-vector. The new 3-D position $\mathbf{X}'$then projects to $\mathbf{u}' = (u', v')^T$, where

$$\begin{aligned} u' &= NX' + t = N(AX + T) + t \\ &= NAX' + (NT + t) = N'X + t' \end{aligned} \tag{3.6}$$

A second property of the affine camera model is that relative coordinates cancel out translation effects, such that $\Delta X = X - X_0$ and $\Delta X' = X' - X_0' = A\Delta X$. Furthermore, in the image, the points are:

$\Delta u = u - u_0 = N\Delta X$, and

$$\Delta u' = u' - u_0' = N'\Delta X = NA\Delta X$$

Therefore, the image coordinates are independent of $\mathbf{T}$, t and t'. Accordingly, the projection matrix can be re-written as:

$$\Pi_{aff} = N = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \end{bmatrix} \tag{3.7}$$

We configure the world coordinates such that the origin of the world coordinates is placed at the centroid of action instants (as described by Tomasi and Kanade in [68]). As a result, the origin of the image coordinates is at the centroid of the image trajectory instants. Furthermore, we represent an action by a sequence of n instants, where each action is represented by the (x, y) image coordinates of each instant: $\mathbf{I} = ((u_1, v_1)^T, (u_2, v_2)^T, \ldots, (u_n, v_n)^T)$. Assume a particular action is captured in k views, represented by: $I_{v_1}, I_{v_2}, \ldots, I_{v_k}$. Our aim is to automatically determine whether these views represent the same action. Let us form a matrix M as follows:

$$M = \begin{bmatrix} I_{v_1} \\ I_{v_2} \\ \vdots \\ I_{v_k} \end{bmatrix} \quad and \quad I_v = \begin{bmatrix} u_1^v & u_2^v & \cdots & u_n^v \\ \nu_1^v & \nu_2^v & \cdots & \nu_n^v \end{bmatrix} \tag{3.8}$$

If the views represent the same action, then we can express M as:

$$M = \begin{bmatrix} \Pi_{v_1} \\ \vdots \\ \Pi_{v_k} \end{bmatrix} S \tag{3.9}$$

where $\mathbf{S}$ (shape) represents the 3-D coordinates of points corresponding to the instants, and $\mathbf{\Pi}_v$ is the projection matrix of each viewpoint. Matrix M is the product of two matrices, each having a rank at the most 3. Therefore, the rank of M is at most 3. This is due to the rank theorem by Tomasi and Kanade [68]. As a consequence of this result, if the views represent the same action, and there are no numerical errors, then all the singular values of the matrix M except the first three will be zero. However, these singular values may not be exactly zero. Therefore, Seitz and Dyer [61] use the sum of the squares of singular values of M, except the first three singular values, to match the different views. This distance is given as follows:

$$dist = \sqrt{\frac{1}{2kn}\sum_{4}^{n}\sigma_i^2} \tag{3.10}$$

where $\sigma_{1..n}$ are the singular values of **M**, k denotes the number of views, and n denotes the number of singular values. This distance gives the average amount necessary to additively perturb the coordinates of each instant in order to produce projections of a single action.

To match two actions $\mathbf{I}_i$ and $\mathbf{I}_j$, we form matrix **M** as follows:

$$M = \begin{bmatrix} \mu_1^i & \mu_2^i & \dots & \mu_n^i \\ \nu_1^i & \nu_2^i & \dots & \nu_n^i \\ \mu_1^j & \mu_2^j & \dots & \mu_n^j \\ \nu_1^j & \nu_2^j & \dots & \nu_n^j \end{bmatrix}$$

We then determine the singular values of M, and compute the distance (Equation 3.10) as $dist_{i,j} = |\sigma_4|$. The distance gives the matching error of two action trajectories.

However, when we match two actions, there are two possible shape matrices S_i, and S_j. In this case the rank theorem may not be valid. We need to prove that if the rank of M is 3 then S_i and S_j represent the same actions. Shapiro et al. [62] studied the affine camera epipolar geometry as follows:

Remember that the affine camera model is represented in general as:

$$P_{aff} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ 0 & 0 & 0 & P_{34} \end{bmatrix} \tag{3.11}$$

An image coordinate $\mathbf{u}=(u,v)^T$ can be represented as a projection of the 3-D point $\mathbf{X}=(X,Y,Z)^T$

$$\mathbf{u} = \mathbf{N}\mathbf{X} + \mathbf{t} \tag{3.12}$$

and

$$\mathbf{u}' = \mathbf{N}'X + \mathbf{t}' \tag{3.13}$$

From Equation 3.12 **N** can be partitioned as $(\mathbf{B}|\mathbf{b})$, where **B** is a 2×2 matrix and **b** a 2×1 vector, and $(X, Y, Z)^T$ are the world coordinates.

$$\mathbf{u} = \mathbf{B}\begin{bmatrix} X \\ Y \end{bmatrix} + Z\mathbf{b} + \mathbf{t} \tag{3.14}$$

and similarly for $\mathbf{N}'$

$$\mathbf{u}' = B'\begin{bmatrix} X \\ Y \end{bmatrix} + Z\mathbf{b}' + \mathbf{t}' \tag{3.15}$$

From Equations 3.14 and 3.15, we can eliminate the world coordinates $(X,Y)^T$, and get:

$$\mathbf{u}' = \Gamma\mathbf{u} + Z\mathbf{d} + \varepsilon \tag{3.16}$$

with $\Gamma = B'B^{-1}$, $\mathbf{d} = \mathbf{b}' - \Gamma\mathbf{b}$, and $\varepsilon = \mathbf{t}' - \Gamma\mathbf{t}$. These quantities depend only on the camera motion – not on the scene structure. Notice Γ is a 2×2 matrix, and **d** and ε are vectors. Multiplying both sides of Equation (3.16) by $\mathbf{d}^{\perp}$, which is perpendicular to **d**, and observing that $\mathbf{d} \cdot \mathbf{d}^{\perp} = 0$, we get:

$$(u' - \Gamma u - \varepsilon)^T d^{\perp} = 0 \tag{3.17}$$

Then, Equation (3.17) can be represented as $au' + bv' + cu + dv + e = 0$. Moreover, the difference vector form is:

$$a(u' - u_0) + b(v' - v_0) + c(u - u_0) + d(v - v_0) = 0 \tag{3.18}$$

We rewrite Equation (3.18) in matrix form, such that

$$\begin{bmatrix} u' - u'_0 & v' - v'_0 & u - u_0 & v - v_0 \end{bmatrix} n = 0,$$

where $n = (a, b, c, d)^T$ and represents the motion parameters. Since all k points on the object share one set of motion parameters, we can represent the relationship before and after the movement as:

$$\begin{bmatrix} u'_1 - u'_0 & v'_1 - v'_0 & u_1 - u_0 & v_1 - v_0 \\ u'_2 - u'_0 & v'_2 - v'_0 & u_2 - u_0 & v_2 - v_0 \\ \vdots & \vdots & \vdots & \vdots \\ u'_k - u'_0 & v'_k - v' & u_k - u_0 & v_k - v_0 \end{bmatrix} n = \mathbf{M}^T n = 0 \tag{3.19}$$

Conclusion: Since $\mathbf{M}^T$ is a $k \times 4$ matrix, in order to obtain a non-trivial solution for **n**, the rank of matrix **M** must be at most 3.

The only exception is that when a set of instants are located on a line. In this situation, the two rows of matrix **M** are linear dependant, then the rank of **M** becomes 3 no matter the values in the other two rows. Fortunately, we can detect this extreme situation easily.

Using this conclusion in the context of human action recognition, we get the following theorem:

Theorem: *Under the affine camera model, if* rank$(M) \leq$ *3 and neither of the actions has all instants aligned in a straight line, then the two actions S_i and S_j match, where M is configured as:*

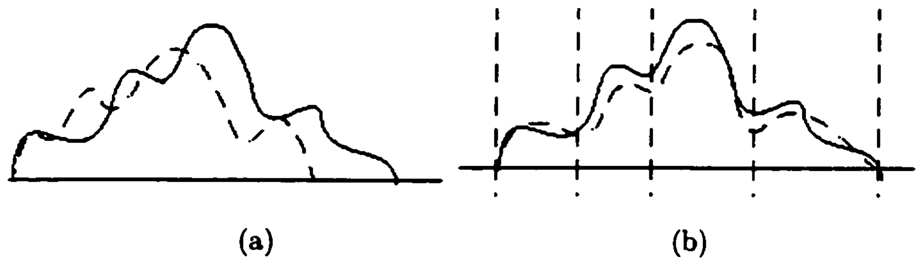
$$\mathbf{M} = \begin{bmatrix} \mu^i_1 - \mu^i_0 & \mu^i_2 - \mu^i_0 & \dots & \mu^i_n - \mu^i_0 \\ \nu^i_1 - \nu^i_0 & \nu^i_2 - \nu^i_0 & \dots & \nu^i_n - \nu^i_0 \\ \mu^j_1 - \mu^j_0 & \mu^j_2 - \mu^j_0 & \dots & \mu^j_n - \mu^j_0 \\ \nu^j_1 - \nu^j_1 & \nu^j_2 - \nu^j_1 & \dots & \nu^j_n - \nu^j_1 \end{bmatrix}, \tag{3.20}$$

where (μ^i_1, ν^i_1) and (μ^j_1, ν^j_1) are the image coordinates of the points of the trajectories I and J respectively, and (μ^i_0, ν^i_0) and (μ^j_0, ν^j_0) are the means of instants I and J respectively.

This is an important discovery. Once the instants are detected and assuming the intervals are just straight paths, the actions can be recognized by using the spatial information of the instants. The recognition system just needs to arrange the image coordinates of instants from two action trajectories in the form of **M**, and the 4^{th} singular value of matrix **M** gives the similarity measurement between the two action trajectories. Moreover, this measurement is view-invariant without explicit 3-D information, which is proved by the affine epipolar geometry study.

Notice that the theorem requires that each column of observation matrix **M** represent the image coordinates of the same specific instant in all the projections. In other words, $(\mu_k^i, \nu_k^i)$and (μ_k^j, ν_k^j) $(k = 1 \ldots n)$ must correspond to the same 3-D instant. This is achieved implicitly by the representation system. Since each observed action video is represented as a sequence of dynamic instants and intervals, if the trajectories i and j are from the same action in 3-D space, then the 1^{st} instant of trajectory i should correspond to the 1^{st} instant of trajectory j, similar to the 2^{nd}, 3^{rd},..., and n^{th} instants in both trajectories. There is no need for time alignment for the action trajectories, as long as the important events during the actions are captured consistently in all the action trajectories. The similarity measurement results are shown in Chapter 5.

However, once there is a false detection of an instant in the representation system, the 4^{th} singular value measurement will fail. Moreover, the motion information of intervals is not used in this measurement by treating the intervals as straight paths. Therefore, we propose a new method that can use all the information in the action trajectory to perform measuring the similarity.



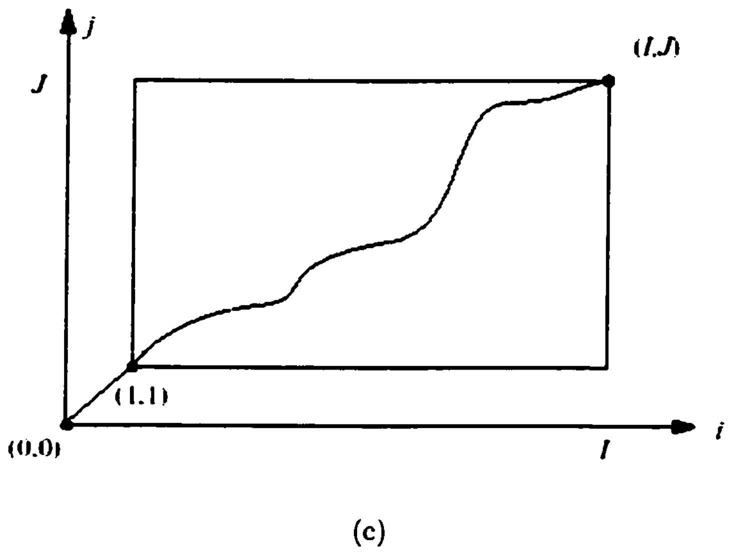


Figure 3.5: a) Two temporal signals, b) after time warping, c) the distance metric C and the warping path.

3.4 View-invariant Dynamic Time Warping Matching

3.4.1 Dynamic Time Warping

There are several methods to measure the similarity between two temporal signals, such as HMM, neural network and dynamic time warping (DTW). DTW is chosen in our system since research shows that it consistently outperforms HMM when the amount of training data is low [81]. Furthermore, in our learning system, based on the similarity measurement between each action trajectory, a nearest neighbor clustering can be easily applied to achieve unsupervised learning, and new action categories are generated when needed. HMM and neural network approaches do not have this capability.

Dynamic Time Warping (DTW) is a widely used technique for matching two temporal signals. It uses an optimum time expansion/compression function to do non-linear time alignment (Figure 3.5). The applications include speech recognition, gesture recognition, signature recognition [81, 17, 46]. For two signals **I** and **J**, a distance metric **C** is computed to represent the alignment between the two actions, with C_{ij} representing the cost of aligning the actions up to the time instants t_i and t_j respectively. The cost of alignment is computed incrementally using the formula:

$$C_{i,j} = d_{i,j} + \min\left\{C_{(i-1,j)}, C_{(i-1,j-1)}, C_{(i,j-1)}\right\} \tag{3.21}$$

Here d_{ij} captures the cost of making time instants t_i and t_j correspond. The best alignment is then found by keeping track of the element that contributed to the minimization of alignment error at each step and following a path backwards through them from element C_{ij}.

So far, the above framework can handle only motion information. We now inject shape information into the analysis through the d_{ij} metric.

3.4.2 View-invariant Dynamic Time Warping

Based on the view-invariant similarity measurement in Section 3.3, we propose a view-invariant DTW as follows:

Step 1: For each trajectory, the system picks 4 instants from the instant detection results, such that the permutations of signs are the same (Section 3.2.1).

Step 2: Execute the classic DTW algorithm, but replace the distance measurement between the t_i and the t_j points of two trajectories with the following:

$d_{(i,j)} = |\sigma_4|$, where σ_4 is the fourth singular value of matrix **M**, and **M** is defined as:

$$M = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & u_i \\ y_1 & y_2 & y_3 & y_4 & v_i \\ x'_1 & x'_2 & x'_3 & x'_4 & u'_j \\ y'_1 & y'_2 & y'_3 & y'_4 & v'_j \end{bmatrix} \tag{3.22}$$

the $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$ and $\{(x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3), (x'_4, y'_4)\}$ are the (x,y) image coordinates of 4 instants in two trajectories separately. (u_i,v_i) is the image coordinate of the i^{th} point in one trajectory, (u'_i,v'_i) is the image coordinate of the j^{th} point in the other trajectory*.

* note: the DTW can establish correspondence "on the fly", which means that it provides the best warping path to the element $C(i,j)$. To achieve more robust measurement for the $C(i,j)$, we put the previously found corresponding points up to $i-1$ and $j-1$ in the observation matrix **M**, and update the original observation matrix **M** (Equation 3.21) as: $d'_{(i,j)} = |\sigma_4|$, where $\sigma_4$is the fourth singular value of matrix **M′**, and **M′** is defined as:

$$\mathbf{M}' = \left[\begin{array}{cccc|cccccc|c} x_1 & x_2 & x_3 & x_4 & u_1 & u_2 & u_3 & u_4 & \cdots & u_{i-1} & u_i \\ y_1 & y_2 & y_3 & y_4 & v_1 & v_2 & v_3 & v_4 & \cdots & v_{i-1} & v_i \\ x'_1 & x'_2 & x'_3 & x'_4 & u'_1 & u'_2 & u'_3 & u'_4 & \cdots & u'_{j-1} & u'_j \\ y'_1 & y'_2 & y'_3 & y'_4 & v'_1 & v'_2 & v'_3 & v'_4 & \cdots & v'_{j-1} & v'_j \end{array}\right], \tag{3.23}$$

where $i = 1 \dots n$, and $j = 1 \dots m$ are the trajectory index numbers. Moreover, trajectory points $1, \dots, i-1$ and the trajectory points $1, \dots, j-1$ are the best match, provided by DTW, from $(1,1)$ to position $(i-1, j-1)$ in cost matrix $\mathbf{C}$.

