NOVEL COST MEASURES FOR ROBUST RECOGNITION OF DYNAMIC HAND GESTURES

by

AMEYA KULKARNI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2011

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards supervising professors Dr. Venkat Devarajan and Dr. Vassilis Athitsos, for motivating me on this thesis and helping me out in every step of my work.

I wish to thank Dr. Michael Manry for taking time from his schedule to serve on my thesis committee.

Finally, I would like to express my gratitude towards my family members for their encouragement and support throughout these years.

November 16, 2011

ABSTRACT

NOVEL COST MEASURES FOR ROBUST RECOGNITION

OF DYNAMIC HAND GESTURES

Ameya Kulkarni, M.S.

The University of Texas at Arlington, 2011

Supervising Professors: Venkat Devarajan and Vassilis Athitsos

Computer vision aided automatic hand gesture recognition system plays a vital role in real world human computer interaction applications such as sign language recognition, game controls, virtual reality, intelligent home appliances and assistive robotics. In such systems, when input with a video sequence, the challenging task is to locate the gesturing hand (spatial segmentation) and determine when the gesture starts and ends (temporal segmentation). In this thesis, we use a framework which at its principal has a dynamic space time warping (DSTW) algorithm to simultaneously localize gesturing hand, to find an optimal alignment in time domain between query-model sequences and to compute a matching cost (a measure of how well the query sequence matches with the model sequence) for the query-model pair. Within the context of DSTW, the thesis proposes few novel cost measures to improve the performance of the framework for robust recognition of hand gesture with the help of translation and scale invariant feature vectors extracted at each frame of the input video. The performance of the system is evaluated in a real world scene with cluttered background and in presence of multiple moving skin colored distractors in the background.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Hand Gesture Recognition Issues and Applications

Hand gestures are a natural way to communicate between humans and also for human computer interaction. The hand gestures have a principal advantage of being natural, intuitive and easy to use over other existing computer interfaces. The automatic hand gesture recognition system plays a vital role in real-world applications such as sign language recognition, human computer interaction (HCI), gesture based gaming consoles, virtual reality, intelligent home appliances and assistive robotics. Television control [1], robotic control [2], game consoles [1] and sign language recognition [3] are a few successfully demonstrated applications of hand gesture recognition system. Most importantly, the hand gesture recognition systems empower computers with an ability to understand hand gestures and respond accordingly.

The usability of hand gesture recognition system directly affects its ability to perform in real-world environments. The system should possess a great degree of user friendliness in order to be effectively deployed in real-world environment. The system should be able to communicate with user without any special or cumbersome devices (such as colored markers or gloves [2], full sleeves shirt, wrist bands) and apparatus like remote controls, mouse and keyboard.

The hand gesture recognition system should be robust enough to be able to operate in real-world scenes with cluttered backgrounds. Commonly, the hand gesture recognition systems use skin detection, motion clues, edges and background subtraction to identify correct hand location in each frame. However, it is unreasonable to assume safely that the gesturing hand can be located reliably in each frame under the constraints imposed by the real-world

scenes. The skin detection would fail to unambiguously locate the gesturing hand in presence of skin-colored objects (like face, non-gesturing hand or body part) in the background while the motion clues and background subtraction would fail in presence of moving objects (like people walking) in the background. It is evident from the figure 1.1(a) and figure 1.1(b) that in presence of skin colored and motion distracters the top hand candidate may not be the correct one every time. Even when the skin and motion clues are used collectively, hand detection may not be reliable in real-world scenes (please refer to figure 1.1(a-b)).

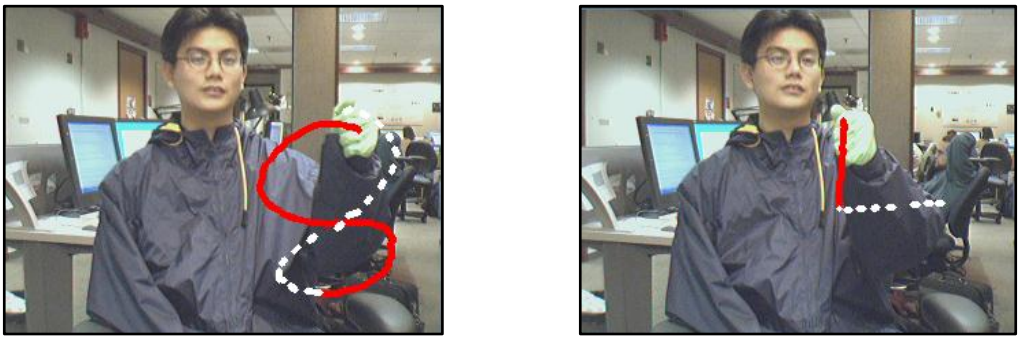
(a) (b)

Figure 1.1 - Illustrates that the low level hand detection can often fail to provide perfect hand detection in presence of moving skin colored distractors. In the examples (a) and (b), the skin color and motion were collectively used to identify hand candidates. The hand candidate marked with white rectangle is the top detection result which is not the correct one.

A common problem when building robust computer vision system is that the higher level modules cannot tolerate inaccuracies of lower level modules. Particularly, in hand gesture recognition systems with bottom-up approach, ambiguities or inaccuracies in lower level hand detection module can propagate into a higher level recognition module consequently degrading overall performance. For a robust hand gesture recognition system it is essential to have a higher level recognition module that tolerates ambiguities of lower level hand detection module.

Translation, rotation and scale invariance are highly desired properties of a computer vision system. With respect to hand gesture recognition systems, as long as the hands are clearly visible, it implies that the gesture could occur in any part of the image frame (translation

invariance), distance between the camera and user does not matter (scale invariance) and the rotations occurring in image plane if any can be tolerated (rotational invariance). This makes the system more flexible plus it allows better degree of freedom to the user performing gestures.



(a) (b)

Figure 1.2 - Illustrates the sub-gesture problem. In example (a) the digit class “5” is similar to digit class “8” and in example (b) the digit class “1” is similar to digit class “4”. Given that the sub-gesture relation is not known, the system is likely to do a false detection.

Further, the hand gestures typically occur in a continuous video sequence, as a result the hand gesture recognition system should detect gestures within a continuous stream. In order to recognize hand gestures continuously the system needs to find on itself the start and end of each gesture. This problem is often referred as the temporal segmentation i.e. task to detect when the gesture starts and ends. While detecting the hand gestures from a continuous stream, it may happen that the gestures (sub-gesture) which appear to the part of another longer gesture (super-gesture) could trigger false detection. This is referred to as the sub-gesture problem. The sub-gesture problem hinders the ability of a hand gesture recognition system by interfering with detection of super-gestures and creating false alarms. The figure 1.2, describes the sub-gesture problem that occurs between the gesture class “5” and gesture class “8” of the palm graffiti digit dataset [4].

1.2 Contributions

In this thesis work, to perform continuous recognition of gesture in a video sequence, we use the Dynamic Space Time Warping algorithm (DSTW) at the core of the framework. The

DSTW algorithm simultaneously localizes gesturing hand (i.e. spatial segmentation), finds an optimal alignment in time domain (i.e. warping path, figure 5.2) and calculates a global matching cost (i.e. a measure of how well a query sequence matches with a model sequence. Please refer to section 5.1 – figure 5.2) between a query-model pair on the basis of a cost measure. In order to perform frame by frame matching in time domain, the dynamic space time warping (DSTW) in its scope defines two cost functions namely the local cost (figure 5.1) and the transition cost (described in section 5.1). The local cost function is a function that can exist between a query and model frame presently being matched. The transition cost function is a function that can exist between present as well as past query and model frames for a pair of query and model frame presently being matched. Conventionally, DSTW implementations use the Euclidean distance between a query and a model frame hand coordinates as the local cost and leave the optional transition cost unimplemented. In this thesis we propose two novel cost measures namely the direction error cost measure and vector error cost measure which when implemented as the local cost or the transition cost or in combination of both, within the scope of DSTW, achieves better system performance (i.e. recognition accuracy and detection rate) in complex real-world environments when using the translation and scale invariant feature vectors.

The first contribution of this thesis is the proposed direction error cost measure. The direction error cost measure when used as the local or the transition cost compares the direction of traversal of the input query sequence and model training sequence at each frame and computes a matching cost in terms of absolute difference in degrees. The second contribution of this thesis is the proposed vector error cost measure. The vector error cost measure when implemented as the local or the transition cost compares a set of vectors traced by input query sequence at each frame to that of the model training sequence and computes a matching cost in terms of Euclidean distance between two vectors. The matching cost returned by these cost measures is compared with a fixed threshold and is used to temporally segment the gestures in the input video sequence and to identify the class of gesture being performed.

Additionally, thesis contributes practically in following two ways: 1) It implements a gesture recognition GUI that helps in comparing the matched trajectories of the query and model sequence 2) It implements a real-time version of the proposed framework.

1.3 Thesis Overview

The remainder of the thesis is organized as follows. The chapter 2 compares and contrasts the various approaches proposed to overcome the challenges in the field of continuous dynamic hand gesture recognition along with the key advantages of the framework used in this work. The chapter 3 describes an overview of the framework. The chapter 4 describes the detection and feature extraction in detail. The chapter 5 explains the core of the framework that is the DSTW algorithm and how it is applied to match a continuous input video with a segmented model sequence. The chapter 6 explains in detail the proposed cost measures, their calculations and their implementation in the DSTW algorithm. The chapter 7 enlists the set of rules implied by the inter-model competition algorithm to temporally segment the gestures in an input video sequence. The chapter 8 tabulates the performed experiments and obtained results followed by the chapter 9 which discusses the outcomes and summarizes conclusions.

CHAPTER 2

RELATED WORK

In many hand gesture recognition systems (e.g. [3], [5]) information flows bottom-up. When input with a video sequence, the lower level analysis module performs the task spatial segmentation (i.e. the task to extracts important parameters such as hand pose, hand shape model and motion features) and/or the temporal segmentation (i.e. the task to determine where the gesture starts and ends). In such bottom-up framework, recognition will fail if the results of hand detection and/or temporal segmentation are incorrect (e.g. [5] – [12])). Despite the advancements in hand detection [13], [14] and tracking [15] – [19], the hand detection can be inaccurate in many real-world settings due to factors like change in environmental brightness, poor image quality, low video resolution, temporary departure of the gesturing hand from the scene and background clutter (such as moving skin colored objects in background).

In this thesis, we use a framework similar to the one proposed by Alon et al. [20]. The framework neither suffers the drawbacks of the bottom-up approach nor does it require spatial and temporal segmentation to be performed as preprocessing. The framework sustains even if, instead of a perfect hand location, a relatively short list of hand candidates encompassing at least one correct hand location is presented at every input frame of the video sequence. Further, within the framework, the information flows in both the bottom-up as well as the top-down direction. In the bottom-up direction, multiple hand candidates are short listed using a lower level hand detection module and their features are fed into a higher level recognition module. In the top-down direction, the higher level recognition module uses information from the training sequences to select an optimal sequence among the exponentially possible many sequences of hand locations. The optimal sequence completes the task of lower level hand detection module by localizing gesturing hand at every input frame. Moreover, the higher level

recognition module also completes the task of temporal segmentation. The framework allows a use of relatively simple and efficient skin and motion based hand detector which can be implemented in a few lines of code.

Some alternative approaches, instead of hand detection, use a set of global image features to recognize hand gestures. For example, motion energy images proposed by Bobick et al. [21], a set of diverse global features (such as thresholded intensity images, difference images, motion history and skin color images) proposed by Dreuw et al. [22], 3D shapes extracted from identified motion areas within a frame proposed by Gorelick et al. [23] and histogram of pair wise distances of edge pixels proposed by Nayak et al. [24]. The key limitation of these approaches is that the approaches are not designed to provide immunity to the noise introduced in presence of moving skin colored distractions in the background. Further, Ke et al [25] propose modeling actions as rigid 3D patterns and extracting action features using 3D extensions of the well-known 2D rectangle filter [26]. But the method was not demonstrated in presence of skin-motion distracters which can severely affect features extracted from the actions. In contrast, the framework has been demonstrated in [20] to work well in presence of the skin-motion distracters.