Step 3: Record this matching distance and the correspondence result. The correspondence results are used for validating the 4 instants matching, since they must be located on the optimum path, or else the result is abandoned.

Step 4: If there are other instants available, go back to step 1 and run steps 1,2 and 3 DTW again until all the combinations of instants are checked.

Find the minimal global distance from step 3, and take the correspondence as the matching of two trajectories. If the distance is lower than some threshold, then the system declares the two action trajectories are from the same 3-D action.

This algorithm performs DTW without being affected by viewpoint variance since the difference measurement itself is not dependent on the viewpoint. Moreover, the instant outliers are suppressed if there are enough correct detections.

The instants outliers are suppressed in the following way. Since only four pairs of instants are needed for view-invariant measurement and DTW, the system can iteratively try all the combinations of four pairs of instants. Because wrong correspondences give high errors with DTW, and we only choose the correspondences that give minimal difference, the right four pairs of instant correspondences are kept, and the correspondence of the rest of the points is provided by DTW. Therefore, the wrongly detected instants will not affect the measurement. Figure 3.6 shows some matching results and the correspondence for every 7 points of the trajectories.

This measurement cannot be applied to the walking sequences (Section 2.2.8), since the camera was moving, and we do not currently apply global motion compensation. The epipolar geometry is not preserved in the sequence.

We treat the action trajectories that have only three events separately, because the view invariant dynamic time warping algorithm requires at least four corresponding events

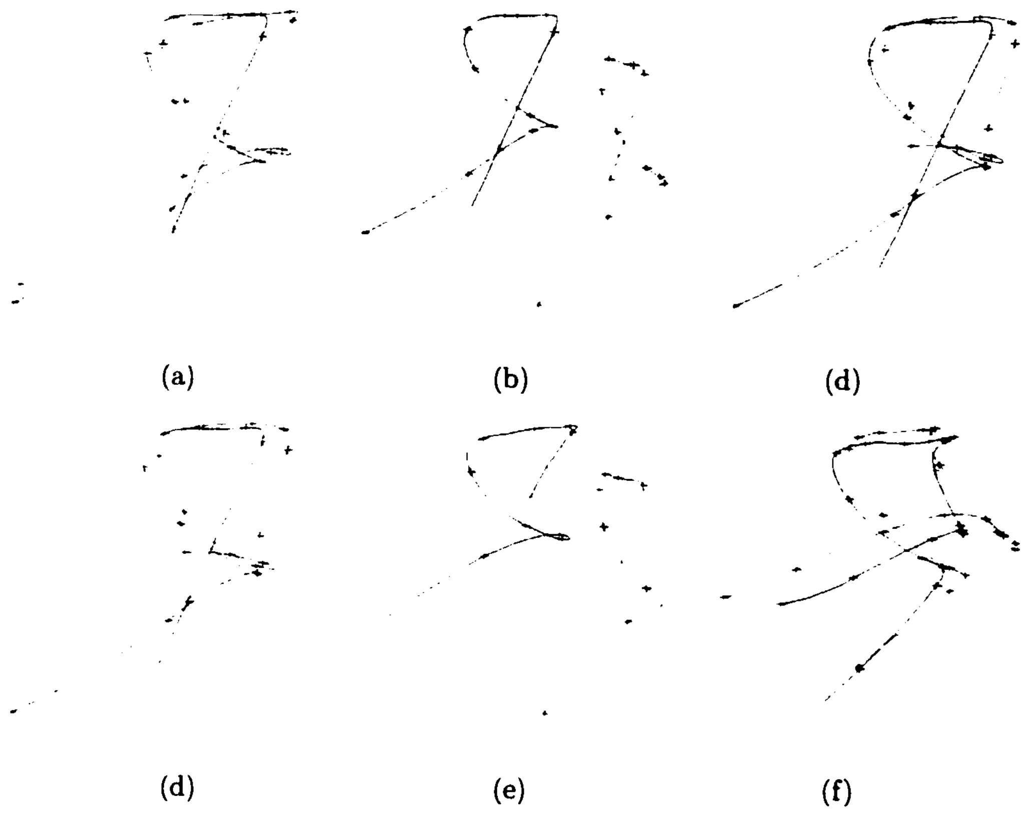


Figure 3.6: Some matching results. The trajectories are shown in different colors, and the red dotted line connect the corresponding points. a) action 1 and action 29, b) action 1 and action 43, c) action 1 and action 38, d) action 29 and action 43. e) action 3 and 6, f) action 3 and 8. (the detailed action descriptions are in Chapter 4).

for each of the trajectories that are being measured. Since the first and the last events represent the hand entering and existing the field of view, only the second event reveals the real phenomenon. Moreover, the interval from the first event to the second event is always approaching, and the interval from the second event to the last one is always receding. They do not show much meaning either. We apply the method discussed in Section 3.2.2 to determine whether this action is the "picking up an object" or "putting down an object".

3.5 Unsupervised Learning

Let's reiterate our goal, which is that the system should achieve both view-invariant recognition of similar actions by different people, and correct categorization of actions into groups without any training process. Is it possible to categorize actions into groups without any training process? If we succeed, then we can set up a new system easily at any place it is needed.

From the discussion of previous work, we can see that not much work has been done in this area. We have solved the problems of representing actions, matching two actions from different viewing directions, and matching two actions with different execution speed. Based on these successes, we can apply an unsupervised algorithm, Transitive Closure Clustering, to discover the groups of instances in which the trajectories are declared as the instances from the same action. Our approach contains two steps:

First, we match each action with all other actions and compute the match score. For each action, we select closely matched actions. All the matches above a certain threshold are eliminated first, and only the three best matches for each action are maintained. If a particular action does not closely match any action of its category, then it is declared a unique action. Its label may change as more evidence is gathered (Tables 5.5 and 5.6).

Secondly, the best matches for individual actions are merged into a compact list using the transitive property. That is, if action 1 is similar to actions 29, 43, and 38; and action

29 is similar to actions 43, 38, and 1; then actions 1, 29, 38, and 43 are all similar actions due to the transitive property. This is implemented by Warshall's algorithm [59].

Warshals algorithm computes the transitive closure of a graph. An adjacency matrix is a representation of a directed graph with n vertices using an $n \times n$ matrix, where the entry at (i,j) is 1 if there is an edge from vertex i to vertex j; otherwise the entry is 0.

The algorithm is described as follows:

```
for k:=1 to N
    for i:=1 to N
        for j:=1 to N
            if ((A[i,k] AND A[k,j]) AND ( dist(i,j) < D))
                A[i,j] = 1;
        end
    end
end
```

The final recognition results are presented in Table 5.8. The system automatically segments video into individual actions, and computes the view-invariant representation for each action. The system is able to incrementally learn different actions starting with no model. It is able to discover different instances of the same action performed by different people, and from different viewpoints.

CHAPTER 4

VIDEO TEMPORAL ALIGNMENT

As we mentioned in Section 1.4, there is a great need for temporally aligning two videos taken from different viewpoints. Inspired by our action recognition research, we propose a view-invariant dynamic temporal alignment approach for videos.

4.1 Related work

There are two main types of approaches for aligning sequences: *sequence-to-sequence* and *trajectory-to-trajectory*. The sequence-to-sequence approach, which is also called direct approach, takes the video frames as an input and applies the computation over all pixels in all frames. On the other hand, the trajectory-to-trajectory approach tracks the movement of the feature points in the field of view, and uses the information contained in the trajectories. The advantages of direct approach include: it can determine for the spatial transformation between sequences more accurately than the trajectory-to-trajectory approach, and it does not require explicit feature detection and tracking. On the contrary, since the trajectories contain explicit geometric information, the trajectory-to-trajectory approach better determines the large spatio-temporal misalignments, can align video sequences acquired by different sensors and is less affected by the background changes. The detailed comparison between these approaches is available in [69, 12]. Since the video sequences in most applications contain a significant spatio-temporal variations, we choose trajectory-to-trajectory approach. As one of the achievements, based on the trajectory information we can align the video sequences, in which different people perform the same action.

Previously, researchers have tried using calibrated/uncalibrated stereo-rigs [21, 30] to recover the projection relationships among the videos. In these approaches, the fundamental matrix is used to find the spatial relationship between the trajectories [14, 75]. However, due to the instability of reconstruction process, those approaches can only be applied to some limited video sequences, such as simultaneously shot videos. To the best of our knowledge, there is no previous method to synchronize two videos of different people performing the same 3D activity at different time employing the fundamental matrix.

Stein [66] achieved the alignment of tracking data obtained from multiple camera assuming homography relationship between the cameras. Stein did not use the trajectory information, but discovered the temporal alignment using exhaustive search among different intervals between video sequences. Due to this, his method computationally quite expensive, and it can only align the videos with a constant time shift.

Giese and Poggio [24] proposed a method to find the spatio-temporal alignment of two video sequences using the dynamic shift of the time stamp of the spatial information. They assumed that a 2D action trajectory can be represented as a linear-combination of prototypical views, and the effect of viewpoint changes can be expressed by varying the coefficients of the linear-combination. Since they did not use the 3D information, this method can only align some simple motion patterns.

Caspi and Irani [12] proposed a direct approach to align two surveillance videos by finding the spatio-temporal transformation that minimizes the sum of squares differences (SSD) between two sequences. They extended the direct approach to the alignment of non-overlapping sequences captured by a stereo rig [13]. In these video sequences, the same motion induces "similar" changes in time. This correlated temporal behavior was used to recover the spatial and temporal transformations between sequences. They also proposed a trajectory-to-trajectory approach for alignment of sequences captured by cameras with significant different viewpoints [14]. In this method the alignment of trajectories is based on computation of the fundamental matrix. Their approaches can only be used for applications, in which the time shift between the video sequences is constant or is a linear function, and will fail for videos with a dynamic time shift.

Wolf and Zomet [75] proposed a method for self calibrating a moving rig. During the movement, the viewing angles between cameras and the time shift are fixed, but the internal camera parameters are allowed to change.

Extensive research has been done in the area of action recognition. Various approaches have been applied to discover the viewpoint difference between videos [67], to measure the difference between actions using view-invariant characteristics [51], or to find the period of the cyclic motion [61].

From this brief review, we can conclude that the existent methods are not appropriate for alignment of video sequences containing the complex 3D motion with significant spatio-temporal variation. In this chapter, we propose a method, which is based on the epipolar constraint, but does not need explicit reconstructing of the 3D relationships between videos. This method can temporally align videos containing 3D actions with large spatio-temporal variance. Since it is a well-studied problem to reconstruct the spatial alignment of video sequences given the correspondent frames, we do not discuss the spatial registration here. The results of experiments show that our method is much more stable, and it can be used in many applications.

4.2 View-invariant Measure

First, let us consider the similarity measure between 2D trajectories, which are represented as $\{(u_1, v_1), (u_2, v_2), \dots, (u_t, v_t)\}$ and $\{(u'_{T(1)}, v'_{T(1)}), (u'_{T(2)}, v'_{T(2)}), \dots, (u'_{T(t)}, v'_{T(t)})\}$. The relationship between a point (X_i, Y_i, Z_i) in 3D, trajectory and its 2D projection (u_i, v_i) is defined as follows:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = P \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}, i = 1, 2, \ldots, t, \tag{4.1}$$

where P is the projection matrix (camera model).

In the Equation 4.1, the general camera projection can be modeled using the following perspective matrix

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{23} & p_{33} & p_{34} \end{bmatrix}.$$

Readers could reference to any computer vision textbook to find the properties of this projection matrix [27]. In this paper, we focus on the epipolar geometry, which represents the extrinsic projective geometry between views.

Using the perspective model, for a pair of matching points $(u_i, v_i) \leftrightarrow \left(u'_{T(i)}, v'_{T(i)}\right)$ in two trajectories, the fundamental matrix (a 3 by 3 matrix), $\mathbf{F}$, is defined by the equation

$$s(i) = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}^T \mathbf{F} \begin{bmatrix} u'_{T(i)} \\ v'_{T(i)} \\ 1 \end{bmatrix} = 0, \tag{4.2}$$

Therefore, given a fundamental matrix, we can use Equation (4.2) to measure the similarity between trajectories, such that $\sum s(i)$ for all points is minimized.

It is a well known fact that the computation of fundamental matrix is not robust. The human motion at different time can further worsen the stability. If a person performs the same movement differently, previous approaches [14, 75] will fail to align these two video sequences. Therefore, we propose a novel approach, which avoids the computation of the fundamental matrix.

Given sufficiently many point matches between two trajectories, Equation (2.5) can be used to compute the unknown matrix F from the following equation:

$$\mathbf{Mf} = \begin{bmatrix} u'_{T(1)}u_1 & \cdots & u'_{T(t)}u_t \\ u'_{T(1)}v_1 & \cdots & u'_{T(t)}v_t \\ u'_{T(1)} & \cdots & u'_{T(t)} \\ v'_{T(1)}u_1 & \cdots & v'_{T(t)}u_t \\ v'_{T(1)}v_1 & \cdots & v'_{T(t)}v_t \\ v'_{T(1)} & \cdots & v'_{T(t)} \\ u_1 & \cdots & u_t \\ v_1 & \cdots & v_t \\ 1 & \cdots & 1 \end{bmatrix}^T \mathbf{f} = 0 \tag{4.3}$$

where $\mathbf{f}$ is the rearrangement of the elements of the fundamental matrix:

$$\mathbf{f} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{bmatrix}^T \tag{4.4}$$

Let us denote the observation matrix by $\mathbf{M}$, which is constructed using the coordinates of points of two 2D trajectories. Since (4.3) is a homogenous equation, for a solution of $\mathbf{f}$ to exist, matrix $\mathbf{M}$ must have rank at most eight. However, due to the noise or the correspondence error, the rank of matrix $\mathbf{M}$ may not be exactly eight. In this case the 9^{th} singular value of $\mathbf{M}$ estimates the necessary perturbation of coordinates of each point in matrix $\mathbf{M}$ to produce two projections of the same 3D trajectory. Therefore, we can use the 9^{th} singular value of matrix $\mathbf{M}$ to measure the similarity between two trajectories. The smallest value of the singular value of $\mathbf{M}$ corresponds to the best match of trajectories, and we denote it as dis.

We generated two trajectories, selected nine points from each trajectory and put them into the observation matrix $\mathbf{M}$. The 9^{th} singular value increases dramatically when there is a large change in the x and y coordinates of one point(Figure 4.1). Therefore, if the points are spread far enough from each other (that is the points are not clustered in one specific

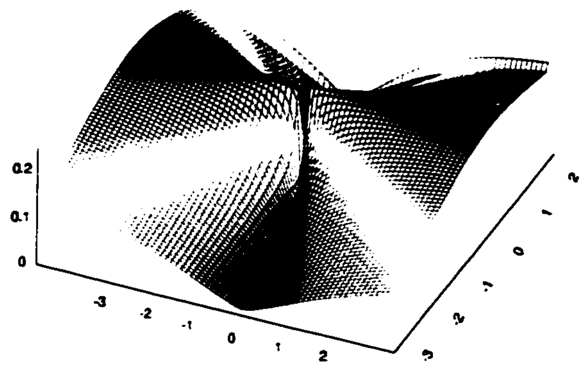


Figure 4.1: The 9th singular value of according to the change of x, y coordinates of (u_9, v_9) in the observation matrix $\mathbf{M}$.

location). by picking the nine corresponding points from each trajectory, we can decide whether two trajectories match or not. Although, the rank constraint we are using here is only necessary condition of the matching of two trajectories, which means that there might be multiple matching results for trajectories, due to the distribution of the last singular value of $\mathbf{M}$. it is still applicable for video alignment. Moreover, since the trajectory contains the temporal information. we can also use this temporal information to align trajectories. We discuss the use of temporal information for alignment in the Section 4.3.

In some applications it is reasonable to assume that the time-warping function is linear. $T(i) = ai + b$. Then a and b, which are the parameters of the time-warping function, can be found by using the exhaustive search and by minimizing the *dist* measure. And to model more complicated time-warping functions, a higher order polynomial must be used. However, these types of time-warping function have very limited applications, such as synchronizing two video sequences that are captured simultaneously, or synchronizing of stereo cameras. Generally, this approach fails to align video sequences shot at different times and contain human activities, since the time-warping function for human activities can not be modelled by a simple polynomial.

Compared this measure with the similarity measure discussed in Section 3.3, there are two main differences: 1) the camera model are different, this measure uses perspective projection model and needs at least 9 points to compute the measure, while previous approach uses affine projection model and needs only 5 points to compute the measure; 2) the rank constraint for perspective projection is a necessary condition for the Equation 2.7 having a solution of **F** such that the points are matched, while the theorem 3.3 claim the fourth singular value measure is both necessary and sufficient to determine the points are matched. However, affine camera model is a simplified projection model, so it can not model the 3D-to-2D project accurately if the actions contain large amount of variance along the camera axis direction. Therefore, for alignment of complex human actions we use the proposed measure in this section, while for human action recognition, the previous approach is more appropriate, since the measure under necessary condition might give wrong answer for recognition system in some situation.

4.3 View-invariant Dynamic Time Warping

In Section 3.4, we proposed a action recognition approach using the view-invariant dynamic time warping, which uses affine camera model to achieve measure the similarity between action trajectories. Moreover, the dynamic time warping path can provide the correspondence of the points along the trajectories (Section 3.4). Therefore, DTW is a good solution for finding the non-linear temporal warping function for two video sequences. In Section 4.2, we proposed a new measure for trajectory points using perspective camera model, so we replace the affine similarity measure used in view-invariant DTW (Section 3.4) with the current measure to achieve aligning more complicated action sequences, and modify the view-invariant dynamic time warping algorithm as follow:

(1) We detect eight corresponding points between the first frames of two videos, and denote the image coordinates as $(x'_1, y'_1), ..., (x'_8, y'_8)$ and $(x_1, y_1), ..., (x_8, y_8)$.

(2) Track the feature points in two videos to acquire trajectories I=$\{(u'_1, v'_1),\ldots,(u'_n, v'_n)\}$ and J=$\{(u_1, v_1),\ldots,(u_m, v_m)\}$. In our experiments we used the mean-shift tracker [16].

(3) For each pair of the corresponding points in the trajectories, construct the 9×9 observation matrix:

$$M_O = \begin{bmatrix} x'_1x_1 & x'_2x_2 & \cdots & x'_8x_8 & u'_iu_j \\ x'_1y_1 & x'_2y_2 & \cdots & x'_8y_8 & u'_iv_j \\ x'_1 & x'_2 & \cdots & x'_8 & u'_i \\ y'_1x_1 & y'_2x_2 & \cdots & y'_8x_8 & v'_iu_j \\ y'_1y_1 & y'_2y_2 & \cdots & y'_8y_8 & v'_iv_j \\ y'_1 & y'_2 & \cdots & y'_1 & v'_i \\ x_1 & x_2 & \cdots & x_8 & u_j \\ y_1 & y_2 & \cdots & y_8 & v_j \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}^T . \qquad (4.5)$$

(4) Execute the classic DTW algorithm using the distance measure between the points of two trajectories at t_i and the t_j respectively: $d_{(i,j)} = dis(i, j)$, where $dis(i, j) = \sigma_9$ is the 9^{th} singular value of the matrix $\mathbf{M_O}$ in step 3.

(5) Generate the time-warping function $T(i) = i, i = 1, \ldots, n$ by back tracing the path that minimize the value of $E(i, j)$ from the upper-left corner of the matrix $\mathbf{E}$. If the cell $E(i, j)$ is on the warping path, it means i^{th} point of trajectory I corresponds to the j^{th} point of trajectory J.

Note that the DTW can establish the correspondence "on the fly", which means that it determines the best warping path to element $E(i, j)$. To achieve more robust measurement for the $E(i, j)$, we put the previously found corresponding points up to i and j in the observation matrix $\mathbf{M}$, and update the original observation matrix $\mathbf{M_O}$ Equation 4.3. The matrix $\mathbf{M_R}$ is given as follows:

$$M_R = \left[\frac{M_P}{M_O}\right]; \quad M_P = \begin{bmatrix} u'_1u_1 & u'_2u_2 & \cdots & u'_{i-1}u_{j-1} \\ u'_1v_1 & u'_2v_2 & \cdots & u'_{i-1}v_{j-1} \\ u'_1 & u'_2 & \cdots & u'_{i-1} \\ v'_1u_1 & v'_2u_2 & \cdots & v'_{i-1}u_{j-1} \\ v'_1v_1 & v'_2v_2 & \cdots & v'_{i-1}v_{j-1} \\ v'_1 & v'_2 & \cdots & v'_{i-1} \\ u_1 & u_2 & \cdots & u_{j-1} \\ v_1 & v_2 & \cdots & v_{j-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T \tag{4.6}$$

This algorithm is not affected by the change in the viewpoint, since the similarity measure does not depend on the viewpoint, and it dynamically computes the non-linear time-warping function between the two 2D trajectories.

4.4 Temporal Coarse-to-fine refinement

As we mentioned in Section 4.2, the similarity measure does not require the explicit computation of the fundamental matrix, therefore the $rank(\mathbf{M}) = 8$ is only a necessary condition to determine whether the two set of points match or not. It can be noticed that the 9^{th} singular value of the observation matrix shows an ambiguity if there are many points are very close to the correct one. Therefore, the matching algorithm might give wrong results due to the noise in the trajectory. The DTW is also sensitive to the errors, such that if the warping function is incorrect at $E(i, j)$, then the error will be propagated to the rest of the warping path. To solve these problems we use temporal pyramids of trajectories.

In the temporal pyramid, the higher level has less number of points, and the distance between consecutive points is relatively greater than the one in the lower level. The larger distance between points generates the larger change of the last singular value. Consequently,

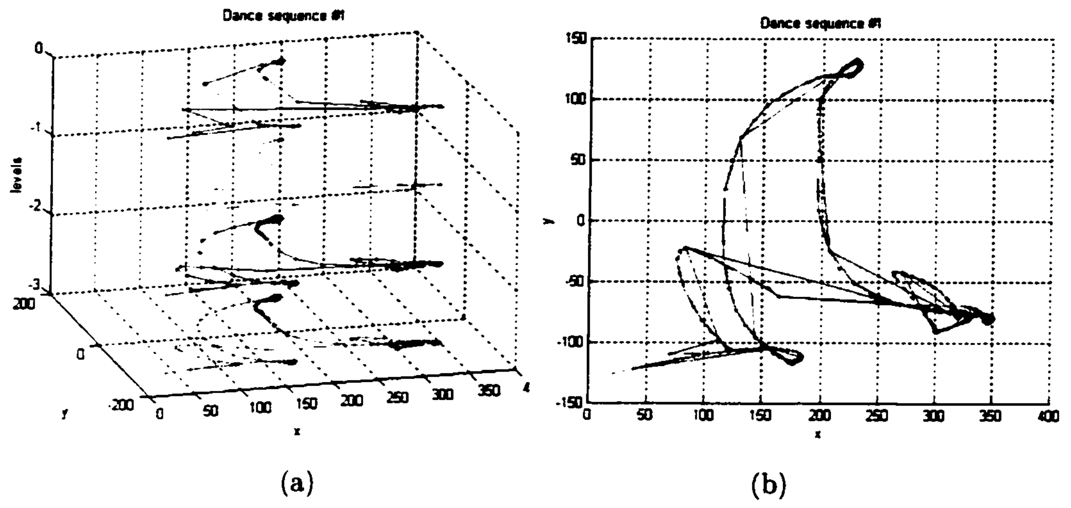


Figure 4.2: (a) The temporal pyramid of a trajectory, the upper level contains fewer points and lower level contains more points in the trajectory. (b) The up-side-down view of the temporal pyramid in (a).

the significant variation of the last singular value determines matching points without the ambiguity. Furthermore, the higher level of the pyramid provides a constraint for the lower level by propagating point correspondence. So by using the coarse-to-fine approach, we can prevent the error propagation to the rest of the time-warping function.

We propose a novel coarse-to-fine refinement for the view-invariant DTW algorithm:

(1) For trajectory I use spline to sub-sample the trajectory by factor of 2, such that $length(\mathbf{I}^k) = 0.5 * length(\mathbf{I}^{(k+1)})$ (length() is the total number of points in the trajectory), where k is the index for the level of the pyramid and the highest level is labelled as $k = 0$. The same approach is applied to the trajectory J. The coordinates of i^{th} point in trajectory I^k is represented as $((u_i')^k, (v_i')^k)$, and the j^{th} point in trajectory J^k is represented as $((u_j)^k, (v_j)^k)$. Figure 4.2 shows the develop of levels of temporal pyramid.

(2) At the top level ($k = 0$) compute view-invariant DTW (Section 4.3) using trajectories I^0 and J^0.

(3) For $k+1$ level, generate the observation matrix, with the first rows are the rows of observation matrix **M** at the k level. The matrix **M** is arranged as follows:

$$M_R = \left[\begin{array}{c} M_P \\ \hline M_Q \\ \hline M_O \end{array}\right]$$

$$M_P = \begin{bmatrix} (u'_1)^k(u_1)^k & (u'_2)^k(u_2)^k & \cdots & (u'_{tn})^k(u_{tm})^k \\ (u'_1)^k(v_1)^k & (u'_2)^k(v_2)^k & \cdots & (u'_{tn})^k(v_{tm})^k \\ (u'_1)^k & (u'_2)^k & \cdots & (u'_{tn})^k \\ (v'_1)^k(u_1)^k & (v'_2)^k(u_2)^k & \cdots & (v'_{tn})^k(u_{tm})^k \\ (v'_1)^k(v_1)^k & (v'_2)^k(v_2)^k & \cdots & (v'_{tn})^k(v_{tm})^k \\ (v'_1)^k & (v'_2)^k & \cdots & (v'_{tn})^k \\ (u_1)^k & (u_2)^k & \cdots & (u_{tm})^k \\ (v_1)^k & (v_2)^k & \cdots & (v_{tm})^k \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T$$

$$M_Q = \begin{bmatrix} (u'_1)^{k+1}(u_1)^{k+1} & (u'_2)^{k+1}(u_2)^{k+1} & \cdots & (u'_{i-1})^{k+1}(u_{j-1})^{k+1} \\ (u'_1)^{k+1}(v_1)^{k+1} & (u'_2)^{k+1}(v_2)^{k+1} & \cdots & (u'_{i-1})^{k+1}(v_{j-1})^{k+1} \\ (u'_1)^{k+1} & (u'_2)^{k+1} & \cdots & (u'_{i-1})^{k+1} \\ (v'_1)^{k+1}(u_1)^{k+1} & (v'_2)^{k+1}(u_2)^{k+1} & \cdots & (v'_{i-1})^{k+1}(u_{j-1})^{k+1} \\ (v'_1)^{k+1}(v_1)^{k+1} & (v'_2)^{k+1}(v_2)^{k+1} & \cdots & (v'_{i-1})^{k+1}(v_{j-1})^{k+1} \\ (v'_1)^{k+1} & (v'_2)^{k+1} & \cdots & (v'_{i-1})^{k+1} \\ (u_1)^{k+1} & (u_2)^{k+1} & \cdots & (u_{j-1})^{k+1} \\ (v_1)^{k+1} & (v_2)^{k+1} & \cdots & (v_{j-1})^{k+1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T$$

$$M_O = \begin{bmatrix} x'_1x_1 & x'_2x_2 & \cdots & x'_8x_8 & (u'_i)^{k+1}(u_j)^{k+1} \\ x'_1y_1 & x'_2y_2 & \cdots & x'_8y_8 & (u'_i)^{k+1}(v_j)^{k+1} \\ x'_1 & x'_2 & \cdots & x'_8 & (u'_i)^{k+1} \\ y'_1x_1 & y'_2x_2 & \cdots & y'_8x_8 & (v'_i)^{k+1}(u_j)^{k+1} \\ y'_1y_1 & y'_2y_2 & \cdots & y'_8y_8 & (v'_i)^{k+1}(v_j)^{k+1} \\ y'_1 & y'_2 & \cdots & y'_1 & (v'_i)^{k+1} \\ x_1 & x_2 & \cdots & x_8 & (u_j)^{k+1} \\ y_1 & y_2 & \cdots & y_8 & (v_j)^{k+1} \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}^T .$$

(4) Continue the measure of matching the points of trajectories $I^{(k+1)}$ and $J^{(k+1)}$ as step 3 till the lowest level.

When compute the measure of matching between the point I_i^{k+1} and J_j^{k+1}, if the k level computation already gives the result that I_i^k and J_j^k are corresponding each other, because the row for I_i^{k+1} and J_j^{k+1} is just repeating the previous row in the observation matrix, the last singular value will be very small. Therefore, the warping path will go through $E^{k+1}(i,j)$ for sure. Thus, the correspondence of points from the upper level is smoothly transitioned to the next level of the pyramid. The ambiguity is resolved and the matching is well guided by the upper levels of pyramid.

CHAPTER 5

EXPERIMENTS

5.1 Action Recognition

We have performed experiments on 60 different action clips performed by seven individuals.The trajectories are given in Figures 5.2-5.5 and described in Tables 5.1-5.4 (Please visit www.cs.ucf.edu/~rcen/research.html for video sequences, results, etc.). People performing the actions were not given any instructions, and entered and exited the field of view from arbitrary directions. While capturing the action clips, the location of the camera was changed from time to time to obtain actions from different viewpoints. We digitized the clips captured by a video camera recorded at 24 fps. The current implementation of the system only deals with one action agent in the scene. For labeling the detected skin blobs as head and hand, we assume the speed of the hand is higher than the speed of the head. This scheme works for the test sequence we used in our experiments, however one can use more complex schemes for labeling them correctly.

Our system does not require any training. We start with an empty "known actions database". For each unique action we update the "known actions database" by adding the representation of that action to the database. The system automatically detects hands using skin detection and generates trajectories of actions by the mean-shift tracking method. In Figures 5.6 and 5.7, we show two examples of actions from the data set along with the trajectories obtained using the proposed method. Once the trajectories are obtained, we compute the curvature given in Equation 2.7 to obtain the view- invariant representation for the action.

The matching of the input action with the actions in the "known actions database" is done using the method discussed in Section 3.4. We present the performance of the method in Tables 5.5 and 5.6 by analyzing the three best matches. Only two of the all actions (actions 31 and 45) have three false matches. Among the rest, three actions (33, 36, 59) are partially correct, ie. best three matches include the correct and wrong instances of actions.

One of the reasons for wrong matches is a noisy trajectory due to the low sampling rate of the continuous hand motion, i.e. for some actions, some of the instants may be missing and some may have extra instants. This cased the system to fail to match correct actions for action 37. Another reason for degraded performance on some of the actions is based on the affine camera model, which is an approximation of the real projection from 3-D world to 2-D image coordinates. The affine model results in unrealistic matching for some actions. Due to this, action 31 and action 36 are matched with other actions. However, an analysis on actions 31 and 36 shows that they are partially matched with an opening action, such as 38. We expect that using projective model will improve the matching performance, but since this model requires more instants, it is not applicable for actions which have only a few instants.