The framework implements the translation and scale invariance proposed by Stefan et al. [27]. It has been observed that, in front of a camera, if a user moves to the left or to the right, the hands and face move equally to the left or to the right. And when a user moves towards or away from the camera the hands and face are scaled accordingly. The key idea in order to implement translation and scale invariance is to reliably locate the face of the user to extract its center location and size. This technique is based on the assumption that it is easy to detect face reliably than the hands. Mature, publicly-available real-time face detection systems have been available for several years [28], [29]; whereas reliable hand detection remains an active research topic, with state-of-the-art systems being too slow for interactive applications [30]. Based on the center of the face, hand coordinates at each frame are translated so as to make

the center of the face the origin of the new coordinate system and further the coordinates are scaled in order to make diagonal of the face a unit length. Thus the newly calculated hand coordinates are inherently translation and scale invariant. It should be noted that the technique does not address rotational invariance (i.e. rotations occurring in an image plane). We assume that the person performing gesture is sitting and facing the camera in the up-front position, and hence the problem of image plane rotation should not arise.

The framework at its core uses the Dynamic Space Time Warping (DSTW) algorithm proposed by Alon et al. [31] which is an extension of Dynamic Time Warping (DTW) originally designed to recognize spoken words for small vocabulary [32, 33]. The DTW algorithm has been applied successfully to recognize a small vocabulary of gestures [6, 7]. The DTW algorithm essentially aligns two sequences (i.e. the query sequence and model sequence) in time domain and computes a matching cost between the two sequences. This matching cost can further be used to classify a closest matching model sequence to a presented query sequence. More efficient variants of the DTW algorithm are proposed in [34, 35] which improve over the quadratic time complexity of the basic algorithm. The DTW assumes that the feature vector can be reliably extracted at each query frame and thus limits its ability to be used in a situation where multiple hypotheses resulting in multiple feature vectors are extracted at each frame and the correct one is not known. In contrast, DSTW is designed to accommodate multiple hypotheses of hand locations at each query frame.

In multiple hypothesis tracking (e.g., [36]) multiple hypotheses are associated with multiple observations. Each observation corresponds to a different object with a different model. In contrast, the framework selects a single consistent hypothesis among multiple distinct observations (detections), only one of which is correct. The CONDENSATION-based method in principle can be applied to the problem of tracking and gesture recognition [37]. But the method requires large number of hypotheses to be propagated at each frame; which in turn yields slower system performance as a result limiting its usage for real-time systems. Additionally, the

framework used here does not need any knowledge of the observation density and propagation density for each state of each class model whereas it is required in the CONDENSATION technique described in [37].

In the Hidden Markov Model (HMM) framework [38], Sato and Kobayashi extended the Viterbi algorithm to accommodate multiple candidate observations at each query frame; the optimal state sequence is constrained to pass through the most likely candidate at every time step. HMMs have found wider application for problems with large vocabulary (of words or gestures) primarily due to their ability to probabilistically encode the variability of the training data. However, DSTW can still be more suitable in some applications especially because of the fact that it is simple and needs no training. Furthermore, proposed approach differs from [38] in that it incorporates translation and scale invariance, and is evaluated in a more challenging setting (users are wearing short sleeved shirts and at least one to three people moving in the background).

Gesture spotting is a challenging task of recognizing gestures in a continuous stream when the temporal segmentation is not known. Various approaches [39–53] have been proposed in the literature for gesture spotting. The approaches proposed in [39-44] count on perfect hand detection, consequently, limiting their use in presence of moving distracters in background. These approaches can be broadly categorized into two major categories, 1) direct approaches 2) indirect approaches. In direct approaches the temporal segmentation is usually followed by recognition (i.e. task of determining gesture class). The direct approaches first compute low-level motion parameters such as velocity, acceleration, and trajectory curvature (Kang et al. [45]) or mid-level motion parameters such as human activity (Kahol et al. [46]), and then identify gesture boundaries based on an abrupt changes in these parameters. These direct approaches require non-gesturing intervals in between the gestures which cannot be satisfied in case of continuous recognition of digit gestures.

In indirect approaches the temporal segmentation is interweaved with the recognition module, and the gesture boundaries are detected when a recognition likelihood score satisfies a fixed or adaptive threshold condition. Most of the indirect approaches [39, 40, 42] use extended dynamic programming algorithms designed to deal with temporally isolated gestures e.g. Dynamic Time Warping (DTW) [47], [48], Continuous Dynamic Programming (CDP) (Oka [42]), various forms of Hidden Markov Models (HMMs) [10], [17], [54], [40], [49]–[51], and most recently, Conditional Random Fields [52], [53]. The framework used in this work is based on the indirect approach proposed by Alon et al. [20] and unlikely of the other approaches does not require hand to be unambiguously located in each frame of the video sequence.

Once a gesture sequence is spotted (i.e. temporally segmented), various approaches have been proposed to identify the best matching model sequence in order to determine the class of the spotted gesture. These approaches include a set of peak finding rules (Morguet and Lang [41]), spotting rules (Yoon et al. [43]), and the user interaction model (Zhu et al. [44]). The framework used in this work on the other hand uses an intermodal competition algorithm to classify the gesture class. Unlikely the other approaches the inter-model competition algorithm takes a vote amongst the nearest N neighbors to decide the gesture class.

Another practically occurring problem is sub-gesture problem. Many a times a gesture (sub-gesture) which appears to be a part of another longer gesture (super-gesture) can cause false alarms and interfere with the detection of the longer gesture (super-gesture). In order to address this issue the two strategies have been proposed by Lee and Kim [40]. 1) Introduce a maximum length non-gesture pattern 2) Use a procedure to detect user's intention to complete a gesture, such as freezing hand for a while or moving hand out of camera view. The first strategy needs a tuning of special parameter while the second limits the naturalness of the interface. To overcome these limitations we use the sub-gesture reasoning approach proposed by Alon et al [20] except that the sub-gesture relations are manually defined considering small size of the gesture classes in the dataset.

The unified framework approach originally proposed by Alon et al. [20] is the most related to the method described in this work but we differ in the implementation of the framework on following two key points. 1) We add the translation and scale invariance proposed by Stefan et al. [27] to make the framework more robust. 2) We use novel cost measures proposed in this work in the implementation of the DSTW algorithm to improve detection rates of the framework in the translation and scale invariant setup. Since our primary aim is to evaluate performance of the proposed cost measures, we optionally choose not to implement the pruning technique proposed in [20]. Considering the small number of gesture classes, we manually define the sub-gesture relations in contrast to the sub-gesture learning approach proposed in [20]. The proposed cost measures significantly improve the detection accuracy of the framework when tested on a hard digit dataset (described in section 8.1). Further, the inter model competition algorithm proposed in [20] selects only the best gesture candidate at every input frame while we present a more generalized version that uses the voting amongst N nearest neighbors to determine candidate gesture class.

CHAPTER 3

SYSTEM OVERVIEW

In hand gesture recognition applications, the type of gestures used for recognition depends upon the type of application e.g. Sign Language Recognition uses gestures from sign language, TV control application uses up-down, left-right gestures. In our experiments, where primary aim is to design a general HCI system, we use a digit dataset consisting of 10 different classes of gestures corresponding to the numerical digits 0-9 [20]. All the gestures are performed in front of a 2D camera. For each gesture class the dataset contains several training examples (i.e. model sequences). In order to simplify the task of annotating hand locations manually, training examples are captured with users wearing full sleeved shirts and using colored gloves. An automatic annotation module is deployed to extract a single hand location per frame from training examples. Further, the start and end frames of each of the model gesture are manually annotated. As long as the end user recognition system does not use visual aids like colored gloves, full sleeved shirt and known start-end frames; there should be no harm in using colored gloves and other clues to aid an automated hand position annotation module. The digit dataset have two different types of test datasets, namely the “easy” and the “hard” dataset. We use the hard dataset in our experiments. Every hard dataset video sequence has ten different numerical gesture (0-9) performed in a continuous manner. The users are wearing short sleeved shirts and there is at least one person moving in the background. Please refer to the subsection 8.1 for the details and examples of the digit dataset.

For test examples (query sequences), the hand detection module extracts K (up to 25) hand candidates in each frame with the help of skin-motion cues. Hand detection is based on a simple assumption that the gesturing hand is the moving skin-colored object in the input video scene. Skin-motion cues when used collectively lessen the possibility of skin-colored non-

moving and non-skin-colored moving objects being falsely detected as hands. The type of skin-motion based hand detector used is easy to implement and efficient. Every short listed hand candidate is represented by the bounding box of size 40x40.

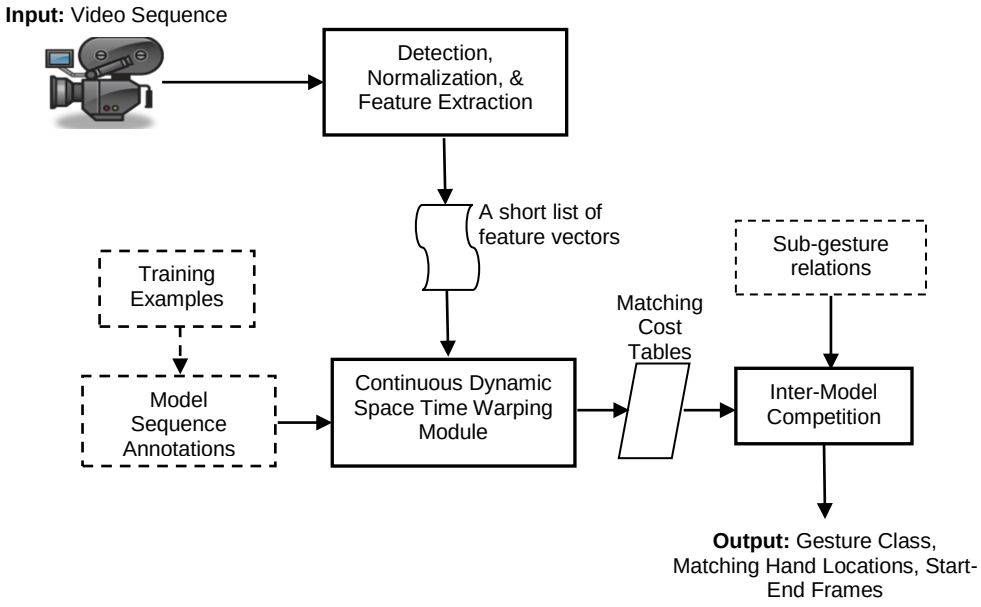


Figure 3.1 – System flowchart. Solid lines indicate flow during continuous recognition. Dotted boxes and arrow represent offline training modules and offline flow respectively.

At the start of each video sequence a face bounding box is extracted using Viola-Jones [29] type face detector. Based on center of face and its size, the hand locations are translated and scaled as described in section 4.3. For each of the translation invariant and scale independent hand location, location features (center *x* and *y* coordinate) are extracted. Thus a 2D feature vector is extracted from every hand candidate. Since we extract K hand candidates per frame, we get K feature vectors per frame.

The feature vectors at every frame are fed to Continuous Dynamic Space Time Warping module. The dynamic space time warping module matches a query video sequence

with model sequences and updates matching cost corresponding to each of the model gesture candidates at every frame.

At every input frame, the inter-model competition module analyzes the matching costs updated by the continuous DSTW module and makes certain observations. Based on the observation of current and/or several previous frames it decides whether a gesture has been performed recently or not. If a gesture was performed, then the module determines the class of the gesture based on nearest neighbor retrieval technique. With the help of manually defined sub-gesture relations and certain rules described in section 7.2, the inter-model competition module resolves sub-gesture recognition problem. E.g. it decides whether the gesture performed by the user belongs to class “5” or the user is in the process of performing gesture that belongs to class “8” (Figure 1.2).

CHAPTER 4

DETECTION AND FEATURE EXTRACTION

4.1 Detection (Query Sequences)

One of the important advantages of DSTW is that it accommodates multiple hypotheses for the hand location in each frame. This property of DSTW makes it easier to design a fairly simple and efficient hand detection module combining mainly two visual cues skin color and motion. The hand detector presented here has been designed with an assumption that the moving skin colored object in a query video sequence corresponds to parts of gesturing hand.

Figure 4.1 – Input video sequence frame

The skin detector, for every pixel in a frame, computes a skin likelihood score. The skin detector uses a Bayesian probabilistic absolute RGB histogram lookup technique learned from samples of skin and non-skin windows similar to one referred in [55]. The probability of a pixel (R, G, B) being skin is given by $P(skin|RGB) = P(RGB|skin) * P(skin) / P(RGB)$ as per the Bay's rule. The $P(RGB)$ is given by the formula $P(RGB) = P(RGB|skin) * P(skin) + P(skin) * P(RGB|non\_skin)$ while, the value of $P(skin)$ has been set to 0.5 based on empirical findings (though it is high, results are good). In this formula the $P(RGB|skin)$ and $P(RGB|non\_skin)$ is learned from the skin and non-skin samples respectively. The figure 4.1 shows an input frame

from a query video sequence while the figure 4.2 shows image with skin likelihood scores. The skin detector output is coarser because of the fact that the a coarser histogram is built in a 32x32x32 color space due to practical limitations instead of the 256x256x256 color space. It should be noted that the skin color detection is efficient as it only requires histogram look up at every pixel.

Figure 4.2 – Skin likelihood score image. The figure shows the skin likelihood score image for the input frame in the figure 4.1.

The motion detector, for every pixel in a frame, computes the motion score based on differencing the current and previous frame. The motion detector first converts 24-bit RGB frames to 32-bit gray-scale images and then computes an absolute difference in intensities at each pixel (i.e. motion score). The motion detector for every input frame n, computes a motion score image, programmatically calculated by following equation $im = abs\big(\, dblgray(frame_n) - dblgray(frame_{n-1})\big)$ where, the operator $dblgray$ computes a 32-bit gray scale image from the input 24-bit RGB frame and the im is the motion score image. The motion detector is efficient as it only requires simple operation like subtraction per pixel. The figure 4.3 shows image with motion scores obtained by taking absolute difference between the frame shown in figure 4.1 and a previous frame.

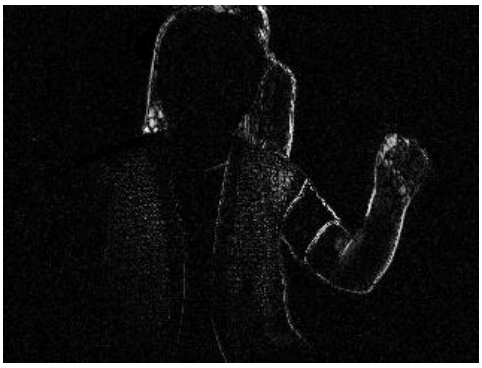


Figure 4.3 – Motion score image. The figure shows image obtained by differencing the input video sequence frame in figure 4.1 and its previous frame.

Figure 4.4 – Skin-motion image. The figure shows image obtained after pixel-wise multiplication of the skin likelihood score image and motion score image for the frame shown in figure 4.1.

Further the skin and motion score at every pixel is multiplied to obtain an image with hand likelihood scores. The figure 4.4 shows a skin-motion image obtained by multiplying the skin and motion colors. This image is filtered with the help of a Gaussian kernel to find the top K sub windows each of size 40x40 at each frame. The figure 4.5 shows Gaussian filtered image and figure 4.6 shows top five hand candidates extracted from the an input frame shown in figure 4.1. We make sure that no two hand candidate sub-windows include the center of each other. By doing this, we eliminate those sub-windows which are centered over same region.

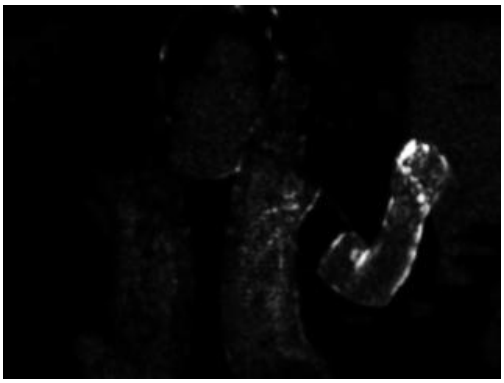


Figure 4.5 – Illustrates a Gaussian filtered image for image shown in figure 4.4.

Another important advantage of the kind of hand detector used here is that it does not use connected component analysis. In related works like [54], the centroid of largest connected component is considered to be the center of hand location. When the gestures are performed by the users wearing short sleeves, a large part arm or overlapping skin colored object can be detected in the connected component and influences the location of centroid, making the hand detector unreliable. In contrast the hand detector presented here generates multiple sub-windows which may occupy different parts of the same connected component. It has been observed that the gesturing hand is typically covered by one or more of these sub-windows (Please refer to figure 4.6).

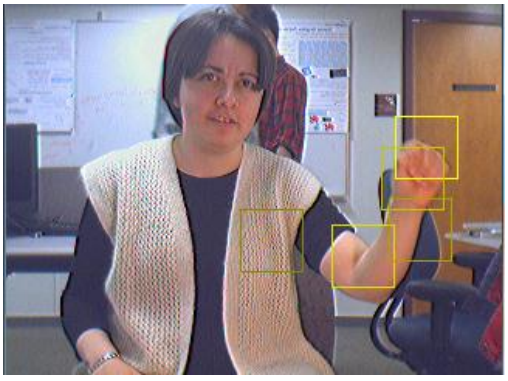


Figure 4.6 – Displays top five hand candidates detected from the input video sequence frame shown in figure 4.1

4.2 Feature Extraction (Query Sequences)

In this way, for every input sequence frame j of the query sequence we find K hand candidates. For every hand candidate k in frame j, a 2D feature vector Q_{jk} is extracted. $Q_{jk} = (x_{jk}, y_{jk})$ and the position (x, y) is the center of the k^{th} sub-window.

4.3 Normalization

Further, in order to achieve translation and scale invariance we extract top face window at the start of each video sequence. The face detector used is a Viola-Jones type rapid object detector [29]. The face detector extracts a single face window corresponding to largest face area. This makes sure that the face closest to camera (which we assume, to be corresponding to the performing user) is picked up. From the face window, we extract center coordinates of the window (cx_j, cy_j) and its diagonal length in pixels l_d . The feature vector is translated such that the (cx_j, cy_j) is the center of the new coordinate system and then scaled to make the face diagonal length l_d correspond to unit length. The normalized feature vector $Q_{jk} = \left(\frac{x_{jk} - cx_j}{l_d}, \frac{y_{jk} - cy_j}{l_d}\right)$ is calculated and fed to the DSTW module.

Figure 4.7 – Hand detection in model sequences with user wearing colored gloves and using full sleeved shirt.

4.4 Detection and Feature Extraction (Model Sequences)

For model sequences in the digit dataset, the user is wearing colored gloves and full sleeved shirts. Hence, we assume that we know the location of gesturing hand and extract only one sub-window at each frame. A feature vector is extracted from this sub-window and then normalized in the same fashion as explained above. The only difference is that we use a different technique to extract hand location. Instead of skin detector we use a specific color detector optimized to work with specific colored glove. At every frame i of model sequence, we extract only one feature vector $M_i = (x_i, y_i)$ and the position (x_i, y_i) represents coordinates of the center of hand sub-window. Use of additional constraints like colored glove is often desirable while for the offline model learning phase as it gives an opportunity to build comprehensive model database and at the same time simplifies the construction of accurate class models. As long as the special gloves or other markers are not used in performing query sequences, they do not affect the naturalness and comfort of the end user. Figure 4.7 shows detected hand candidate in a training sequence frame.

CHAPTER 5

DYNAMIC SPACE TIME WARPING

The Dynamic Space Time Warping algorithm originally described in [31] is an extension of Dynamic Time Warping algorithm (described in [35]) to handle the multiple candidate detection in each frame. The DSTW algorithm performs the important job of spatiotemporal matching (i.e. not only it finds out hand locations at each frame but also align the sequences in time domain). For a pair of query and model sequence to be matched, the DSTW aligns query and model sequence frames, localizes location of the gesturing hand in each query frame and at the same time finds out optimal matching cost between the pair. This query-to-model matching cost is further used to classify the nearest neighbor from the dataset of model sequences. In this section, at first we describe how DSTW can be used to align two temporally segmented video sequences against each other and then we explain how the DSTW can be adapted to the case of continuous video sequence in order to perform the temporal segmentation.

5.1 Matching a Query Sequence to a Model Sequence

This subsection explains how the DSTW matches query and model sequence provided the temporal segmentation is known. Let $M = (M_0, \ldots, M_{m-1})$ be a model sequence of length m frames in which each M_i is a feature vector. Since we assume we know temporal segmentation (i.e. we know where the gesture starts and ends) the query sequence Q consists of known finite feature vectors. Let $Q = (Q_0, \ldots, Q_{n-1})$ be a query sequence of length n frames in which each Q_j is a feature vector. In the DSTW algorithm, each feature vector consists of multiple hypotheses of the hand location, each of those different hypotheses defines a different feature vector. In our experiments the feature vector Q_j is a set of feature vectors $Q_j =$

$\{Q_{j0}, \ldots, Q_{j(K-1)}\}$, where each Q_{jk}, for $k \in \{0, \ldots, K-1\}$, is a candidate feature vector. K is the number of feature vectors extracted in each frame. Though we assume K to be fixed in our experiments, principally it can vary from frame to frame. A warping path W defines an alignment between the model sequence M and a query sequence Q. $W = w_0, \ldots, w_{T-1}$, where, $\max(m, n) \leq T \leq m + n - 1$. Each of $w_t = (i, j, k)$ is a triple which specifies the relationship between model feature vector M_i matched with query feature vector Q_{jk}. The i and j of the w_t represents the temporal dimensions while the k represents spatial dimension. The warping path is generally subjected to several constraints (adapted from [35] to fit the DSTW framework):

1) Boundary conditions: $w_0 = (0,0,k)$ and $w_{T-1} = (m-1, n-1, k')$. This condition requires the warping path to start by matching the first and last frame of the query and model sequence to each other. But there is no restriction on the k and k', which can take any value between 0 to $K-1$.
2) Temporal continuity: Given $w_t = (a, b, k)$ then $w_{t-1} = (a', b', k')$, where $a - a' \leq 1$ and $b - b' \leq 1$. This restricts the allowable steps in the warping path to adjacent cells along the two temporal dimensions.
3) Temporal monotonicity: Given $w_t = (a, b, k)$ then $w_{t-1} = (a', b', k')$, where $a - a' \geq 0$ and $b - b' \geq 0$. This ensures that the warping path sequence increases monotonically in the two temporal dimensions.

Input: A sequence of model feature vectors $M_i, 0 \leq i \leq m-1$ and a sequence of sets of query feature vectors $Q_j = \{Q_{j0}, \ldots, Q_{jK}\}, 0 \leq j \leq n-1$.

Output: A global matching cost $mcost$ and an optimal warping path $W^* = (w_0^*, \ldots, w_{T-1}^*)$

Initialization:
$i = 0$ & $j = 0$
for $k = 0{:}K-1$ ***do***
  $gcost(i, j, k) = initcost(i, j, k)$
end

$j = 0$
for $i = 1{:}\, m-1$ **do**
  for $k = 0{:}\, K-1$ **do**
    $gcost(i,j,k) = gcost(i-1,j,k) + initcost(i,j,k)$
  end
end

$i = 0$
for $j = 1{:}\, n-1$ **do**
  $k^{\wedge} = argmin_k\{gcost(i,j-1,k)\}$
  $w^{\wedge} = (i,j-1,k^{\wedge})$
  for $k = 0{:}\, K-1$ **do**
    $gcost(i,j,k) = gcost(w^{\wedge}) + initcost(i,j,k)$
  end
end

Iteration:
for $j = 1{:}\, n-1$ **do**
  for $i = 1{:}\, m-1$ **do**
    for $k = 0{:}\, K-1$ **do**
      $w = (i,j,k)$
      for $w' \in N(w)$ **do**
        $cumcost(w',w) = tcost(w',w) + gcost(w')$
      end
      $gcost(w) = min_{w' \in N(w)}\, cumcost(w',w) + lcost(w)$
      $b(w) = argmin_{w' \in N(w)}\, cumcost(w',w)$
    end
  end
end

Termination:
$k^* = argmin_k\{gcost(m-1,n-1,k)\}$
$mcost = gcost(m-1,n-1,k^*)$
$w_T^* = (m-1,n-1,k^*)$

Backtrack:
$w_{t-1}^* = b(w_t^*)$

The continuity and monotonicity is required in temporal dimensions but there exists no such restriction for the spatial dimension; the warping path can jump from any of the spatial candidates k' to any other spatial candidate k . The transition cost $(tcost)$ is used to evaluate the cost that can be incurred by a warping path while transitioning from candidate k' at frame $j-1$ to candidate k at frame j. The set $N(i,j,k)$ is a set of all possible values of w_{t-1} that satisfies the warping path constraint $N(w_t) = N(i,j,k) = (\{(i,j-1),(i-1,j-1)\} \times \{0,\dots K-1\}) \cup \{i-1,j,k\}$. In addition to the transition cost a local cost $lcost$ and an initialization cost $initcost$ are used to evaluate the similarities that exist between the model feature vector M_i and query feature vector Q_{jk}. In this thesis we have proposed a few new cost measures and implemented these cost measures as the transition cost or the local cost or in combination of both. In all experiments the initialization cost is the Euclidean distance between vector M_i and Q_{jk}. In an experiments involving direction error cost being used as the transition cost the initialization cost is suitably scaled when adding to direction error cost. The chapter 6 details the proposed cost measures and their implementation along with the calculations. The matching cost $mcost$ for the warping $W^* = (w_0^*), \dots, (w_{T-1}^*)$ can be found out as $mcost(W^*) = \sum_{t=0}^{T-1} gcost(w_t^*)$. The matching cost is a measure of how well the query sequence matches the model sequence. For dataset with multiple gesture classes, the matching cost between the query sequence and each class model can be compared to decide which model provides the closest match. The warping W^* further provides a candidate hand location at every query frame that optimizes the matching between the query sequence and the model.