We compared these results with the results of a view-invariant matching algorithm that uses only event information [57]. Without dynamic time warping, the matching algorithm gives three wrong matching results to 3 actions (actions 31, 36 and 58) and partial wrong matching results to 7 actions (actions 4, 8, 41, 43, 48, 59 and 60). Therefore, using the full trajectory information (both spatial and temporal information) improves the matching result dramatically. We can conclude that using more information helps improve the performance of algorithm.

In Table 5.8, we show the performance of the Warshall algorithm for learning the actions by grouping them using the transitivity property. Actions 5, 15, 20, 26, 27, 28, 30, 32, 34, 36, 37, 39, 41, 42, 47, and 52 are correctly detected as unique actions and are not grouped with any other action. For the "opening cabinet" action the proposed system correctly grouped the actions 1,14,16,21,29,38 and 43, missed only action 4 and included action 31 as a false positive. Note that even though trajectories of these actions, shown in Figures 5.2-5.5, are different, due to the strength of our representation, the system was able to learn that they represent the same action. Similarly, the system was able to correctly match the actions 3, 6, 18 and 23 as the "put down the object, and then close the door" action. Actions 7, 8, 33 and 48 are learned as one group of actions, which represents "open the cabinet door, wait, then close the door". The system learned actions 49, 50, 51 and 53 as a group of "pick up an object from the floor and put it on the desk", including action 13 as a false positive. Actions 54 and 56 are learned as the "erase the white board" action, with actions 55 and 57 missing. Actions 58 and 60 are learned as "Pour water into a cup" and the system missed action 59. The action trajectories that have only three events are recognized by using the method in Section 3.2.2. The actions 2, 9, 11, 19, 22 and 24 are correctly recognized as "picking up", and the actions 10, 12, and 25 are correctly recognized as "putting down". Only actions 17 and 44 are wrong, because the object is too small or the object is occluded by some other object. Note that all these matches are based on only a single instance of an action. Therefore the performance of the proposed approach is remarkable.

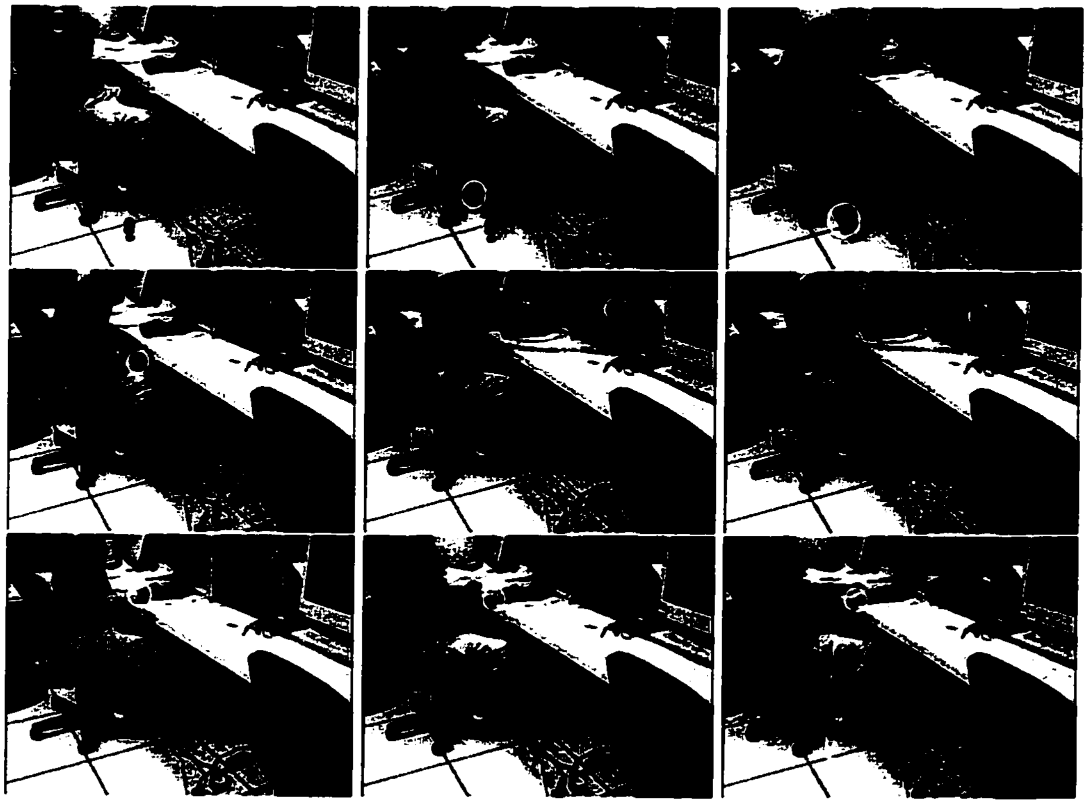


Figure 5.1: Action 37: pick up an object from the floor and put it down on the desk. Every 8th frame of the sequence is shown. The hand is highlighted with a white circle,and its trajectory is superimposed on the last frame.

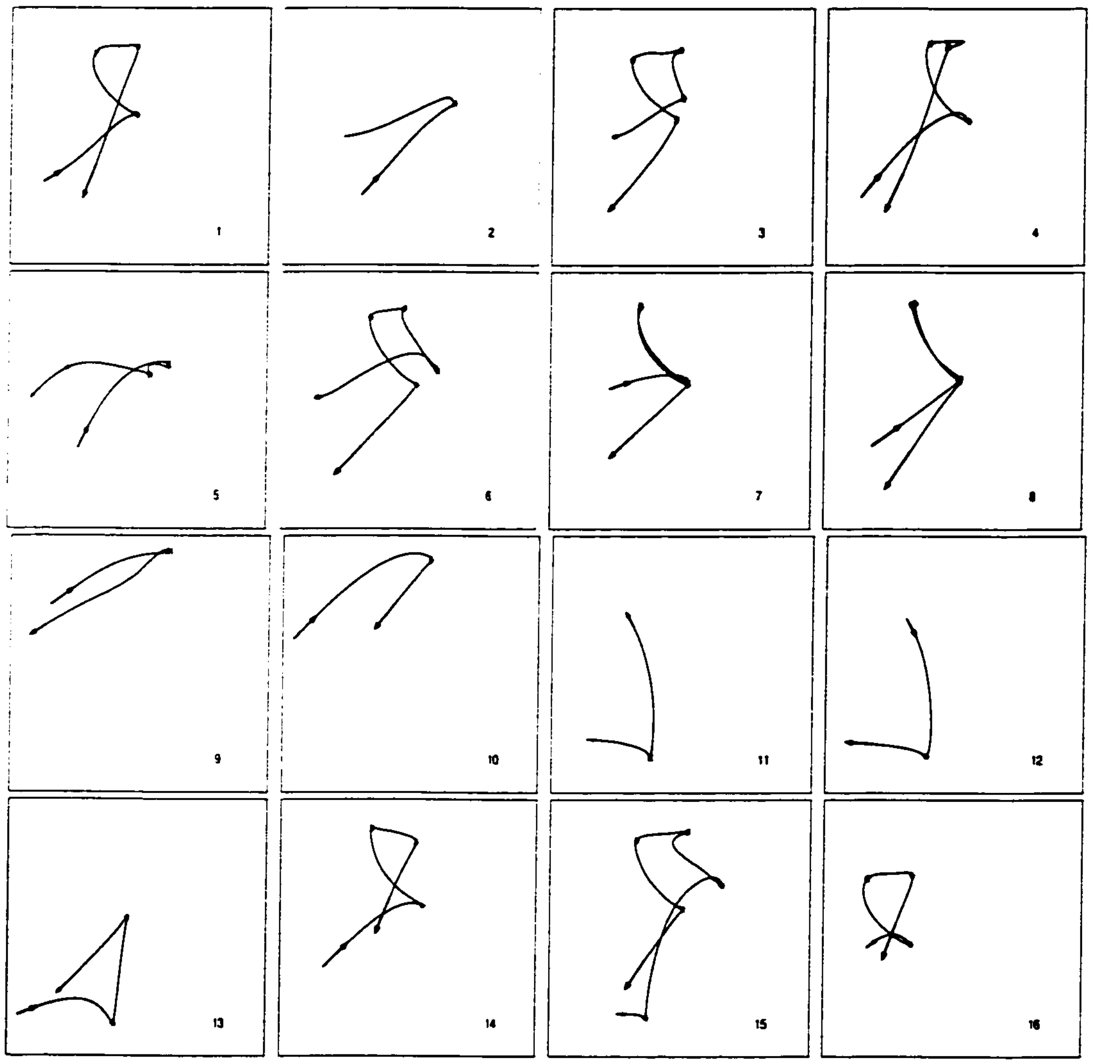


Figure 5.2: Trajectories of actions 1 to 16. The *instants* are shown with "$*$". And the definitions of these actions are given in Table 5.1.

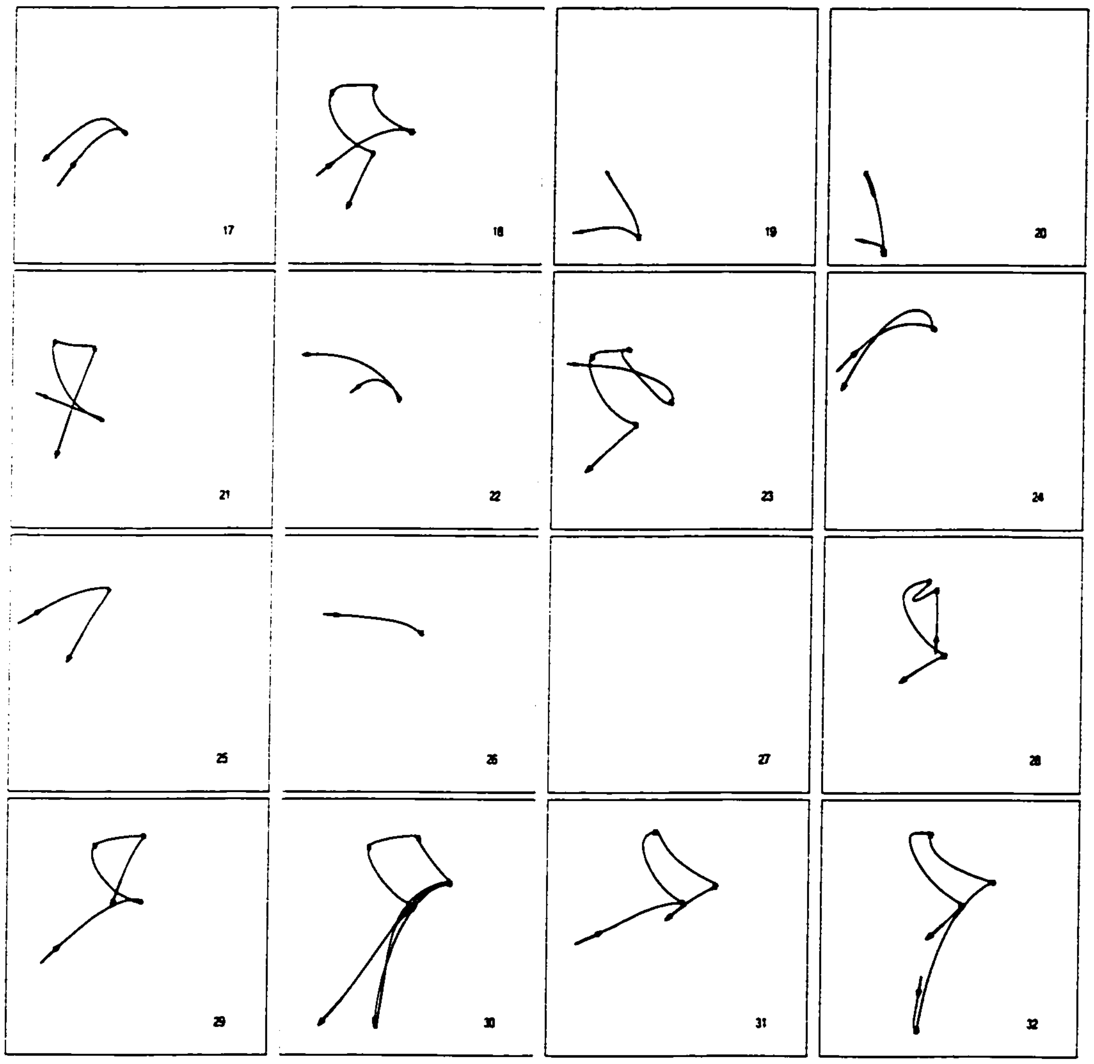


Figure 5.3: Trajectories of actions 17 to 32. The *instants* are shown with "*". And the definitions of these actions are given in Table 5.2.

Table 5.1: Description of actions 1 to 16 in Figure 5.2.

1st	Open the cabinet.
2nd	Pick up an object (umbrella) from the cabinet.
3rd	Put down the object in cabinet, then close the door.
4th	Open the cabinet, with touching the door an extra time.
5th	Pick up an object (disks) with twisting hand around.
6th	Put back the object (disks) and then close the door.
7th	Open the cabinet door, wait, then close the door.
8th	Open the cabinet door, wait, then close the door.
9th	Pick up an object from top the of the cabinet.
10th	Put the object back on the top of the cabinet.
11th	Pick up an object from the desk.
12th	Put the object back on the desk.
13th	Pick up an object, then make random motions.
14th	Open the cabinet.
15th	Pick up an object, put it in the cabinet, then close the door.
16th	Open the cabinet.

Table 5.2: Description of actions 17-32 in Figure 5.3.

17^{th}	Pick up an object (umbrella) from the cabinet.
18^{th}	Put the object (umbrella) back in the cabinet.
19^{th}	Pick up a bag from the desk.
20^{th}	Make random motions.
21^{th}	Open the cabinet.
22^{th}	Pick up an object (a bag of disks).
23^{th}	Put down an object (a bag of disks) back in the cabinet, then close the door.
24^{th}	Pick up an object from the top of the cabinet.
25^{th}	Put the object back in the cabinet top.
26^{th}	Make random motions with two hands.
27^{th}	Continue the action 26.
28^{th}	Close the door, with some random motion.
29^{st}	Open the cabinet.
30^{nd}	Pick up an object (remote controller) from the cabinet, put it down on the desk, pick up another object (pencil) from the desk, put it in the cabinet, then close the door.
31^{rd}	Open the cabinet door, with the door half pushed, pick up an object (pencil) from the cabinet.
32^{th}	Pick up an object (remote controller) from the desk, put it in the cabinet, then close the door.