5.2 Matching a Query Video to a Model Sequence

This subsection explains how the query-model sequence matching as described in subsection 5.1 can be extended to the case of continuous video sequence where the temporal segmentation is not known (i.e. unknown start and end frames). Let $V = (V_1, V_2 \dots.)$ be the video sequence frames where we need to detect gestures. We call V as a query sequence, and from

each of the frames V_j we extract a set Q_j of K feature vectors. $Q_j = \{Q_{j1}, \ldots, Q_{jK}\}$. Each of these feature vectors correspond to the hand candidate region extracted at each frame. Let $M = (M_1, \ldots, M_m)$ be a model sequence of length m frames in which each M_i is a feature vector. To this model sequence we add a state M_0 such that local cost between model feature vector M_0 and query feature vector Q_{jk} is zero (i.e. $lcost(M_0, Q_{jk})$ = 0. Now the boundary conditions require warping path to start by matching the dummy model state M_0 with the first frame V_1 of the query (i.e. $w_1 = (0,1,k)$) and by matching the end frame of the query with end frame of the model (i.e. $w_T = (m, n, k')$). The matching cost $mcost$ of the warping path $W = (i_1, j_1, k_1), \ldots, (i_T, j_T, k_T)$ is defined as $mcost(W) = \sum_{t=1}^{T} min_{w \in A(w_t)}\{cumcost(w)\} + gcost(w_t)$ where $A(w_t) \equiv A(i,j,k) = (\{(i, j-1), (i-1, j-1)\} \times \{0, \ldots K-1\}) \cup \{i-1, j, k\}$.

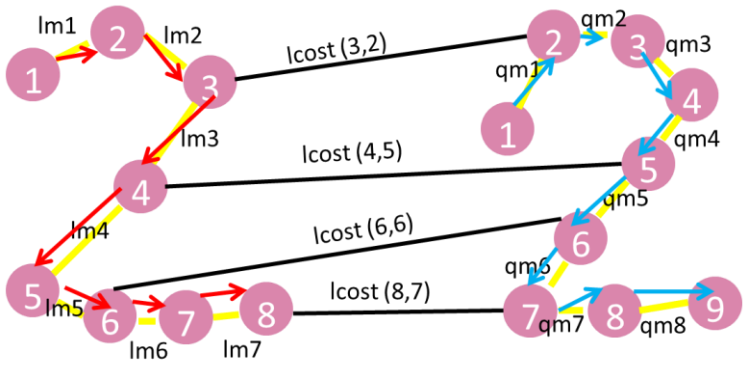


Figure 5.1 – Visualizes local and transition cost functions. The figure shows the local cost function $lcost(M_i, Q_{jk})$ that exists between a model sequence state M_i and a query sequence state Q_{jk}. For the basic method described the local cost function is the Euclidean distance between hand co-ordinates of the model and query sequence state.

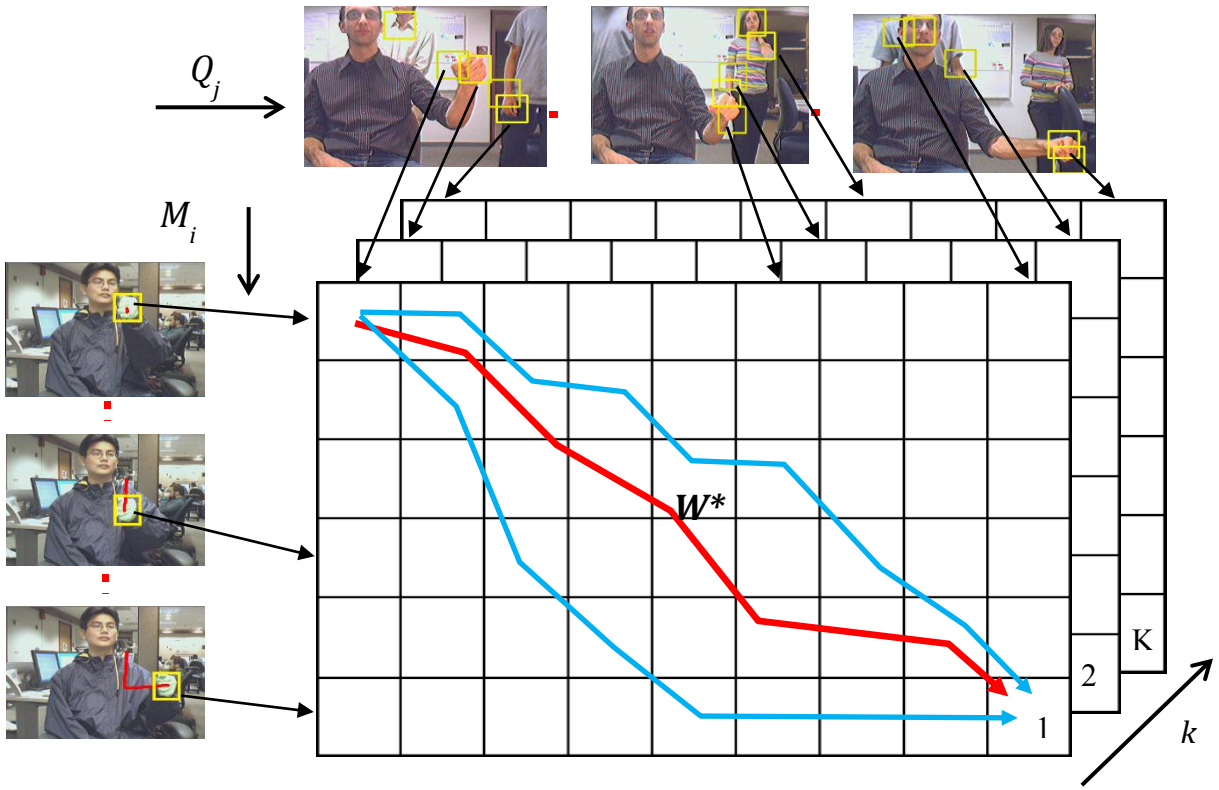


Figure 5.2 Graphical representation of the dynamic space time warping algorithm and warping path. The model sequence M is being matched with a query sequence Q. The blue arrows shown in the figure shows various warping Paths. The red arrow shown in the figure corresponds to the optimal warping path W^* for the query and model pair being matched. The i and j are the temporal dimension and the k corresponds to the spatial dimension. The global matching cost $mcost$ is the matching cost of the optimal warping path.

For solving the above recursion the dynamic programming can be implemented effectively using a 3D table, indexed by (i, j, k). Every time the system observes new input frame V_n , the dynamic programming algorithm needs to compute a 2D part of the table that corresponds to frame n, storing at every cell (i, n, k) the $cumcost(i, n, , k)$, for $i = 0, \dots, m$ & $k = 1, \dots, K$ and a predecessor $A(i^*, n^*, k^*)$ which correspond to the minimum cost among all $cumcost(i, n, , k)$. The predecessor can be used to back track the warping path $W^* = (w_1^*), \dots, (w_T^*)$.

Since we do not know where the gesture ends, we make an assumption that the gesture ends at current input frame n and thus at every input frame n we evaluate whether the gesture ends or not. The conditions for temporal continuity and monotonicity remain unchanged. The algorithm remains essentially the same except we now add an intermodal competition module, which, based on matching cost $mcost$ makes a decision if the gesture ends at the current frame n or not. This inter-model competition module is explained in chapter 7 of the thesis. Additionally, in the case of continuous video sequence apart from important information such as matching cost and optimal hand candidate at each input sequence frame the warping path (under the hypothesis that a gesture has been just performed) specifies optimal temporal segmentation of the gesture.

CHAPTER 6

NOVEL COST MEASURES

The DSTW algorithm essentially finds out the warping path between the given input query sequence and known model sequence. The optimal warping path specifies important information such as the matching cost between the query and model sequence, optimal spatial segmentation (i.e. optimized hand candidates at each frame) and optimal temporal segmentation. The optimal warping path returned by DSTW algorithm depends upon the cost measure used while matching query and model feature vectors. The unified framework proposed in [20] uses Mahalanobis distance between the model state M_i and feature vector Q_{jk} as the local cost. The implementation in work by [27] uses sum of Euclidean distance of position and motion vectors between the model state M_i and feature vector Q_{jk} as the local cost while simply ignores transition cost. In this thesis work, we propose two novel cost measures namely the direction error measure and vector error measure which can be implemented as the local or transition cost within the scope of DSTW and evaluate their performance separately. The proposed cost measures have demonstrated improved recognition accuracies when used with translation and scale invariant (i.e. normalized) feature vectors when compared to the basic Euclidean cost measure. It should be noted that the direction error and vector error cost measures need the knowledge of only one past query and model sequence state.

6.1 Direction Error Cost Measure

The direction error method compares the direction traced by past and current, model and query sequence states. The direction error cost measure can be implemented as the local as well as the transition cost measure. When the direction traced by first and current model sequence state is compared against the direction traced by first and current query sequence state, the direction error can be implemented as a local cost function. We call this type of

implementation as *first-to-current frame direction error as local cost* explained in subsection 6.1.1. When the direction traced by immediate previous and current model sequence state is compared with the direction traced by immediate previous and current query sequence state, the direction error needs to be implemented as the transition cost function. We call this type of implementation as *previous-to-current frame direction error as transition cost* explained in detail in subsection 6.1.2.

6.1.1 First-to-Current Frame Direction Error as Local Cost

The error between the direction traced by the first model sequence state $M_1 = \{fmx, fmy\}$ and current model state $M_i = \{mx_i, my_i\}$ and the direction traced by first query sequence state $Q_{1f} = \{fqx, fqy\}$ and current query sequence state $Q_{jk} = \{qx_{jk}, qy_{jk}\}$ can be calculated as follows:

$$dir\_err = lcost(fmx, fmy, mx_i, my_i, fqx, fqy, qx_{jk}, qy_{jk})$$

$$= \frac{180}{\pi} * abs\left(\tan^{-1}\left(\frac{my-fmy}{mx-fmx}\right) - \tan^{-1}\left(\frac{qy-fqy}{qx-fqx}\right)\right)$$

$$= 360^0 - \left[\frac{180}{\pi} * abs\left(\tan^{-1}\left(\frac{my-fmy}{mx-fmx}\right) - \tan^{-1}\left(\frac{qy-fqy}{qx-fqx}\right)\right)\right] \ldots\ldots\ldots \{if\ dir\_err > 180\}$$

It should be noted that the first query sequence feature vector (Q_{1f}) has been selected to minimize the Euclidean distance between query feature vectors $Q_{1(0:K-1)}$ and model feature vector M_1. The direction error method returns the error cost in degrees and in order to match this cost the initialization cost it scaled appropriately. While calculating the direction error cost in degrees we ensure the cyclical nature of angle.

The figure 6.1 depicts how the direction error technique when used as the local cost matches the query and model sequence states. The figure on the left hand side shows the model sequence states corresponding to each of the model sequence frames while the figure on the right shows query sequence states to be matched. The arrows shown in red color corresponds to the direction traced by the current and first model sequence frames while the arrows in blue corresponds to the direction traced by the current and first query sequence frames. The local

cost between a model state $M_{i=1:8}$ and a query state $Q_{j=1:9}$ can simply be calculated by taking the absolute difference between the directions traced by the model sequence state M_i and the query sequence state Q_j. E.g. the direction error between the model sequence state M_3 and query sequence state Q_4 is simply the error between direction traced by M_3 and Q_4 (i.e. $lcost(M_3, Q_4) = abs(direction(M_1 \rightarrow M_3) - direction(Q_1 \rightarrow Q_4))$.

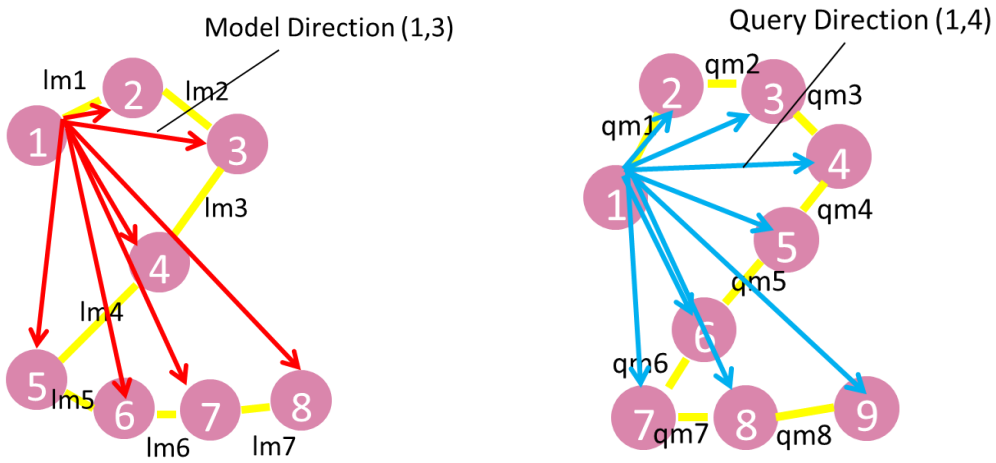


Figure 6.1 – Shows the model sequence states (on left) being matched with the query sequence states (on right) for the direction error cost measure used as the local cost.

6.1.2 Previous-to-Current Frame Direction Error as Transition Cost

The error between the direction traced by the previous model sequence state $M_{i-1} = \{pmx, pmy\}$ and current model state $M_i = \{mx_i, my_i\}$ and the direction traced by previous query sequence state $Q_{(j-1)k*} = \{pqx_{k*}, pqy_{k*}\}$ and current query sequence state $Q_{jk} = \{qx_{jk}, qy_{jk}\}$ can be calculated as follows:

$$dir\_err = tcost(pmx, pmy, mx_i, my_i, pqx_{k*}, pqy_{k*}, qx_{jk}, qy_{jk})$$

$$= \frac{180}{\pi} * abs\left(\tan^{-1}\left(\frac{my-pmy}{mx-pmx}\right) - \tan^{-1}\left(\frac{qy-pqy}{qx-pqx}\right)\right)$$

$$= 360^0 - \left[\frac{180}{\pi} * abs\left(\tan^{-1}\left(\frac{my-pmy}{mx-pmx}\right) - \tan^{-1}\left(\frac{qy-pqy}{qx-pqx}\right)\right)\right] \ldots\ldots\ldots \{if\ dir\_err > 180\}$$

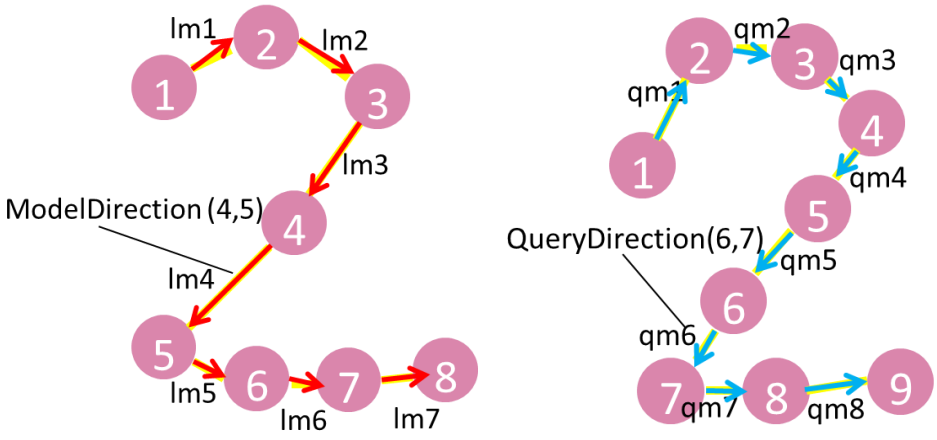


Figure 6.2 – Shows the model sequence states (on left) being matched with the query sequence states (on right) for the direction error cost measure used as the transition cost.

The figure 6.2 depicts the how the direction error technique when used as the transition cost matches the query and model sequences. The figure on the left hand side shows the model sequence states corresponding to each of the model sequence frames while the figure on the right shows query sequence states to be matched. The arrows shown in red color corresponds to the direction traced by the previous and current model sequence frames while the arrows in blue corresponds to the direction traced by the previous and current query sequence frames. The transition cost between a model state $M_{i=1:8}$ and a query state $Q_{j=1:9}$ can simply be calculated by taking the absolute difference between the directions traced by the model sequence state M_i & M_{i-1} and the query sequence state Q_j & Q_{j-1}. E.g. the direction error between the model sequence state M_5 and query sequence state Q_7 is calculated as following: $tcost(M_5, Q_7) = abs(Direction(M_4 \rightarrow M_5) - direction(Q_6 \rightarrow Q_7))$.

6.2 Vector Error Cost Measure

The vector error method finds out the error between the vector traced by the past and current model sequence state and query sequence state. The vector error cost measure can be

implemented as the local as well as the transition cost measure. When the vector traced by the first and current model sequence state is compared against the vector traced by first and current query sequence feature vector, the vector error can be implemented as a local cost function. We call this type of implementation as *first-to-current frame vector error as local cost* and is explained in subsection 6.2.1. When the vector traced by the immediate previous and current model sequence state is compared against the vector traced by immediate previous and current query sequence feature vector, the vector error can be implemented as a transition cost function. We call this type of implementation as *previous-to-current frame vector error as transition cost* and is explained in subsection 6.2.2.

6.2.1 First-to-Current Frame Vector Error as Local Cost

The error between the vector traced by the first model sequence state $M_1 = \{fmx, fmy\}$ and current model state $M_i = \{mx_i, my_i\}$ and the vector traced by first query sequence state $Q_{1f} = \{fqx, fqy\}$ and current query sequence state $Q_{jk} = \{qx_{jk}, qy_{jk}\}$ can be calculated as follows:

$$lcost\left(fmx, fmy, mx_i, my_i, fqx, fqy, qx_{jk}, qy_{jk}\right)$$

$$= \sqrt{\left((mx_i - fmx) - (qx_{jk} - fqx) \right)^2 + \left((my_i - fmy) - (qy_{jk} - fqy) \right)^2}$$

It should be noted that the first query sequence feature vector (Q_{1f}) has been selected to minimize the Euclidean distance between query feature vectors $Q_{1(0:K-1)}$ and model feature vector M_1.

The figure 6.3 depicts the how the vector error technique when used as the local cost matches the query and model sequences. The figure on the left hand side shows the model sequence states corresponding to each of the model sequence frames while the figure on the right shows query sequence states to be matched. The arrows shown in red color corresponds to the vector traced by the current and first model sequence frames while the arrows in blue corresponds to the vector traced by the current and first query sequence frames. The local cost

between a model state $M_{i=1:8}$ and a query state $Q_{j=1:9}$ can simply be calculated by finding the Euclidean distance between the vectors traced by the model sequence state M_i and the query sequence state Q_j. E.g. the vector error between the model sequence state M_3 and query sequence state Q_4 is calculated as $lcost(M_3, Q_4) = vector\_err(vector(M_1 \rightarrow M_3), vector(Q_1 \rightarrow Q_4))$.

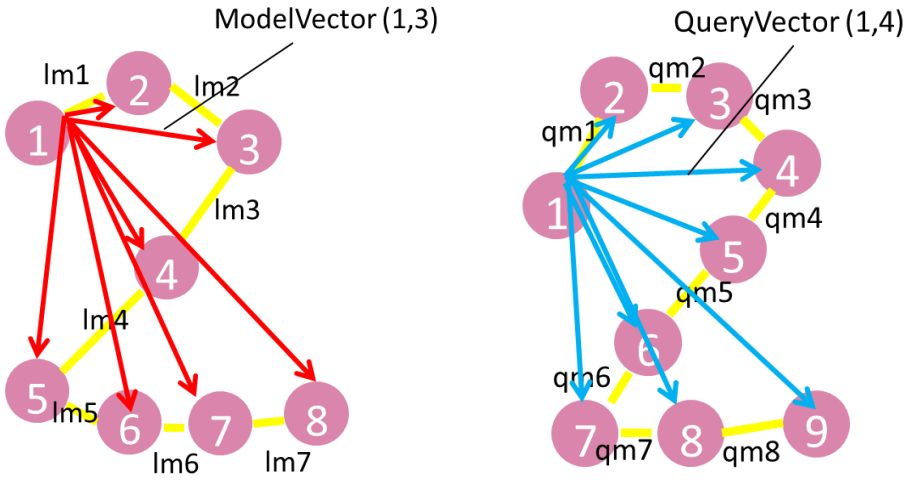


Figure 6.3 – Shows the model sequence states (on left) being matched with the query sequence states (on right) for the vector error cost measure used as the local cost.

6.2.2 Previous-to-Current Frame Vector Error as Transition Cost

The error between the vector traced by the previous model sequence state $M_{i-1} = \{pmx, pmy\}$ and current model state $M_i = \{mx_i, my_i\}$ and the vector traced by previous query sequence state $Q_{(j-1)k*} = \{pqx_{k*}, pqy_{k*}\}$ and current query sequence state $Q_{jk} = \{qx_{jk}, qy_{jk}\}$ can be calculated as follows:

$$tcost(pmx, pmy, mx_i, my_i, pqx_{k*}, pqy_{k*}, qx_{jk}, qy_{jk})$$

$$= \sqrt{\left((mx_i - mx_{i-1}) - (qx_{jk} - pqx_{k*}) \right)^2 + \left((my_i - my_{i-1}) - (qy_{jk} - pqy_{k*}) \right)^2}$$

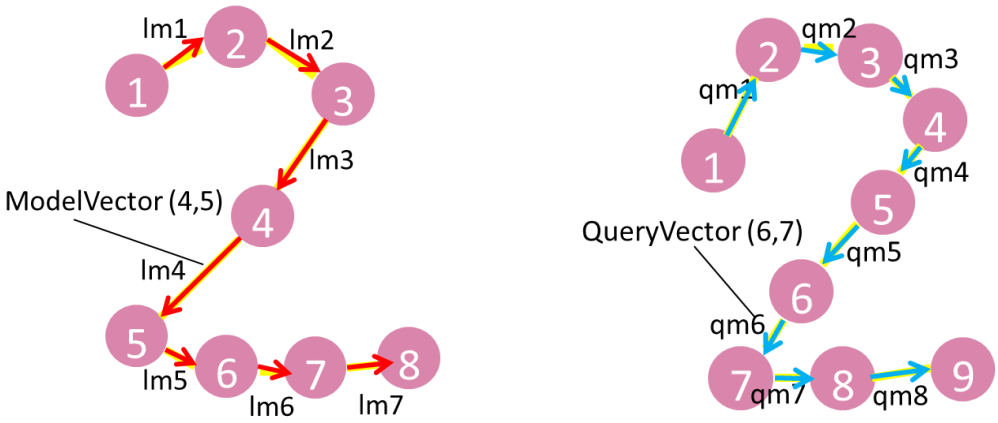


Figure 6.4 – Shows the model sequence states (on left) being matched with the query sequence states (on right) for the vector error cost measure used as the transition cost.

The figure 6.4 depicts the how the vector error technique when used as the transition cost matches the query and model sequence states. The figure on the left hand side shows the model sequence states corresponding to each of the model sequence frames while the figure on the right shows query sequence states to be matched. The arrows shown in red color corresponds to the vectors between the previous and current model sequence frames while the arrows in blue corresponds to the vectors between the previous and current query sequence frames. The transition cost between a model state $M_{i=1:8}$ and a query state $Q_{j=1:9}$ can simply be calculated by taking the Euclidean distance between the vectors between the model sequence state M_i & M_{i-1} and the query sequence state Q_j & Q_{j-1}. E.g. the vector error between the model sequence state M_5 and query sequence state Q_7 is calculated as following: $tcost(M_5, Q_7)$ = $vector\_err(vector(M_4 \rightarrow M_5) - vector(Q_6 \rightarrow Q_7))$.