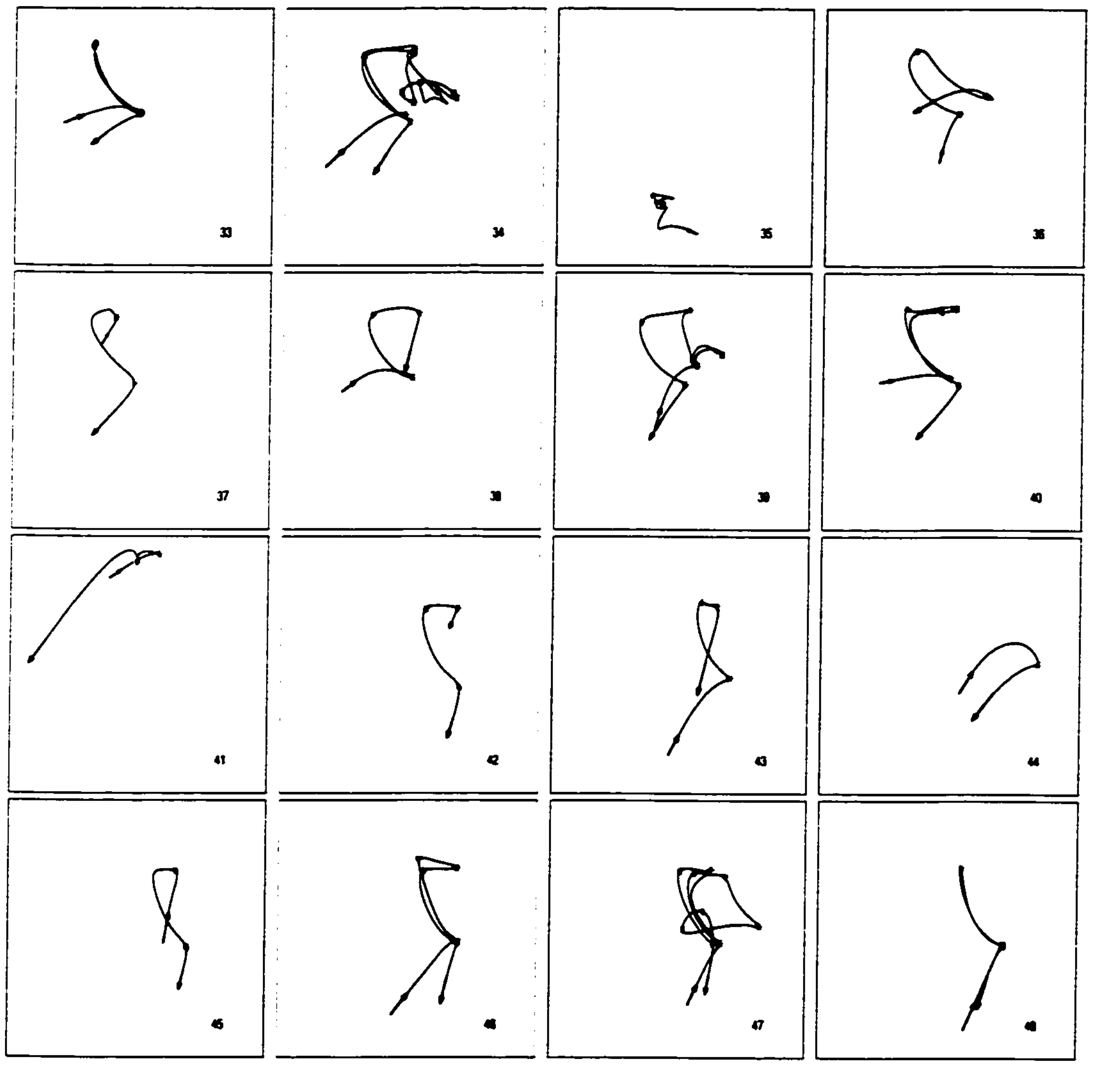


Figure 5.4: Trajectories of actions 33 to 48. The *instants* are shown with "$*$". And the definitions of these actions are given in Table 5.3.

Table 5.3: Description of actions 33 to 48 in Figure 5.4.

33^{th}	Open the cabinet door, wait, then close the door.
34^{th}	Open the cabinet door, make random motions, then close the door.
35^{th}	Pick up some objects.
36^{th}	Open the door, pick up an object, with the door half opened.
37^{th}	Close the half opened door.
38^{th}	Open the cabinet door.
39^{th}	Pick up an object, move it within the cabinet, pick up another object, move it, then close the door.
40^{th}	Open the cabinet door, wait, then close the door.
41^{th}	Pick up an object from the top of the cabinet.
42^{th}	Close the cabinet.
43^{th}	Open the cabinet.
44^{th}	Put down a disk.
45^{th}	Close the half closed door.
46^{st}	Open the door, wait, then close the door.
47^{nd}	Open the cabinet door, pick up an object, then put it back, then close the cabinet door.
48^{rd}	Open, then close the cabinet door.

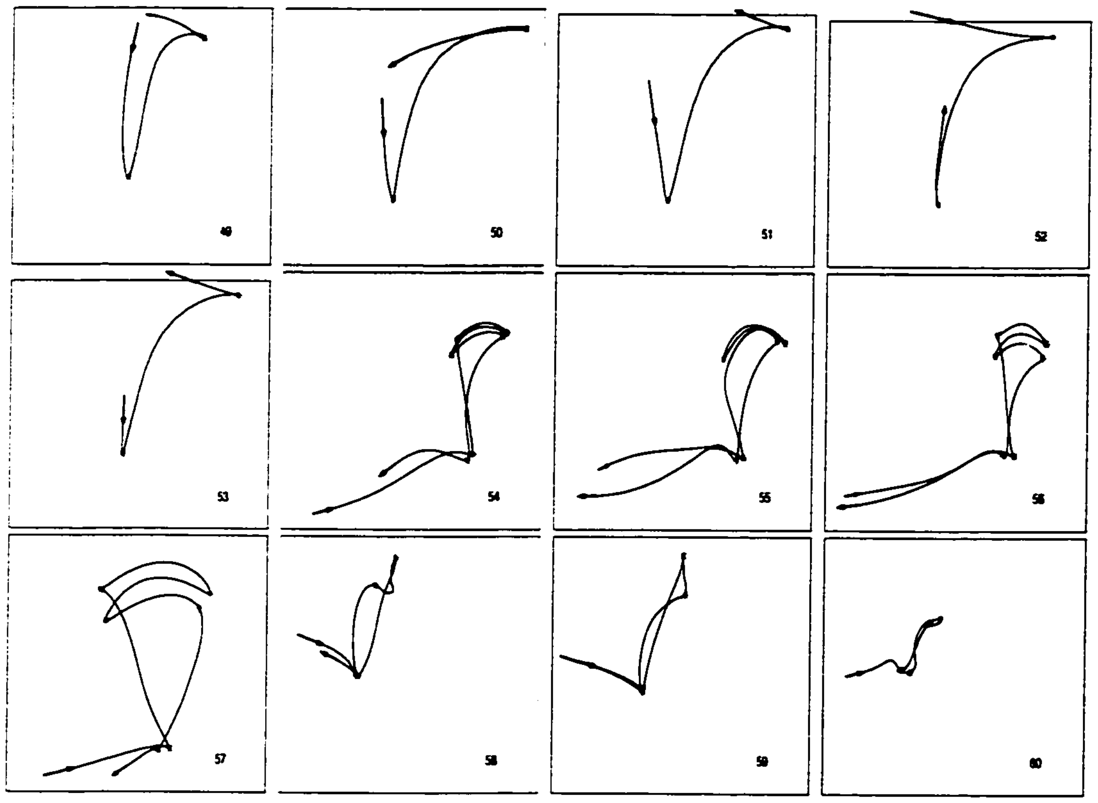


Figure 5.5: Trajectories of actions 49 to 60. The *instants* are shown with "$*$", the definitions of these actions are given in Table 5.4.

Table 5.4: Description of actions 49 to 50 in Figure 5.5.

49th	Pick up an object from the floor and put it on the desk.
50th	Pick up an object from the floor and put it on the desk.
51th	Pick up an object from the floor and put it on the desk.
52th	Pick up an object from the desk and put it on the floor.
53th	Pick up an object from the floor and put it on the desk.
54th	Erase the white board.
55th	Erase the white board.
56th	Erase the white board.
57th	Erase the white board.
58th	Pour water into a cup.
59th	Pour water into a cup.
60th	Pour water into a cup.

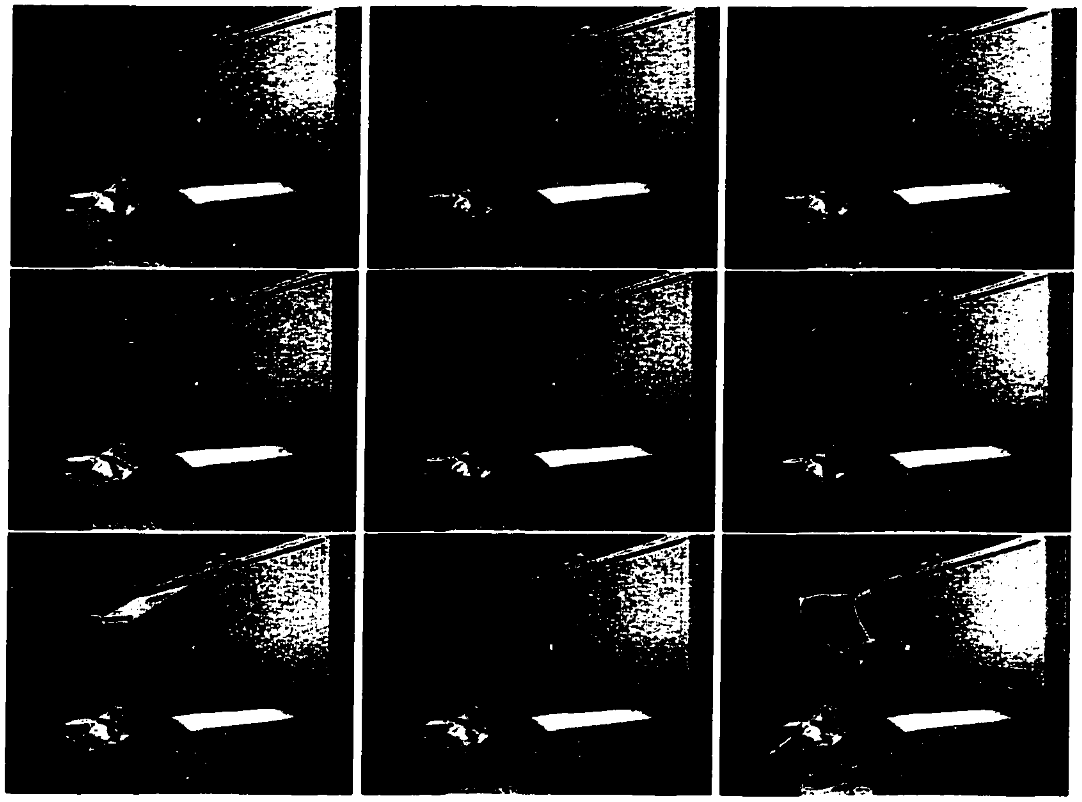


Figure 5.6: Action 2: put down the object in the cabinet, then close the door. Every 15th frame of the sequence is shown. The hand is highlighted with a white circle, and its trajectory is superimposed on the last frame.

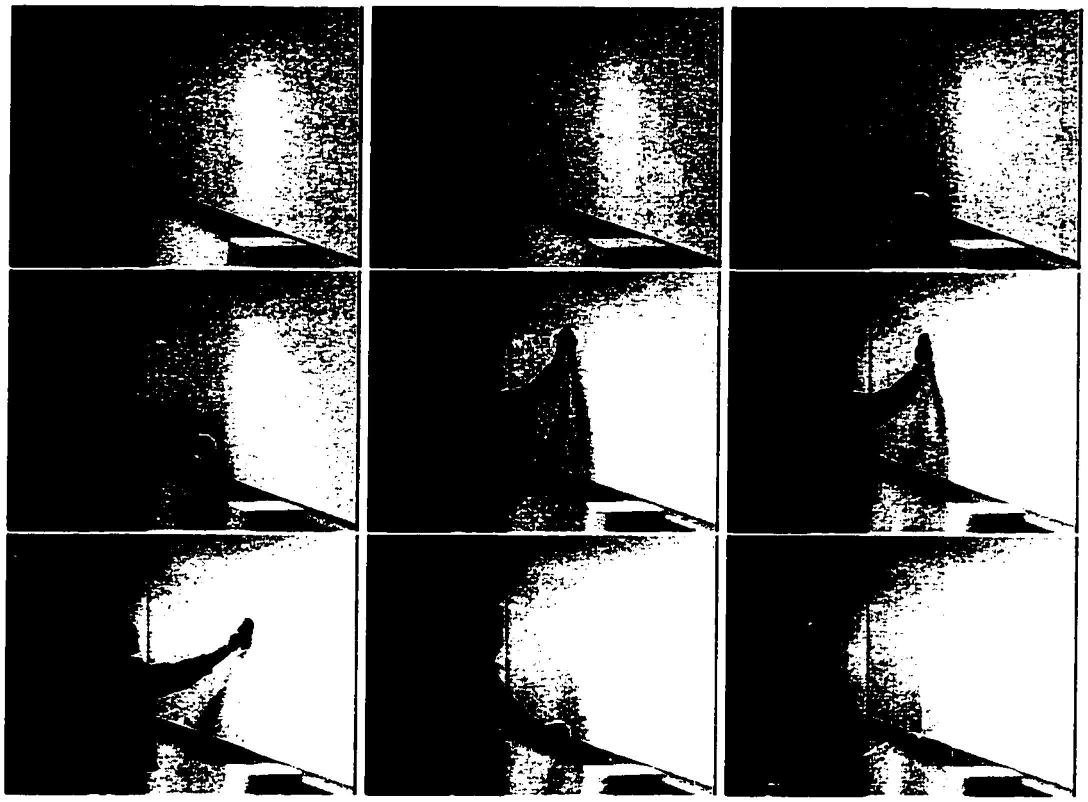


Figure 5.7: Action 43: erase the white board. Every 12th frame of the sequence is shown. The hand is highlighted with a white circle, and its trajectory is superimposed on the last frame. The trajectory and its description are in Figure 2.19 a.

Table 5.5: The matching results and evaluations (part 1).

Actions	3 Best matches	Evaluation and comments	3 Best matches by instants-only matching
1	29 43 38	Correct	38 29 14
2	Pick up	Correct	Pick up
3	18 23 6	Correct	18 6 23
4	1 14 16	Correct	**36** 29 14
5		Correct	
6	18 3 23	Correct	23 3 18
7	48 33 8	Correct	33 8 48
8	48 33 7	Correct	33 7 **60**
9	Pick up	Correct	Pick up
10	Put down	Correct	Put down
11	Pick up	Correct	Pick up
12	Put down	Correct	Put down
13		Unique action	
14	43 16 1	Correct	16 1 29
15		Unique action	
16	14 29 1	Correct	38 14 29
17	**Pick up**	Object hidden	**Pick up**
18	6 3 23	Correct	3 23 6
19	Pick up	Correct	Pick up
20		Unique action	

Table 5.6: The matching results and evaluations (part 2).

Actions	3 Best matches	Evaluation and comments	3 Best matches by instants-only matching
21	43 38 16	Correct	14 38 16
22	Pick up	Correct	Pick up
23	6 3 18	Correct	18 6 3
24	Pick up	Correct	Pick up
25	Put down	Correct	Put down
26		Unique action	
27		Unique action	
28		Unique action	
29	43 38 1	Correct match	1 16 4
30		Unique action	
31	**43 38 29**	Incorrect	**43 16 38**
32		Unique action	
33	48 7 **59**	One wrong	8 7 48
34		Unique action	
35	Put down	The action is confusing	Put down
36	**43** 31 **48**	Two wrong	**43 16 38**
37		Unique	
38	21 16 1	Correct	1 16 29
39		Unique action	
40		**46** is missing	

Table 5.7: The matching results and evaluations (part 3).

Actions	3 Best matches	Evaluation and comments	3 Best matches by instants-only matching
41	**35**	Unique action	
42		Unique action	
43	14 29 1	Correct	**31** 14**36**
44	**Pick up**	Object is too small	**Pick up**
45		Unique action	
46		**40** is missing	
47		Unique action	
48	33 8 7	Correct	**59** 8 7
49	51 53 50	Correct	51 53 50
50	51 53 50	Correct	51 53 50
51	50 53 49	Correct	50 53 49
52		Unique action	
53	51 49 50	Correct	51 49 50
54	56 57	Correct	56 57
55		**Incorrect** one instant missing	
56	54 57	Correct	54 57
57	56 54	Correct	56 54
58	60 59	Correct	**48 33**
59	60 **33**	One wrong	**48** 60
60	58 59	Correct	59 **8 48**

Table 5.8: The detection of action groups.