CHAPTER 7

INTER-MODEL COMPETITION AND SUB-GESTURE RELATIONS

The inter-model competition algorithm decides whether a gesture has been just performed or not. In order to determine if a gesture has been performed or not, the inter-model competition algorithm compares the matching cost returned by the model gestures at each frame. At this stage the knowledge of sub-gesture relations is required. Sub-gesture relations play an active part in deciding if a gesture has ended and in determining the gesture class of the gesture. The sub-gesture relations solves the sub-gesture problem discussed in section 1.1 (figure 1.2).

7.1 Sub-gesture Relations

The inter-model competition algorithm requires knowledge of the sub-gesture relationships between the gestures. The sub-gesture relationships can be manually specified typically when sub-gesture relations are obvious and number of gesture classes is less. However, if the sub-gesture relations are less obvious, an automated learning method described in [56] can be used. A gesture h_1 can be a sub-gesture of another gesture h_2 if h_1 is similar to a part of h_2 under the similarity model of the matching algorithm. Figure 1.2 explains the sub-gesture problem occurring between the digit gesture “5” and digit gesture “8”. The manually defined sub-gesture relations for the digit dataset are tabulated in table 8.3.

7.2 Inter-model Competition

The inter-model competition algorithm at every input frame, decides if a gesture ends at the current input frame n or not. If a gesture ends at current input frame n then the inter-model competition algorithm finds out the closest matching model gesture candidate. At every input frame the inter-model competition algorithm is invoked once. When invoked, the inter-model competition algorithm firstly updates a list of selected model candidates at every frame and

based on certain set of rules, it decides from the list (of selected model candidates), which of the model candidate gesture actually corresponds to the gesture performed by the user in the input video sequence.

Before the first input video sequence frame is observed a list i_1 of selected model candidates is initialized to be empty. At every input frame, the continuous DSTW module updates the matching costs corresponding to all model sequence candidates. The inter-model competition algorithm then scans the matching cost of each model gesture candidate and if the matching cost (normalized over path length) is below a certain fixed threshold then the model gesture candidate is appended to another list i_2 . The list i_2 is initialized to be empty at each frame. For every selected candidate h in the list i_2, important parameters such as start frame s , end frame n (i.e. current frame), path length l , matching cost c and gesture class g are recorded. The nearest N neighbor candidates (i.e. candidates with least matching cost) from the list i_2 vote to determine the class of the gesture being performed at the current frame n. When using sub-gesture relations: if the candidate h_i is a sub-gesture of another candidate h' or vice versa, then the sub-gesture candidate votes for the class of the super-gesture candidate. Once the gesture class is determined, a candidate belonging to recognized gesture class g^* with minimum matching cost c^* is appended to the list i_1.

The intermodal competition algorithm observes the list i_1 for several frames before it makes any decision regarding the spotted gesture. At every frame, the candidates in the list i_1 are subjected for edition based on set of rules defined as follows: 1) When using sub-gesture relations: If a candidate h_i is a sub-gesture of candidate h' or vice versa, then the sub-gesture candidate is replaced by the super-gesture candidate. 2) If start frame of a candidate h_i occurred before the end frame of the other candidate h' and model gesture candidates h_i and h' do not overlap then the candidate with worst matching cost is deleted. 3) If the candidate h_i overlaps with candidate h' then the one with lower matching cost survives. In other words a selected candidate h_i in the list i_1 is accepted only when all other overlapping candidates have

been rejected. Once a gesture has been detected (spotted and recognized), the list i_1 is reset to empty. All matching cost tables are cleared and the entire process of gesture detection (i.e. spatiotemporal matching) starts again assuming the next input frame to be the first frame of next sequence.

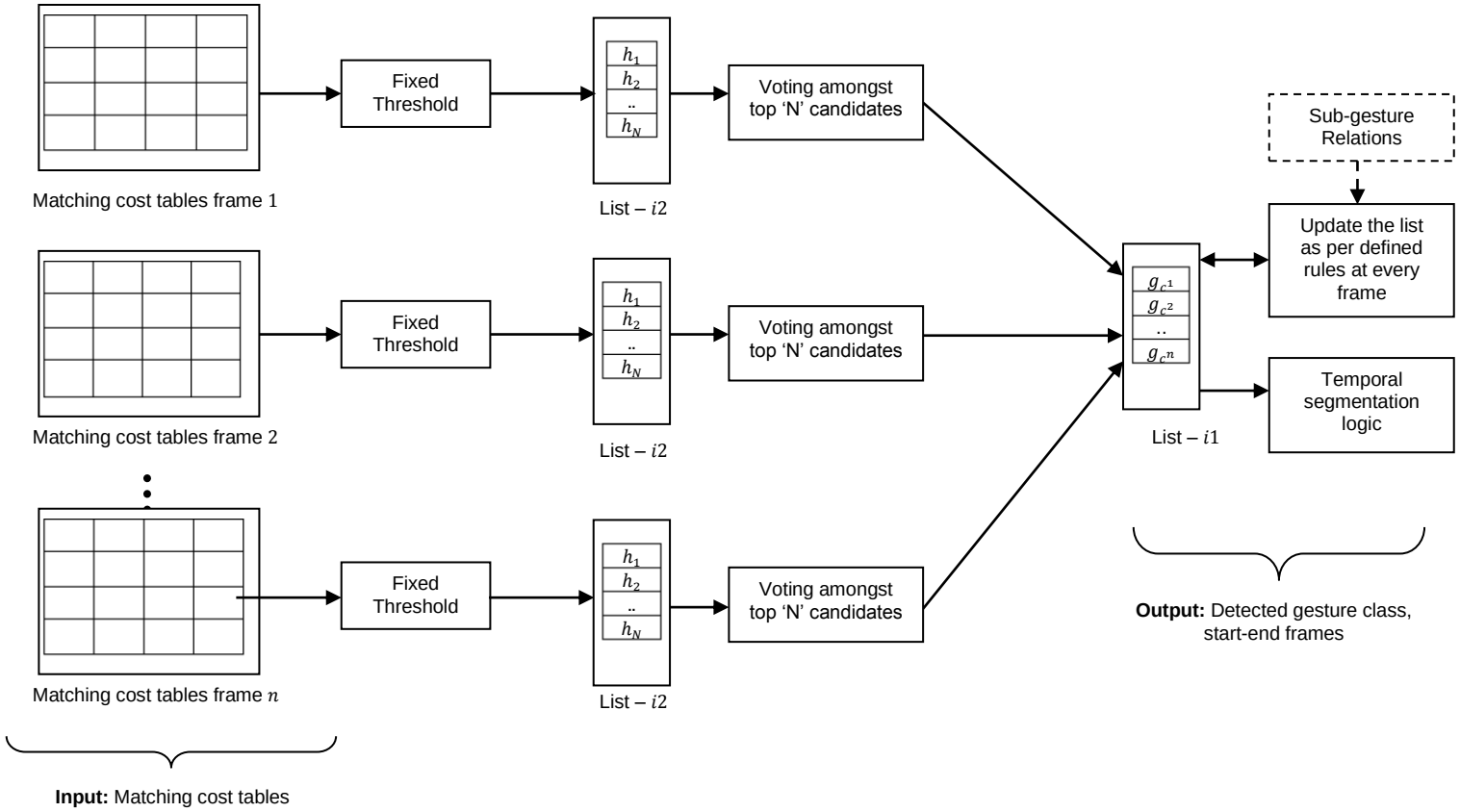


Figure 7.1 Graphical representation of the model-competition algorithm.

CHAPTER 8

EXPERIMENTS AND RESULTS

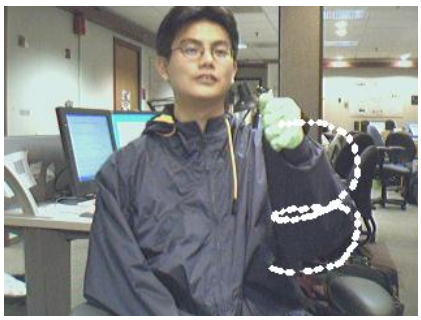
8.1 Digit Dataset

The proposed cost measures are tested using the digit dataset sequences. The digit dataset sequences contain gestures corresponding to ten digit classes (0-9) in the style of Palm Graffiti Alphabet [37] as shown in the Figure 8.1(a-f) and Figure 8.2(a-d). The video clips were captured using a Unibrain Firewire camera at the rate of 30 frames per second with an image resolution of 240x320. The digit dataset contains 30 training (model) video sequences (3 from each of the 10 users) each comprising the ten gestures (0-9) performed by users wearing colored gloves and 14 test (query) video sequences (2 from each of the 7 users) each comprising the ten gestures (0-9) performed by the users wearing short sleeved shirts and with distractors in the form of one to three humans (in addition to the gesturing user) moving back and forth in the background as shown in Figure 8.3(a-b). The presence of such distractors makes these test sequences quite challenging for the methods assuming reliable hand localization and methods dependent on global features. This set of test sequences is often referred to as the “hard” digit dataset. It should be noted that the test sequences have gestures performed sequentially in order 0 to 9. But the system is not cognizant about the order and thus makes no difference in terms of recognition accuracy. The model gestures are trained using segmented examples, so no information about the preceding digit is available. Also, there are no articulatory effect between the gestures as the hand comes back to rest position after every digit gesture is performed. All the video sequences in the training dataset as well as the manually annotated ground truth files are available at [57].

(a)

(b)

(c)

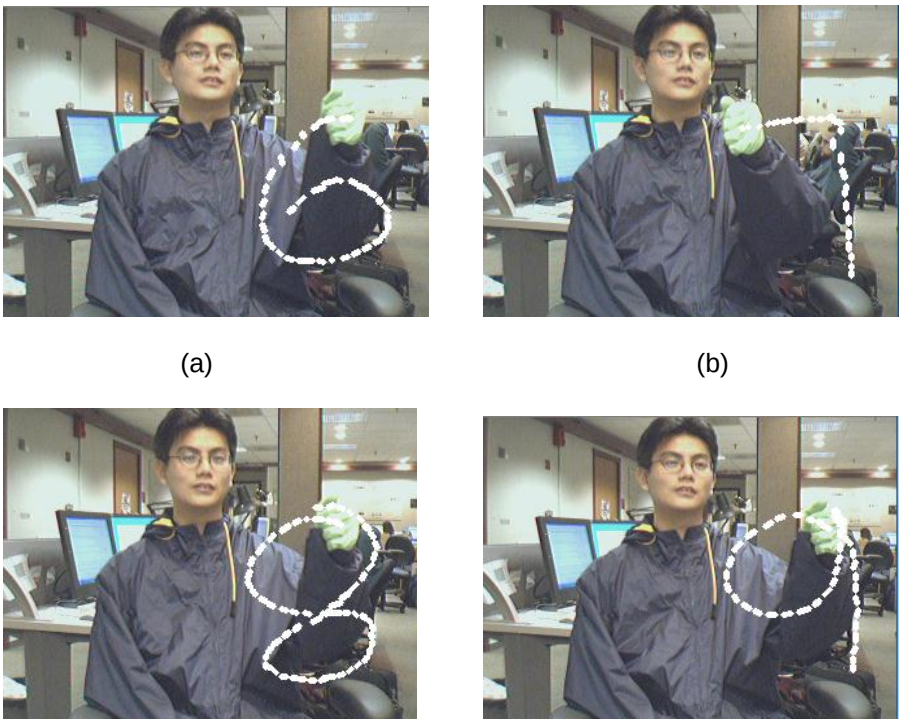


(d)

(e)

(f)

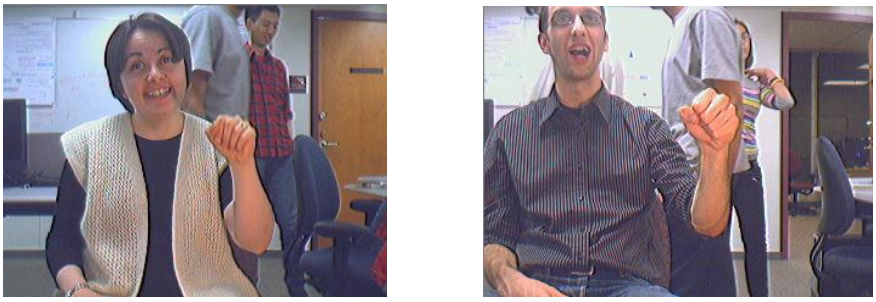
Figure 8.1(a-f) – Shows the 0-5 digit gestures of the digit dataset



(a) (b)

(c) (d)

Figure 8.2(a-d) – Shows the 6-9 digit gestures of the digit dataset



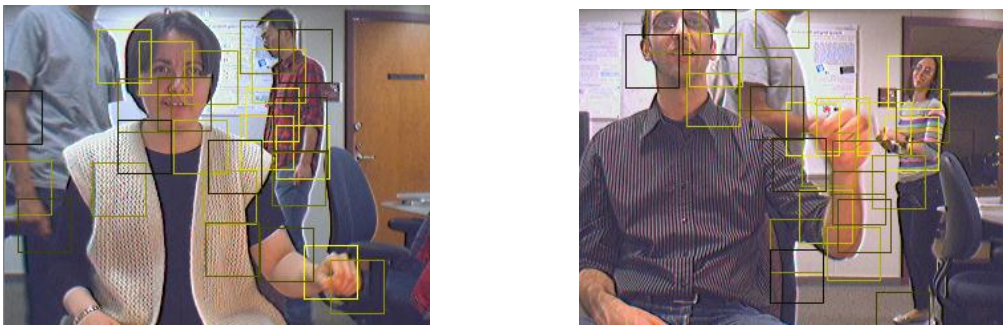
(a) (b)

Figure 8.3 (a-b) – Shows the distractors in the “hard” digit dataset present in the form of one to three persons moving in the background.