Action transitive closure	*Evaluation and comments*
1 14 16 21 29 38 43 **31**	**31** shouldn't belong to the group, **4** should have been in the group
3 6 18 23	Correct grouping
7 8 33 48	Correct grouping
35 41 45	**Incorrect grouping**
49 50 51 53 **13**	**31** shouldn't belong to the group,
54 65	**55 57** are missing
58 60	**59** is missing
2 9 11 19 22 24 **17 44**	the object is too small
10 12 25	
5 15 20 26 36 27 28 30 32 34 37 39 41 42 47 52	Unique action, correct grouping

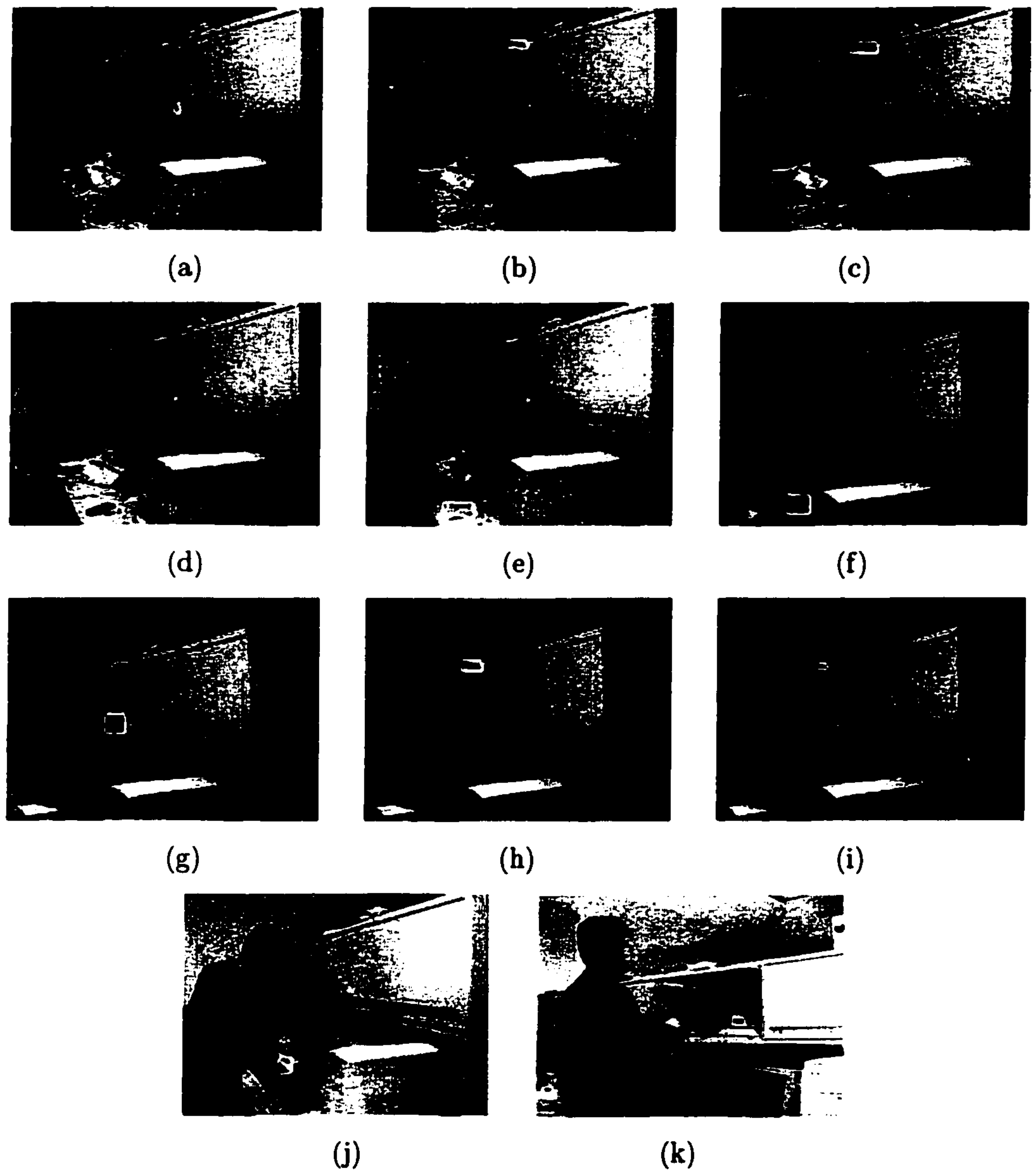


Figure 5.8: Trajectory based change detection results (white bounding box). (a)-(k) are action 2, 9, 10, 11, 12, 19, 22, 24, 25, 35, and 44.

Table 5.9: The performance evaluation for different model based approaches. Each approach was tested with perfect and degenerated noisy data.

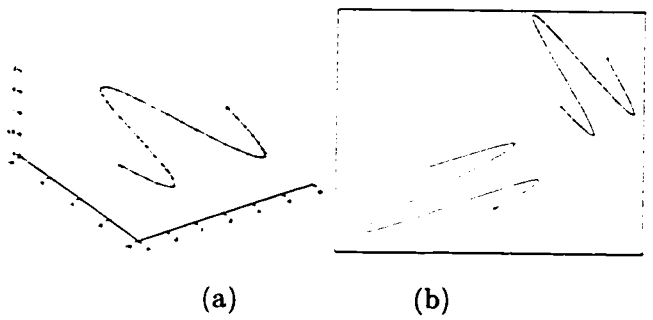
	Perspective camera model with rank constraint similarity	Fundamental matrix based similarity
No noise	Fig.5.10(a): excellent results	Fig.5.10(a): excellent results
With noise	Same as Fig.5.10(a): excellent results	Fig.5.10(b): very bad results

5.2 Examples and Applications of Video Synchronization

We have applied our algorithm on various video sequences. First, we used synthetic trajectory data for an accurate evaluation of the proposed approach (Section 4.4). Next, we applied our method to synchronize the real videos. From Caspi and Irani's experiments [13] we chose sequences acquired by the cameras with non-overlapping FOVs, and the cameras with zoom and no zoom overlapping FOV in order to show the view-invariace of the proposed approach. The alignment of videos, containing human activities captured by moving and stationary cameras, illustrates the robustness of the view-invariant measure used in DTW. The synchronization of the videos of different dancers and matching results can be applied in training dancers. Finally, we applied the algorithm to a long video, containing 60 actions performed by different people, to retrieve automatically similar actions.

5.2.1 Synthetic Examples

We generated a 3D sinusoidal trajectory, and projected it onto 2D plane using different projection matrices. Fig. 5.9 (a) shows the synthetic 3D trajectory, and Fig. 5.9 (b) shows the projected 2D trajectories.



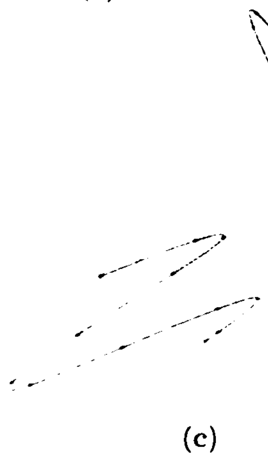


Figure 5.9: (a) A synthetic trajectory in 3D space. (b) The two projected trajectories of (a) in 2D space. (b) The view-invariant dynamic time warping result, where the dotted lines connect the corresponding points.

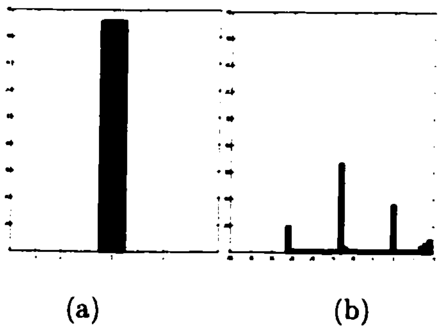


Figure 5.10: (a) The histogram of matching error using the rank constraint employing the perspective camera with/without noise. (b) The histogram of matching error using the fundamental matrix with very small noise in the data.

First, we used the (x, y) coordinates of the trajectories for general DTW algorithm. The DTW using Euclidian distance cannot correctly solve the correspondence at all, since the shapes of two trajectories are significantly different due to the projection effects. Second, we compared the view-invariant measure using the rank constraint and applied view-invariant DTW to obtain correspondences. Fig. 5.9(c)shows the result, here the dotted lines connect the corresponding points in each trajectory. Table 5.9 shows the error under different conditions.

The noise with a normal distributed with $\sigma = 0.00001$ and $mean = 0$ was added to the 2D points in trajectories. Fig. 5.10 shows the histogram of correspondence errors for different methods. In this figure, 0 error represents the correct correspondence result, 1 and -1 represent the forward and backward one frame error in trajectory correspondences, and so on. In other words, the horizontal axis is the error (number of frames) between corresponding frames, and the vertical axis is a total number of frames that have a certain error. There are total 183 points in the sequences. Rank based results are not affected by this small disturbance, however, the fundamental matrix based results degraded dramatically. We used the toolbox provided by Torr to compute the fundamental matrix and applied the linear and non-linear approaches. We can conclude that the rank constraint based approach is much more stable than the fundamental matrix based approach.

5.2.2 Zoomed and Non-overlapping Sequences

In [13], Caspi and Irani propose an attractive method to align two non-overlapping video sequences. Their approach is based on the computation of inter-frame transformations along each video sequence. This approach requires two fixed cameras installed on a common platform. In their experiments, the scene is static, but the video cameras are moving. It is equivalent to the static cameras capturing the dynamic scene. Although the fields of views are non-overlapping, the spatial relationship (epipolar geometry) is still maintained.

We applied our method to sequences used in experiments of Caspi and Irani [13]. The first experiment contains one sequence captured by a camera with a wide FOV and the other captured by a camera with a zoomed FOV. The length of sequences is 300 frames. Fig. 5.11 shows the input frames. We tracked the lower left corner of the blue logo in both sequences to obtain trajectories. After alignment only nine frames had incorrect correspondences. Fig. 5.12 shows the results and the the histogram of matching error.

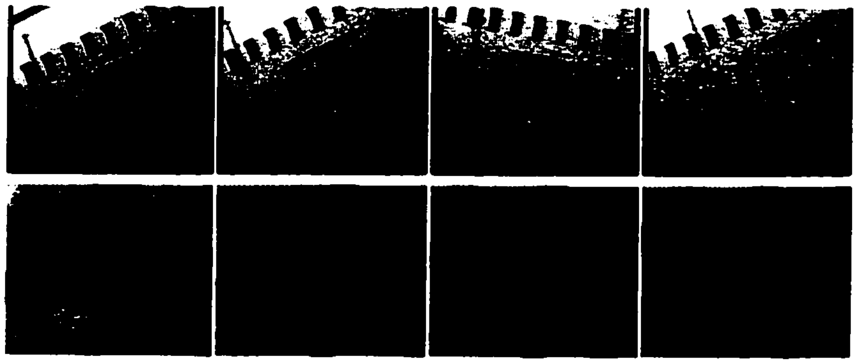


Figure 5.11: The input sequences from Caspi and Irani's paper(frame 1,100,200,299), the first row is a wide field of view scene, and the second row is the zoomed scene.

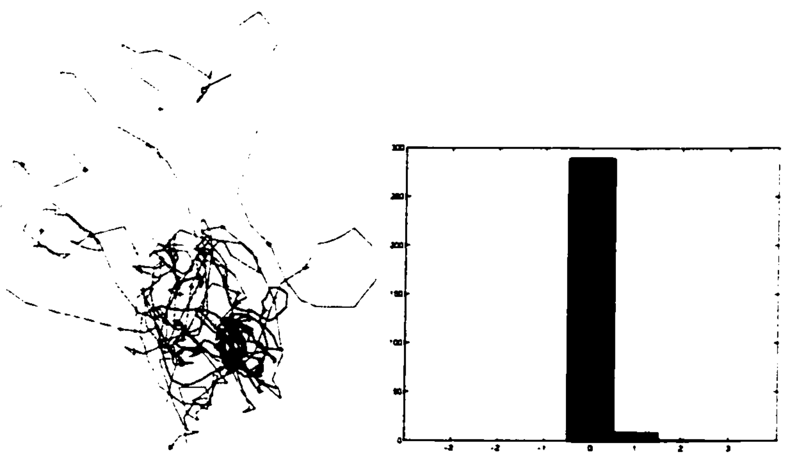


Figure 5.12: The correspondence result for the zoom sequences.

In the second experiment they used videos captured by the moving cameras. Fig. 6 shows the input sequences from the left and the right cameras. There are 80 frames in each video. We tracked the right-upper corner of the gate in the right camera sequence and the left-upper corner of the gate in the left camera sequence. The view-invariant DTW discovered 71 correct correspondences, and eight frames with one frame shift. Fig. 7 shows results of the trajectories and the histogram of matching error.

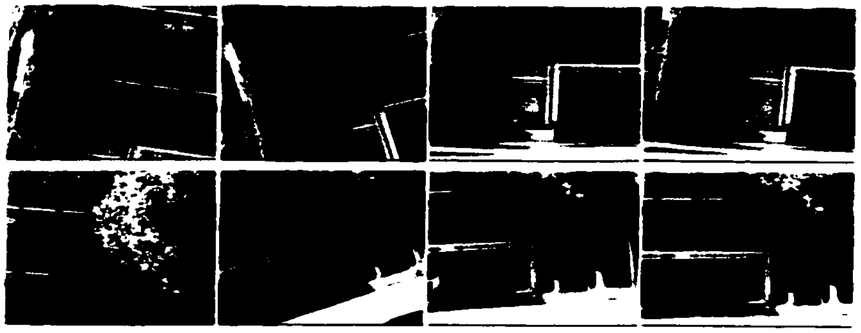


Figure 5.13: The non-overlapping sequences (jump sequence), frame 1,27,54 and 80 are shown. The first row is from the left camera and the second row is from the right camera.

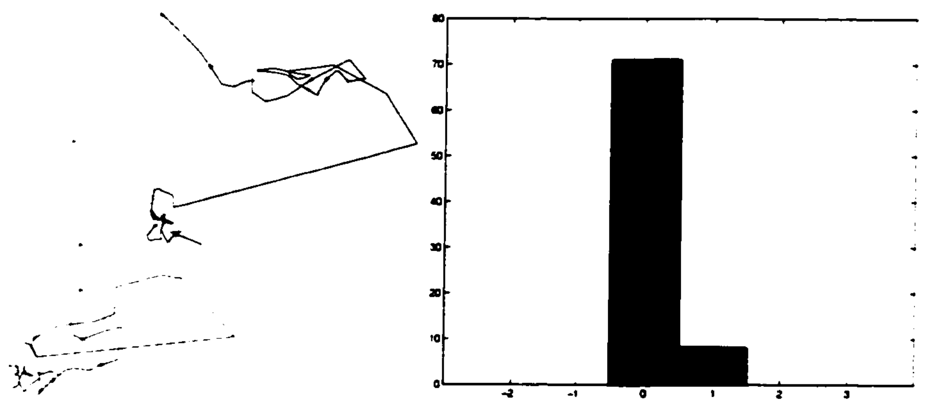


Figure 5.14: The view-invariant DTW correspondence result for jump sequences.

In the third experiment they used non-overlapping sequences. The first half of the videos contains the building around the football stadium. We tracked one feature on the wall of the football stadium and the corner of the window. Fig. 5.15 shows the input images, and Fig. 5.16 shows the results. The view-invariant DTW discovered 151 correct correspondences, 21 frames with one frame shift, and 28 frames with two frames shift. Fig. 8 shows the results of trajectories and the histogram of matching error. The error may due to the tracking error.