8.2 Detection and Feature Extraction

At every query sequence frame, $K = 25$ hand candidate locations each of size 40x40 are detected as described in the chapter 4. For every hand candidate a 2D feature vector is

extracted and normalized using the method described in the chapter 4. The figure 8.4 (a-b) shows top 25 hand candidates extracted from each query sequence frame.

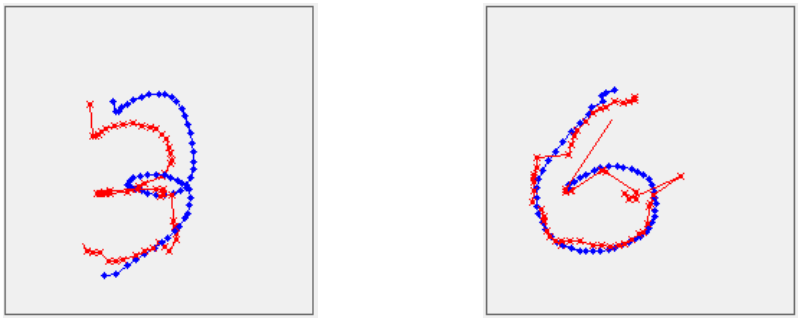


(a) (b)

Figure 8.4 (a-b) – Shows the extracted top 25 candidates at each frame.

8.3 Experiments – I

In the first phase of experiments we evaluate the each of the proposed cost measure separately. In order to evaluate each cost measure separately for recognition accuracy, it has been assumed that the system knows temporal segmentation (i.e. start and end frames of the query sequence is known). Each of the test sequence gesture is matched to the model sequence using the algorithm described in section 5.1 and the class of the query gesture is identified based on a vote taken amongst ten nearest neighbors (10-NN) rule. The figure 8.5(a) and 8.5(b) shows a correct and a false match respectively. The figure 8.6 shows a comparison graph between the classification accuracy of the various proposed cost measures in comparison with the Euclidean cost measure used as local cost (basic cost measure).



(a) (b)

Figure 8.5 – Illustrates matching of gestures. (a) shows correct gesture match, (b) shows how a gesture of class “0” is wrongly matched with gesture of class “6”

Table 8.1 – Enlists recognition accuracies for different cost measures. K is the number of hand candidates extracted per frame. The method outlined using a gray colored box and highlighted in blue is the baseline method while the rest are proposed methods.

Experiment	Initialization cost	Local cost	Transition cost	K	Recognition Accuracy (%)
Basic	Euclidean Distance	Euclidean Distance	X	5	43.57
1	constant*Euclidean Distance	Start-to-current direction error	X	7	36.43
2	constant*Euclidean Distance	X	Previous-to-current direction error	3	65.71
3	Euclidean Distance	Start-to-current vector error	X	7	66.43
4	**Euclidean Distance**	**X**	**Previous-to-current vector error**	**25**	**80.71**
5	Euclidean Distance	Start-to-current vector error	Previous-to-current vector error	14	67.85

The table 8.1 illustrates the recognition accuracy obtained for the hard dataset on various proposed and the basic cost measure. The best result was attained for $K = 25$, with a detection rate of $113/140 = 80.71\%$ when the previous-to-current frame vector error is used as

the transition cost. The scaling constant used for matching the Euclidean distance initialization cost to the direction error cost was set as 100. The mark “X” in the table indicates that the corresponding type of cost was not used in that experiment. Additionally, we perform an experiment 5 where we try to combine the start-to-current vector error cost measure with the Previous-to-current vector error cost measure.

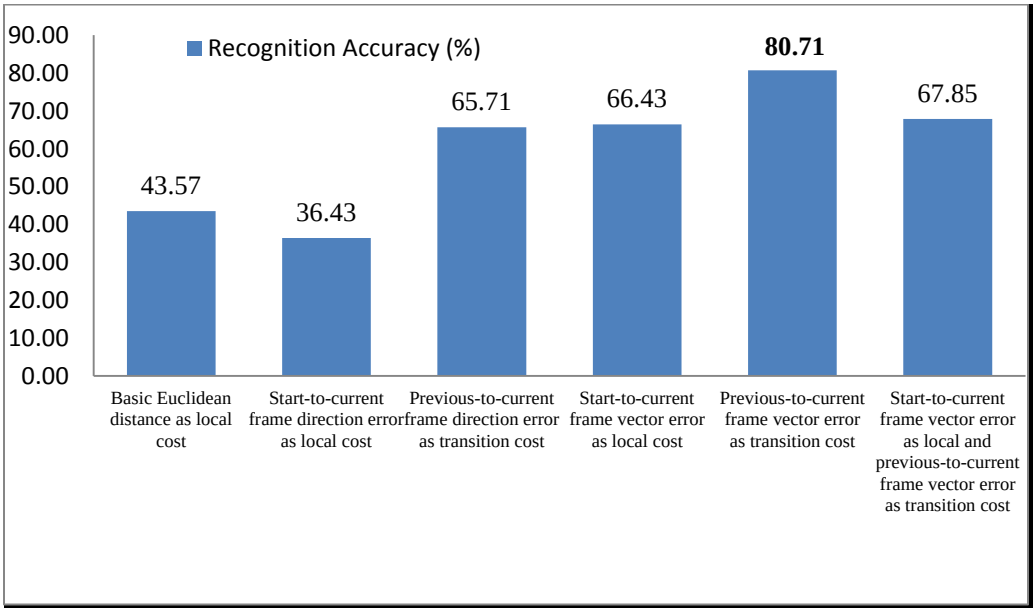


Figure 8.6– Shows the comparison graphs for recognition accuracy obtained for proposed cost measures.

8.4 Experiments II (a)

Based on the results obtained in the first phase of experiments, it is evident that the previous-to-current frame vector error method when used as the transition cost yields significantly better recognition accuracy than the basic as well as other proposed methods. We use this cost measure and apply it in a detection framework which neither knows the temporal segmentation nor the spatial segmentation. After extracting the feature vectors at each query sequence frame as described in chapter 4, the continuous DSTW modules (described in section 5.2) updates the matching cost tables corresponding to each of the model gesture candidates.

The inter-model competition algorithm (described in chapter 7) performs the challenging task of temporal segmentation and reports accuracy of the proposed frameworks in terms of detection rate (i.e. number of correct detections) and the number of false positives. The correct detection is any detection which satisfies following two necessary conditions: 1) The recognized gesture class matches with the gesture class in manually annotated ground truth gestures. 2) At least half of the detected gesture overlaps with the manually annotated ground truth gesture and vice versa, as proposed by Williams [58].

Table 8.2 – Tabulates detection rate and false positives for Experiments II (a) – GSwoSR (i.e. Gesture Spotting without Sub-gesture Relations). The table also compares the basic Euclidean distance as local cost method and proposed Previous-to-current frame vector error as transition cost method on continuous recognition of digits.

Results for Experiment II(a) – Gesture Spotting without sub-gesture relations				
	Basic Euclidean distance as local cost method		**Previous-to-current frame vector error as transition cost method**	
K	**Detection Rate (%)**	**False Positives**	**Detection Rate (%)**	**False Positives**
1	19.28571	42	30.71429	49
3	20	60	40.71429	72
5	21.42857	76	40.71429	64
7	**22.85714**	**64**	44.28571	71
10	22.85714	69	52.85714	47
15	21.42857	70	58.57143	52
20	21.42857	63	59.28571	48
25	19.28571	79	**61.42857**	**51**

The table 8.2 illustrates the detection rate and false positives obtained for the hard dataset in the experiment II (a) i.e. Gesture spotting without sub-gesture relations. The best result without using sub-gesture relations was attained for $K = 25$, with a detection rate of $86/140 = 61.43\%$ and 51 false positives when using the proposed Previous-to-current vector error as the transition cost. The basic Euclidean distance as local cost method in comparison attained best detection rate of $32/140 = 22.86\%$ at $K = 7$.

8.5 Experiments II (b)

Further we extend the experiments performed in second phase by adding the knowledge of sub-gesture relations to the framework. The inter-model competition algorithm

used in these experiments use manually defined sub-gesture relationships as stated in table 8.3. We note that the use of sub-gesture relations improve the performance in overall by increasing detection accuracy and bringing down the false detections.

Table 8.3 – Enlists manually defined sub-gesture relations

Sub-gesture class	Super-gesture class	Sub-gesture class	Super-gesture class
1	{ 9, 7 }	2	{ 7 }
4	{ 5, 8, 9 }	5	{ 8 }
7	{ 2 , 3 }	3	{ 7 }

The table 8.4 illustrates the detection rate and false positives obtained for the hard dataset in the experiment II (b) i.e. Gesture spotting with sub-gesture relations. The best result with using sub-gesture relations was attained for $K = 25$, with a detection rate of $99/140 = 70.71\%$ and 39 false positives when using the proposed Previous-to-current vector error as the transition cost. The basic Euclidean distance as local cost method in comparison attained best detection rate of $37/140 = 26.43\%$ at $K = 15$.

Table 8.4 – Tabulates detection rate and false positives for Experiments II(b) – GSwSR(i.e. Gesture Spotting with Sub-gesture Relations). The table also compares the basic Euclidean distance as local cost method and proposed Previous-to-current frame vector error as transition cost method on continuous recognition of “hard” digit dataset when using the sub-gesture relations.

Results for Experiment II(b) – Gesture Spotting with sub-gesture relations				
	Basic Euclidean distance as local cost method		**Previous-to-current frame vector error as transition cost method**	
K	**Detection Rate (%)**	**False Positives**	**Detection Rate (%)**	**False Positives**
1	20	41	32.14286	49
3	21.42857	59	40.71429	74
5	25.71429	71	45	74
7	25	65	46.42857	76
10	25	72	55.71429	44
15	**26.42857**	**68**	60	42
20	20.71429	38	63.57143	42
25	22.14286	73	**70.71428**	**39**

The methods such as [5 – 12] assuming reliable hand localization and [21 – 26] using global features have not been demonstrated on the scenes where the presence of distractors is

comparable to the presence of distractors in the hard digit dataset. Further, when we make $K = 1$ the basic method reduces to the CDP method of Oka [42]. It is evident from the table 8.4 that the method proposed here (previous-to-current frame vector error as the transition cost) with any $K \neq 1$ produces significantly higher detection rates. From these results, it can be inferred that the DSTW which uses simple hand detector and allows multiple hand candidate at each frame, to a great extent, can overcome the challenge posed by presence of distractors in the hard dataset.