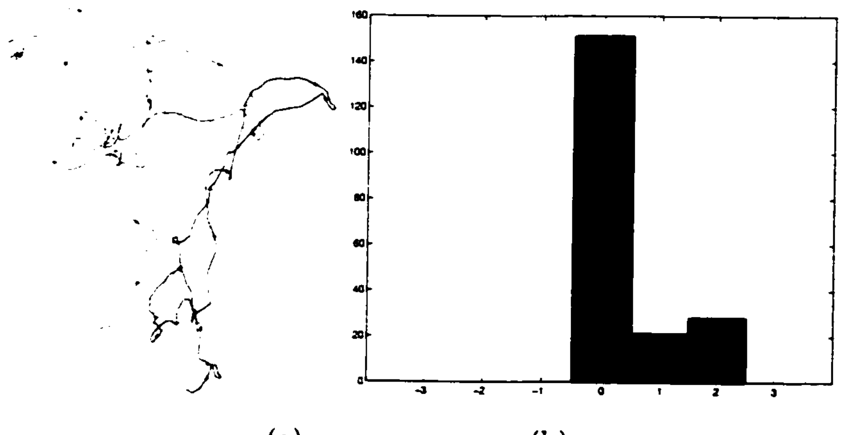
5.2.3 Alignment of Videos Containing Human Activities

From the previous experiments it is hard to evaluate the effectiveness of DTW function. Video sequences were captured simultaneously so the trajectories do not contain the dynamic change among the corresponding frames. Therefore, we performed other experiments at different time and from distinct viewpoints.

In the first experiment, two students were moving their hands up and down with different speeds. We recorded three videos using one camera. The first two videos were captured using static cameras from two different viewpoints, while the third one was captured using a moving camera. The hands were tracked using the mean-shift tracker. We stabilized the frames of the third video (which was captured by the moving camera) by subtracting the image coordinates of a static point (the corner of desk) from the image coordinates of the hand. There was a time-shift of approximately the half cycle in one of the videos relative to the other. We used the perspective camera model in the approach to synchronize these videos. Despite of the changes in the viewpoints and the non-linear time shift, our method successfully established the correspondence between videos. Fig. 5.17 shows the input videos. Fig. 5.18 shows the results of the view-invariant DTW. The results are quite impressive, since the large temporal variation had been compensated.

The next experiment dealt with synchronizing of videos that contain more complicated human activities. We recorded three dancers performing the same movements. For each dancer we captured two video sequences from two significantly distinct view points. Fig.

Figure 5.15: The non-overlapping sequences (football sequence), frames 0,49,99 and 149 are shown. The first row is from the left camera and the second row is from the right camera. There are total over 300 frames in each sequence.



(a) (b)

Figure 5.16: The view-invariant DTW correspondence result for football sequences. (a) shows the two trajectories and the corresponding points connected with dotted lines. (b) The histogram of matching error.

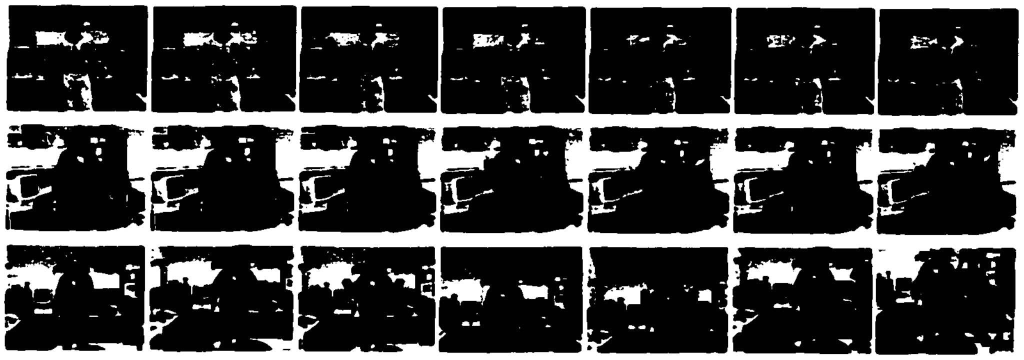


Figure 5.17: The human activity sequences. The first, second and third rows respectively shows the first. second and third input sequences, which are not synchronized. The columns are ordered as frame 0.20.40.60.80,100, and 120 for each sequence.

Figure 5.18: The output of the view invariant dynamic time warping. The columns represent the synchronized corresponding frames. Every 40th of the output frames are shown, they are 11,51,91,131,171,211,251,291.

5.19 shows the trajectories of the left foot of dancers in the six sequences. The difference between trajectories includes viewpoint difference, temporal difference and the difference due to the non-rigid motion of the dancers. We computed the temporal correspondence for each trajectory point with respect to the points in the other five trajectories. So there are total ($C_6^2 = 15$) combinations. Based on the pair-wise correspondence we generated a video containing all six synchronized dance sequences, such that the sequence #1 is warped towards the #6 based on the warping function computed from the trajectories #1 and #6, the sequence #2 is warped towards the sequence #6 also, and so on. From example readers can notice from trajectories #3 and #4 that there is a huge difference between trajectories. Fig. 5.20 shows one of the warping results, in which all sequences are warped toward the sequence #6. Each row contains some key frames in the video, and the corresponding frames are shown in each column. Please reference to the supplemental material to get the full size input/output movies. Although the videos contain large amount of non-rigid motion, our algorithm successfully computed the correspondence among sequences. We are very happy to see that the algorithm runs very robustly and the results are synchronized with a high accuracy.

5.2.4 Computer Aided Training

The time-warping function is a path that minimizes the alignment error at each step through the similarity measure **E**. Each point from the path represents the correspondence between the i^{th} point in trajectory, I, and the j^{th} point in trajectory, J. If many points in the trajectory I correspond to the same point in the trajectory J, then it means that the movement of sequence I is slower than the movement of sequence J at that moment. This observation can help us evaluating performance. We took sequences, #6 as a model and #1 as a test, and computed the warping path between them. Fig. 5.21(a) shows the result. From this figure we can notice that the dancer #1 had a pause at around the frame 150.

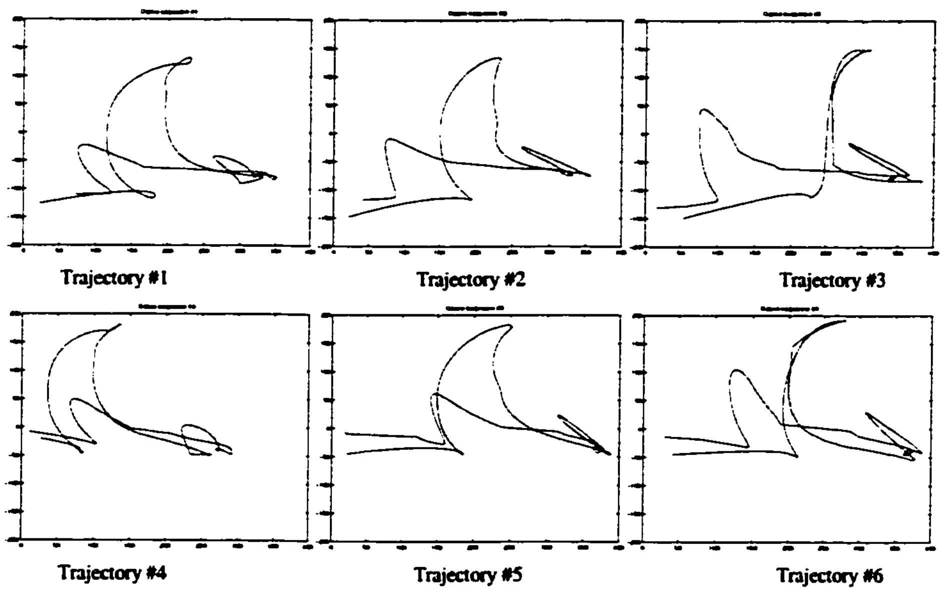


Figure 5.19: The trajectories of the right feet of dancers in 6 sequences. The first row contains trajectories #1, #2 and #3 that correspond to the 1st, 2nd and 3rd dancers respectively. The second row contains trajectories #4, #5 and #6 that correspond to the 1st, 2nd and 3rd dancers respectively also.

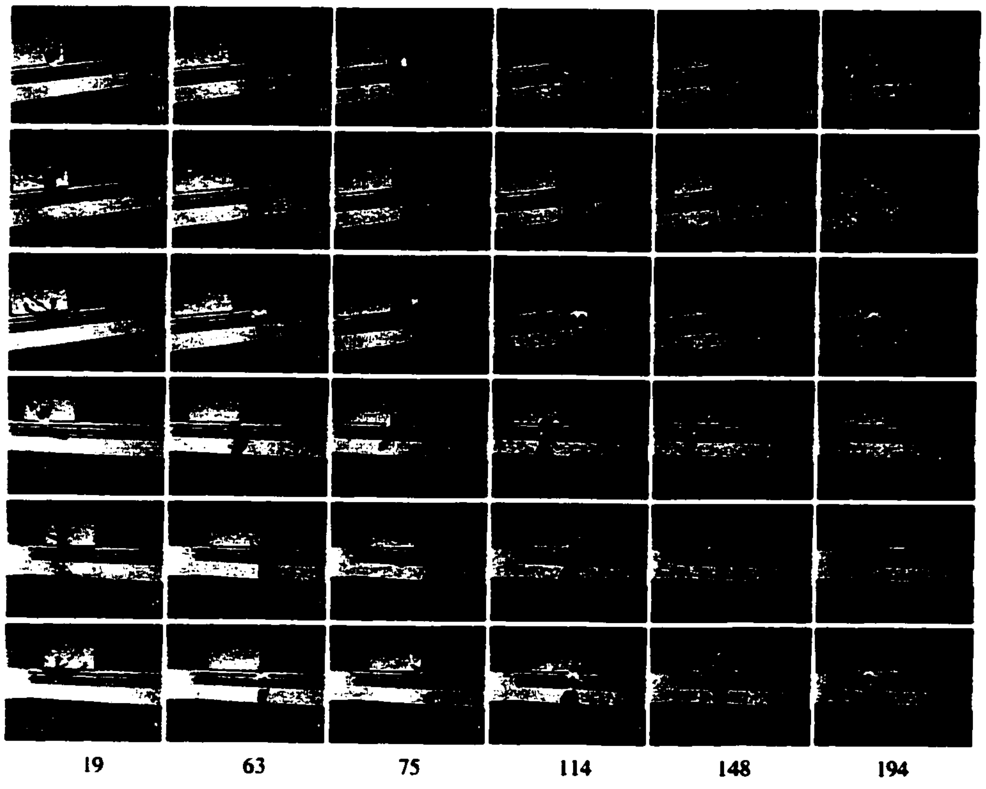
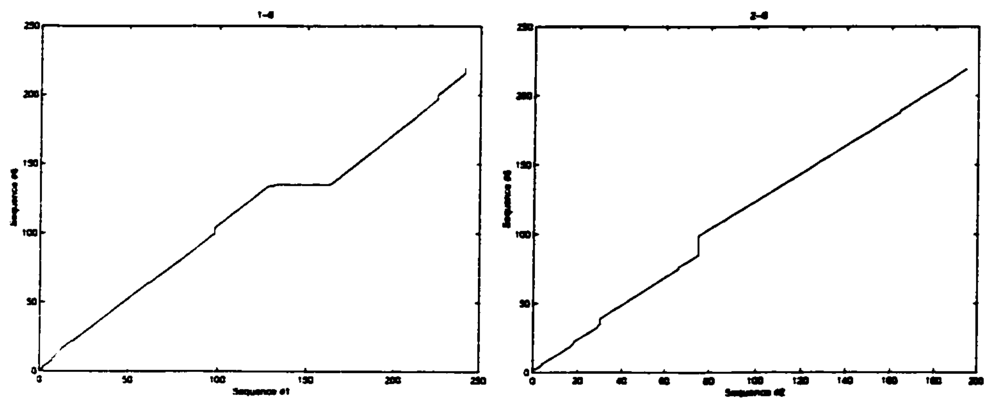


Figure 5.20: The key frames of the output sequences (the frame index is shown at the bottom of figures). The sequences #1, #2, #3, #4, #5 are warped towards the sequence #6 and are shown according to the rows. The 1st and 4th are correspond to the first dancer, the 2nd and 5th correspond to the second dancer, and the 3rd and 6th correspond to the third dancer.



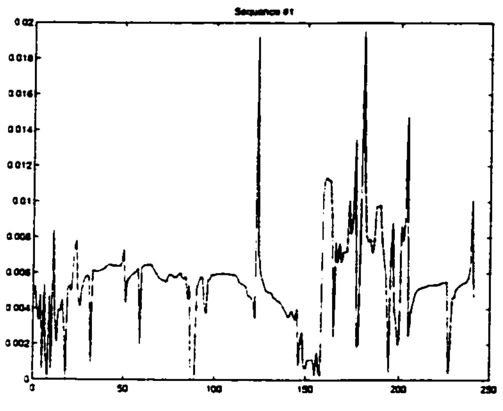
(a) (b)

Figure 5.21: (a) The time-warping path between the sequences #1 and #6, at the frame 150 there is a pause in sequence #1. (b) The time-warping path between the sequences #2 and #6. at the frame 80 the sequence #2 is faster.

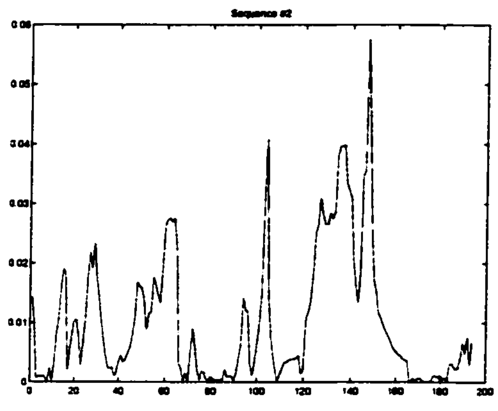
Fig. 5.21(b) shows the time-warping path between sequences #2 and #6. This figure shows the dancer #2 did not decrease the speed at the frame 80. This way, the users can easily find the places for improvement.

Fig. 5.22(a) shows the similarity measure along the time-warping path for sequences #1 and #6. We noticed that the dancer did well overall, but she had a bad movement during frames 150 to 200. We checked the input sequence, and found that she lowered her leg from the upper most position around that time. Therefore, we concluded that she may need to improve that part. Fig.5.22(b) shows the similarity measure for sequences #2 and #6, we detected the dancer #2 had the same problem as the dancer #1.

With the help of view-invariant DTW, we can easily develop a self-training system, such that the users (dancers #1 and #2) record their performance, and compare to the master's (dancer #3). Then the system will give suggestions about the speed and the extent of their movement. Note that the beginner's and master's camera viewpoints can be different. Therefore, this method has a great potential.



(a)



(b)

Figure 5.22: (a) The similarity measurement between the sequences #1 and #6, from frame 150 to 200 are contain large spatial difference. (b) The similarity measurement between the sequences #2 and #6, from frame 120 to 160 are contain large spatial difference.

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we presented a view-invariant representation of human actions. Our representation of the 2-D trajectory of an action is composed of atomic units called *dynamic instants* and *intervals*. The dynamic instants are important motion events, which capture significant changes in the motion trajectory, due to the change in force applied to the object, during an action. This applied force causes a change in the direction and/or speed. We proposed using the spatio-temporal curvature of an action trajectory to detect these dynamic instants. We established this representation to be not only physically meaningful, but also consistent with research in human perception. Moreover, it is view-invariant and as a result simplifies the higher-level task of recognition substantially.

We further demonstrated such higher-level recognition in our system, learning human actions without training. Both spatial and temporal information of trajectories were considered by the spatio-temporal matching algorithm, and it was observed that viewpoint change and variance of execution speed of subjects did not affect the matching results. An unsupervised clustering algorithm was then applied to the matching results to classify the actions into consistent groups. Due to the invariance of our representation and the spatio-temporal matching algorithm, this relatively simple algorithm achieved an excellent recognition rate, over 85% similarity matchings are correct.

In conclusion, our system automatically computes a view-invariant representation for each action, and is able to incrementally learn different instances of actions starting with no model. Our experiments convincingly concluded that the proposed algorithm reliably recognizes human actions.

In addition to action recognition, we generalized the view-invariant dynamic time warping approach using perspective camera model, and applied it to align two videos containing human activities, which were taken from different viewpoints. The approach shows convincing robustness for various applications.

6.2 Future Work

In the current study, only the position and the orientation of the action agent are used for detecting instants. There are other characteristics, such as area, eccentricity and shape, that can be used to get richer information about the movement of agents. Furthermore, only actions that are performed by a single action agent are considered. When we study the actions that are performed by two hands, for instance, the nature of their *interaction* draws our attention more than the individual trajectories - although it is still valid to treat the trajectories from the left hand and the right hand separately. Towards this end, we have done some preliminary research about the timing between the two hands during an action. We observe that there are only two types of interactions: (1) the events happen simultaneously, which means that the both hands are involved in the same event or (2) events occur when one hand is performing, at the same time the other hand remains static or moves unintentionally. Human beings rarely perform two different tasks simultaneously, and as a result it is unlikely that each hand is involved in an independent task. An amusing illustration (that the interested reader may want to test!) is that it is rather difficult for a person to draw a circle with their left hand and simultaneously draw a rectangle with their right hand. This is because our attention cannot focus on two separate things at the same time. The same argument holds for a pianist, who needs extensive training to synchronize his/her hands on the keyboard. As our understanding of human psychology improves, new avenues in human action understanding will unquestionably open. It is our conviction that the proposed system will provide a solid foundation for the inclusion of further knowledge of context, human behavior and psychology, as science progresses.

There are numerous applications can be developed based on the view-invariant dynamic time warping video alignment algorithm, such as a judging system in sports (i.e. figure skating, gymnastics, ballroom dancing, and diving) and training systems (i.e. aerobic, ballet). In those sports, judges are needed to evaluate the performance of an athlete. Our algorithm can give unbiased evaluation in both spatial and temporal domain. The training system can give the feedback to the person who is practicing. Therefore, this approach has great commercial potential. Due to the time limit, we have not experimented the video sequences containing multiple persons, however, the current algorithm can be generalized for those videos.

List of References

[1] H. Alborzi, A. Druin, J. Montemayor, L. Sherman, G. Taxn, J. Best, J. Hammer, A. Kruskal, A. Lal, S. T. Plaisant, L. Sumida, R. Wagner, and J. Hendler. Designing storyrooms: Interactive storytelling spaces for children. In *Proceedings of Designing Interactive Systems (DIS 2000)*, pages 95–104. ACM Press, 2000.

[2] A. Amir, G. Ashour, and S. Srinivasan. Towards automatic real time preparation of on-line video proceedings for conference talks and presentations. In *Thirty-Fourth Hawaii Int. Conf. on System Sciences, HICSS-34*, Maui, Jan. 2001.

[3] D. Ayers and M. Shah. Recognizing human actions in a static room. In *Proc. IEEE Workshop on Applications of Computer Vision, WACV'98*, pages 42–47, 1998.

[4] P. J. Besl and R. C. Jain. Invariant surface characteristics for 3d object recognition in range images. *CVGIP*, 33:33–80, 1986.

[5] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. pages 63–84, 1998.

[6] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. pages 374 –381, June 1995.

[7] A. Bobick, Intille J. D. S., F. Baird, L. Cambell, Y. Irinov, C. Pinhanez, and A. Wilson. Kidsroom: Action recognition in an interactive story environment. Technical report, M.I.T Perceptual Computing, 1996.

[8] A. F. Bobick and Y. Ivanov. Action recognition using probabilistic parsing. In *Proc. of CVPR'98*, pages 196–202, Santa Barbara, CA, 1998.

[9] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. pages 13–15, 2000.

[10] L.W. Campbell, D.A. Becker, A. Azarbayejani, A.F. Bobick, and A. Pentland. Invariant features for 3d gesture recognition. In *Proceedings, International Conference on Automatic Face and Gesture Recognition*, pages 157–162, 1996.

[11] M.A. Casey, W.G. Gardner, and S. Basu. Vision steered beam-forming and transaural rendering for the artificial life interactive video environment (alive). In *99th Convention of the AES*, New York, Oct. 1995.

[12] Y. Caspi and M. Irani. A step towards sequence-to-sequence alignment. pages 682–689, 2000.

[13] Yaron Caspi and Michal Irani. Alignment of Non-Overlapping sequences. In *ICCV'01*, pages 76–83, 2001.

[14] Yaron Caspi, Denis Simakov, and Michal Irani. Feature-based sequence-to-sequence matching. In *VAMODS (Vision and Modelling of Dynamic Scenes) workshop with ECCV*, Copenhagen, 2002.

[15] Ting chuen Pong and Hong jian Zhang. Recent advances in content-based video analysis. *International Journal of Image and Graphics, World Scientific Publishing Company*, 1(3):445–468, 2001.

[16] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 142–149, 2000.

[17] Trevor J. Darrell, Irfan A. Essa, and Alex P. Pentland. Task-specific gesture analysis in real-time using interpolated views. *IEEE Trans. PAMI*, 1995.

[18] J. Davis and A. Bobick. The representation and recognition of action using temporal templates. In *CVPR*, pages 928–934, 1997.

[19] J. Davis, A. Bobick, and W. Richards. Categorical representation and recognition of oscillatory motion patterns. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 628–635, June 2000.

[20] J. Davis and M. Shah. Three-dimensional gesture recognition. In *Proc. of Asilomar Conference on Signals, Systems, And computers*, 1994.

[21] David Demirdjian, Andrew Zisserman, and Radu Horaud. Stereo autocalibration from one plane. In *ECCV*, pages 625–639, 2000.

[22] Jaime Dever, Niels da Vitoria Lobo, and Mubarak Shah. Automatic visual recognition of armed robbery. In *IEEE International Conference on Pattern Recognition*, Canada, 2002.

[23] S. Gibet, A. Braffort, C. Collet, F. Forest, R. Gherbi, and T. Lebourque. Gesture in human-machine communication: capture, analysis-synthesis, recognition, semantics. In *Gesture Workshop'96, Publi dans Gestural Interaction*, York, U.K., 1996. Springer-Verlag.

[24] M. Giese and T. Poggio. Synthesis and recognition of biological motion patterns based on linear superposition of prototypical motion sequences. In *Proceedings of the MVIEW 99 Symposium at CVPR*, pages 73–80, Fort Collins, CO, 1999.

[25] Kristine Gould and Mubarak Shah. The trajectory primal sketch: A multi-scale scheme for representing motion characteristics. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 79–85, San Diego, June 1989.

[26] I. Haritaoglu, D. Harwood, and L. Davis. W4: Real-time surveillance of people and their activities. 22(8):809–830, 2000.

[27] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, September 2000.

[28] Jesse Hoey and James J. Little. Representation and recognition of complex human motion. In *Proceedings of CVPR*, Hilton Head, SC, 2000.

[29] D.C. Hogg. *Interpreting Images of a Known Moving Object.* PhD thesis, University of Sussex, 1984.

[30] Radu Horaud and Gabriella Csurka. Autocalibration and euclidean reconstruction using rigid motion of a stereo rig. In *Proc. of the Sixth International Conference of Computer Vision*, pages 96–103, Bombay, India, 1998.

[31] Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29(1):5–28, 1998.

[32] Omar Javed, Sohaib Khan, Zeeshan Rasheed, and Mubarak Shah. A framework for segmentation of interview videos. In *Proc. of The Eighth IEEE International Conference on Computer Vision*, Vancouver, Canada, July 2001.

[33] Omar Javed, Khurram Shafique, and Mubarak Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *IEEE Workshop on Motion and Video Computing*, Orlando, FL, Dec 2002.

[34] Omar Javed and Mubarak Shah. Tracking and object classification for automated surveillance. In *The seventh European Conference on Computer Vision*, Denmark, 2002.

[35] S. Ju, M. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 38–44, 1996.

[36] Sohaib Khan, Omar Javed, and Mubarak Shah. Tracking in uncalibrated cameras with overlapping field of view. In *Performance Evaluation of Tracking and Surveillance PETS 2001, (with CVPR 2001)*, Kauai, Hawaii, Dec 2001.

[37] R. Kjeldesn and J. Kender. Finding skin in color images. In *Int. workshop on Automatic Face and Gesture Recognition*, pages 312–317, 1996.

[38] D. Koller, D Heinze, and H-H Nagel. Algorithmic characterization of vehicle trajectories from image sequences by motion verbs. In *CVPR'91*, pages 90–95, 1991.

[39] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10(6), Dec 1994.

[40] N. Li, S. Dettmer, and M. Shah. *Motion-Based Recognition*, chapter Visually recognizing speech using eigensequences, pages 345–371. Kluwer Academic, Dordrecht, 1997.

[41] P. Maes, B. Blumburg, T. Darrell, and A. Pentland. The alive system: Full-body interaction with autonomous agents. In *Proceedings of Computer Animation 95*. IEEE Press, Apr. 1995.

[42] A. Mansouri. Region tracking via level set pdes without motion computation. *IEEE Trans. on PAMI*, 24(7):947–961, July 2002.

[43] D. Moore, I. Essa, and M. Hayes. Exploiting human actions and object context for recognition tasks. In *Proceedings of IEEE International Conference on Computer Vision 1999 (ICCV99)*, Corfu, Greece, March 1999.

[44] D. Moore, I. Essa, and M. Hayes. Objectspaces: Context management for action recognition. In *Proceedings of the 2nd Annual Conference on Audio-Visual Biometric Person Authentication*, Washington, March 1999.

[45] Joseph L. Mundy and Andrew Zisserman. *Geometric Invariance in Computer Vision*. The MIT Press, 1992.

[46] Mario E. Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proc. of the 7th Inter. Conf. on Computer Vision, ICCV 99*, pages 108–115, Corfu, Greece, 1999.

[47] D. Newtson and G. Engquist. The perceptual organization of ongoing behavior. *Journal of Experimental Social Psychology*, 12(5):436–450, 1976.

[48] A. Nishikawa, A. Ohnishi, and F. Miyazaki. Description and recognition of human gestures based on the transition of curvature from motion images. In *Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 552–557, 1998.

[49] S. Niyogi and E.H. Adelson. Analyzing and recognizing walking figures in xyt. pages 469–474, 1994.

[50] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. In *Proceedings of ICVS99*, Gran Canaria, Spain, January 1999.

[51] V. Parameswaran and R. Chellappa. Quasi-invariants for human action representation and recognition. In *16th International Conference on Pattern Recognition*, volume 1, pages 307–310, 2002.

[52] D. H Parish, G. Sperling, and M. S. Landy. Intelligent temporal subsampling of american sign language using event boundaries. *Journal of Experimental Psychology*, 16:282–294, 1990.

[53] M. Petkovic and W. Jonker. Content-based video retrieval by integrating spatio-temporal and stochastic recognition of events. In *proceedings of IEEE Intl. Workshop on Detection and Recognition of Events in Video*, Vancouver, Canada, 2001.

[54] R. Polana and R.C. Nelson. Detecting activities. *Jl. of Visual Communication and Image Representation*, 5:172–180, 1994.

[55] F. Quek. Hand gesture interface for human-machine interaction. In *Proc. of Virtual Reality Systems*, volume Fall, 1993.

[56] Krishnan Rangarajan and Mubarak Shah. Establishing motion correspondence. *Computer Vision, Graphics and Image Processing: Image Understanding*, pages 56–73, July 1991.

[57] Cen Rao and Mubarak Shah. View invariance in action recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition, CVPR 2001*, Kauai, Hawaii, Dec. 2001.

[58] Zeeshan Rasheed and Mubarak Shah. Movie genre classification by exploiting audio-visual features. In *Proc. of IEEE International Conference on Pattern Recognition*, Canada, 2002.

[59] Kenneth H Rosen. *Discrete Mathematics and its Applications.* McGraw-Hill Inc., New York, fourth edition, 1999.

[60] J.M. Rubin and W.A. Richards. Boundaries of visual motion. In *Tech. Rep. AIM-835.* Massachusetts Institute of Technology, Apr. 1985.

[61] S. M. Seitz and C. R. Dyer. View-invariant analysis of cyclic motion. *International Journal of Computer Vision*, 25:1–25, 1997.

[62] Larry S. Shapiro, Andrew Zisserman, and Michael Brady. 3d motion recovery via affine epipolar geometry. *Int. J. of Computer Vision*, 16:147–182, 1995.

[63] J. M. Siskind and Q. Moris. A maximum likelihood approach to visual event classification. In *ECCV-96*, pages 347–360, 1996.

[64] F. Sparacino, K. Hall, C. Wren, G. Davenport, and A. Pentland. Improvisational theater space. In *Proc. of the Sixth Biennial Symposium for Arts and Technology*, pages 207–208, New London, Connecticut, March 1997.

[65] T. Starner and A. Pentland. *Motion-Based Recognition*, chapter Real-Time American Sign Language Recognition from Video Using Hidden Markov Models. Computational Imaging and Vision Series. Kluwer Academic Publishers, 1996.

[66] G. P. Stein. Tracking from multiple view points: Self-calibration of space and time. In *DARPA IU Workshop*, pages 521–527, 1998.

[67] Tanveer Syeda-Mahmood, A. Vasilescu, and S. Sethi. Recognizing action events from multiple viewpoints. In *IEEE Workshop on Detection and Recognition of Events in Video (EVENT'01)*, Vancouver, Canada, July 2001.

[68] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *Int. J. of Computer Vision*, 9(2):137–154, 1992.

[69] P. H. S. Torr and A Zisserman. Feature based methods for structure and motion estimation. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *International Workshop on Vision Algorithms*, pages 278–295, 1999.

[70] J.K. Tsotsos and et al. A framework for visual motion understanding. *IEEE PAMI*, 2(6):563–573, Nov. 1980.

[71] M. N. Wallick, N. d. V. Lobo, and M. Shah. A computer vision framework for analyzing computer and overhead projections from within video. *International Journal of Computers and Their Applications - Special Issue on Intelligent Systems*, 8(2):89–100, June 2001.

[72] K. Waters, J. M. Rehg, M. Loughlin, S. B. Kang, and D. Terzopoulo. *Visual Sensing of Humans for Active Public Interfaces*, chapter Visual Sensing of Humans for Active Public Interfaces, pages Computer Vision for Human–Machine Interaction. Cambridge University Press, 1998.

[73] Donna Williams and Mubarak Shah. A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics and Image Processing*, 55(1):14–26, Jan. 1992.

[74] A. Wilson and A. Bobick. Learning visual behavior for gesture analysis. In *IEEE Int'l. Symp. on Comp. Vis.*, Coral Gables, Florida, Nov. 1995.

[75] L. Wolf and A. Zomet. Sequence to sequence self-calibration. In *Proceedings of the European Conference on Computer Vision(ECCV)*, Copenhagen, May 2002.

[76] Y. Yacoob and M. J. Black. Parameterized modeling and recognition of activities. In *ICCV'98*, pages 120–127, Mumbai, India, Jan. 1998.

[77] J. Yamato, J. Ohya, and L. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 624–630, 1995.

[78] J. Yang, Y. Xu, and C.S. Chen. Human action learning via hidden markov model. *IEEE Trans. on System, Man, and Cybernetics*, 27(1):34–44, 1997.

[79] M. Yang and N. Ahuja. Extracting gestural motion trajectories. In *Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 10–15, 1998.

[80] Ming-Hsuan Yang, Narendra Ahuja, and Mark Tabb. Extraction of 2d motion trajectories and its application to hand gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1061–1074, 2002.

[81] K. Yu, J. Mason, and J. Oglesby. Speaker recognition using hidden markov models, dynamic time warping and vector quantisation. In *IEEE Proceedings- Vision, Image and Signal Processing*, volume 142, pages 313–318, Oct. 1995.

[82] J. Zacks and B. Tversky. Event structure in perception and cognition. *Psychological Bulletin*, 127(1):3–21, 2001.

[83] L. Zelnik-Manor and M. Irani. Event-based analysis of video. In *IEEE Conference on Computer Vision and Pattern Recognition*, Dec. 2001.