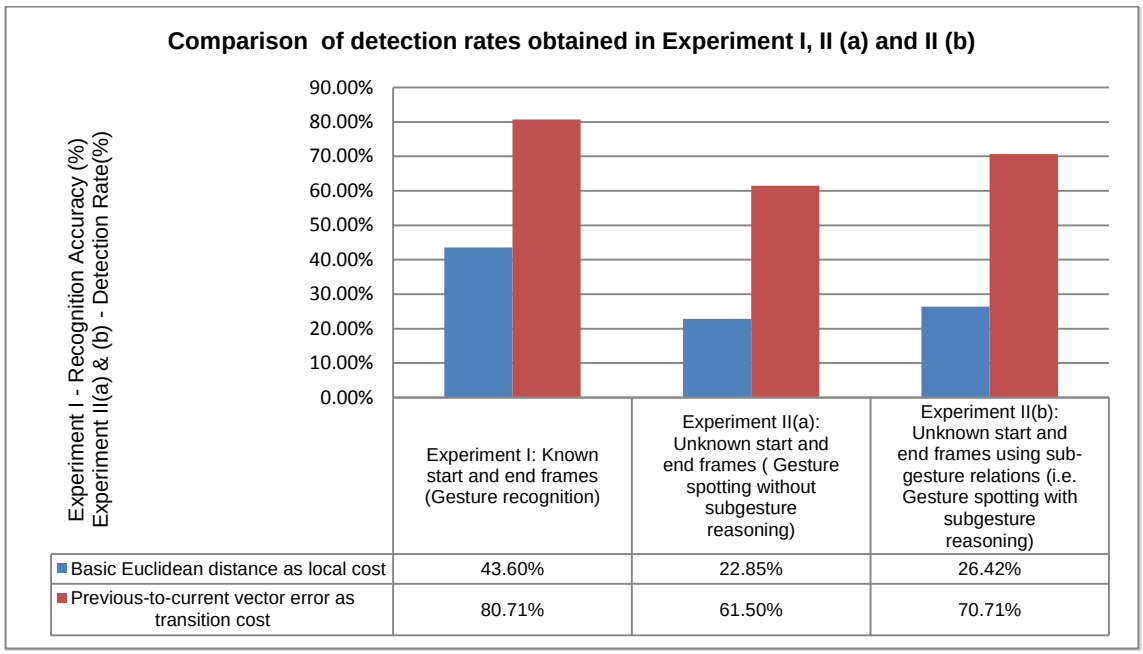


Figure 8.7 – Shows comparison graph between Experiments-I (Gesture Recognition), Experiment II (a) (i.e. Gesture Spotting without Sub-gesture Relations) and Experiment II (b) (i.e. Gesture Spotting using Sub-gesture Relations).

8.6 Software Implementation

All the experiments have been performed on a 2.0 GHz dual core processor using software implementation in MS VC++ [59].The detection and feature extraction is performed with the help of MATLAB [60]. For real-time implementation of the framework the openCV library [61] was used for detection and feature extraction. In experiments II (a) and II (b) only

one out of every five input frames were used by the continuous DSTW module for further processing.

CHAPTER 9

DISCUSSION, CONCLUSION AND FUTURE WORK

In this thesis we have presented a few novel cost measures to achieve improved recognition accuracy for robust recognition of hand gesture using the translation and scale invariant feature vectors in a real world scene with cluttered background and in presence of multiple moving skin colored distractors in the background. The DSTW algorithm addresses the need to accommodate multiple hand locations per frame instead of assuming prefect hand detection. The method does not require low level detection module to perform spatial or temporal segmentation. The higher level modules namely the continuous DSTW module and inter-model competition module handle the spatial and temporal segmentation; thus avoiding the drawbacks of the bottom-up approach. The method incorporates robustness into the recognition at no additional computational cost by introducing translation and scale invariance into the feature vectors. The system is robust enough to sustain with users performing gestures wearing short sleeved shirts and in presence of distractors in the form of one to three persons in the background.

The use of DSTW algorithm improves the recognition accuracy by a factor of 2.6 in terms of detection accuracy of the vector error as transition cost method as compared to the CDP approach proposed by Oka et al. [42] using the basic Euclidean distance as the local cost method. The vector error cost measure when implemented as the transition cost outperforms other proposed cost measures. It should be noted that the improved accuracy in the case of vector error cost measure comes at the computational cost of calculating the transition cost of $(2K-1)$ previous hand candidates for each of the hand candidate in present frame. The computational complexity of the proposed vector error cost measure is $O(K^2mn)$ which higher

as compared to the approach proposed by Alon et al [20] $O(Kmn)$ and CDP proposed by Oka et al. [42] $O(mn)$.

Also the direction error cost measure does not seem to perform well as compared to the basic Euclidean distance cost measure when used as the local cost. The important reason for this being the nature of the direction error cost measure which does not take into account the distance between the hand candidates. E.g. in the figure 9.1 the direction traced by frames 1 and 6 (i.e. we call $Direction(1 \rightarrow 6)$) and direction traced by frames 1 and 8 (i.e. we call $Direction(1 \rightarrow 8)$) are same. The direction error cost measure cannot differentiate between $Direction(1 \rightarrow 6)$ and $Direction(1 \rightarrow 8)$.

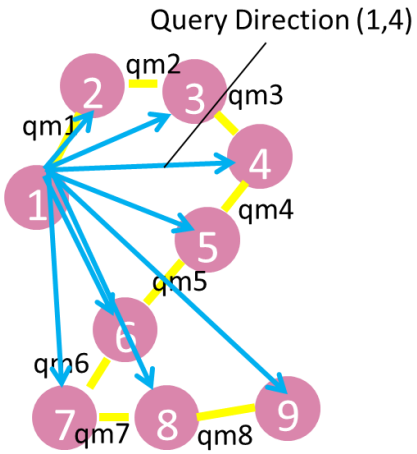


Figure 9.1 – Illustrates limitation of the direction error cost measure. The direction error cost measure cannot differentiate between $Direction(1 \rightarrow 6)$ and $Direction(1 \rightarrow 8)$.

The direction error cost measure can often wrongly match model-query frame with distant hand locations. Hence the vector error cost measure was introduced in order to add a distance measure to the direction. Despite of the limitations of the direction error cost measure, it should be noted that the direction error is inherently translation and scale invariant.

In order to make the framework more robust a rotational invariance can be introduced which will take care of the rotations occurring in the image plane of the user performing gesture.

The use of pruning classifiers as proposed by Alon et al. [20], [39] can be used to make the framework more efficient in terms of computational complexity. An interesting idea would be to learn a classifier by adding several weighted cost measures. Further, the depth color information can improve the hand detection (i.e. need of less hand candidate gesture per frame) and thus improve the overall performance of the system in terms of the efficiency and accuracy [62]. The proposed method can be extended and applied to the American Sign Language recognition. Some key features of the method like the ability to perform well in translation and scale invariant feature vector as well as the ability to operate in absence of reliable hand detection are desirable for building robust sign language recognition systems.

REFERENCES

[1] W. Freeman. Computer vision for television and games. In Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems (RATFG-RTS), page 118, 199.

[2] J. Triesch and C. von der Malsburg. Robotic gesture recognition. In Gesture Workshop, pages 233–244, 1997.

[3] T. Starner and A. Pentland. Real-time American Sign Language recognition from video using hidden markov models. In IEEE International Symposium on Computer Vision, pages 265–270, 1995.

[4] Palm. Grafitti alphabet. http://www.palmone.com/us/products/input/.

[5] R. Cutler and M. Turk, “View-based interpretation of real-time optical flow for gesture recognition,” in Proc. Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998, pp. 416–421.

[6] A. Corradini, “Dynamic time warping for off-line recognition of a small gesture vocabulary,” in Proc. IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001, pp. 82–89.

[7] T. Darrell and A. Pentland, “Space-time gestures,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1993, pp. 335–340.

[8] M. Gandy, T. Starner, J. Auxier, and D. Ashbrook, “The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring,” in International Symposium on Wearable Computing, 2000.

[9] K. Oka, Y. Sato, and H. Koike, “Real-time fingertip tracking and gesture recognition,” IEEE Computer Graphics and Applications, vol. 22, no. 6, pp. 64–71, 2002.

[10] T. Starner, J. Weaver, and A. Pentland, “Real-time American Sign Language recognition using desk and wearable computer based video,” Pattern Analysis and Machine Intelligence, vol. 20, no. 12, pp. 1371–1375, December 1998.

[11] M. H. Yang, N. Ahuja, and M. Tabb, “Extraction of 2D motion trajectories and its application to and gesture recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 8, pp. 1061–1074, 2002

[12] V. Pavlovic, R. Sharma, and T. Huang, “Visual interpretation of hand gestures for human-computer interaction: A review,” Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 677–695, July 1997.

[13] Y. Cui and J. Weng, “Appearance-based hand sign recognition from intensity image sequences,” Computer Vision and Image Understanding, vol. 78, no. 2, May 2000.

[14]] E. Ong and R. Bowden, “A boosted classifier tree for hand shape detection,” in Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004.

[15] M. Isard and A. Blake, “CONDENSATION - conditional density propagation for visual tracking,” Intenational Journal of Computer Vision, vol. 29, no. 1, pp. 5–28, 1998.

[16] M. Kolsch and M. Turk, “Fast 2D hand tracking with flocks of features and multi-cue integration,” in IEEE Workshop on RealTime Vision for Human-Computer Interaction, 2004, pp. 158–165.

[17] N. Stefanov, A. Galata, and R. Hubbold, “Real-time hand tracking with variable-length Markov Models of behaviour,” in Real Time Vision for Human-Computer Interaction, 2005, pp. III: 73–73.

[18] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla, “Filtering using a tree-based estimator,” in Proc. IEEE International Conference on Computer Vision, 2003

[19] E. Sudderth, M. Mandel, W. Freeman, and A. Willsky, “Visual hand tracking using nonparametric belief propagation,” in Proc.IEEE CVPR Workshop on Generative Model

Based Vision, 2004.

[20] Jonathan Alon, Vassilis Athitsos, Quan Yuan, and Stan Sclaroff. A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation.*IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(9), September 2009.

[21] A. Bobick and J. Davis, “The recognition of human movement using temporal templates,” Pattern Analysis and Machine Intelligence, vol. 23, no. 3, pp. 257–267, March 2001.

[22] P. Dreuw, T. Deselaers, D. Keysers, and H. Ney, “Modeling image variability in appearance-based gesture recognition,” in ECCV Workshop on Statistical Methods in Multi-Image and Video Processing, 2006, pp. 7–18.

[23] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, “Actions as space-time shapes,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 12, pp. 2247–2253, 2007.

[24] S. Nayak, S. Sarkar, and B. Loeding, “Unsupervised modeling of signs embedded in continuous sentences,” in IEEE Workshop on Vision for Human-Computer Interaction, 2005.

[25] Y. Ke, R. Sukthankar, and M. Hebert, “Efficient visual event detection using volumetric features,” in IEEE International Conferenceon Computer Vision (ICCV), vol. 1, 2005, pp. 166–173

[26] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, pages 511–518, 2001.

[27] Alexandra Stefan, Vassilis Athitsos, Jonathan Alon, and Stan Sclaroff. Translation and Scale-Invariant Gesture Recognition in Complex Scenes. *Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, July 2008.

[28] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face

detection. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 38–44, 199

[29] Paul Viola and Michael Jones, Robust Real-time Object Detection, second international workshop on statistical and computational theories of Vision – modeling, learning, computing, and sampling, 2001.

[30] J. Wang, V. Athitsos, S. Sclaroff, and M. Betke. Detecting objects of variable shape structure with hidden state shape models. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 30(3):477–492, 2008

[31] Jonathan Alon, Vassilis Athitsos, Quan Yuan, and Stan Sclaroff. Simultaneous Localization and Recognition of Dynamic Hand Gestures. *IEEE Motion Workshop*, pages 254-260, January 2005.

[32] J. B. Kruskall and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. In Time Warps. Addison-Wesley, 1983

[33] L. Rabiner and B. Juang. Fundamentals of speech recognition. Prentice Hall, 1993

[34] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. In IEEE Transactions on Acoustics, Speech, and Signal Processing, volume 34(1), pages 43–49, 1978

[35] E. Keogh. Exact indexing of dynamic time warping. In International Conference on Very Large Data Bases, pages 406–417, 2002.

[36] C. Rasmussen and G. Hager. Probabilistic data association methods for tracking complex visual objects. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 23(6):560–576, June 2001

[37] M. Black and A. Jepson. Recognizing temporal trajectories using the condensation algorithm. In Automatic Face and Gesture Recognition, pages 16–21, 1998

[38] Y. Sato and T. Kobayashi, “Extension of Hidden Markov Models to deal with multiple

candidates of observations and its application to mobile-robot-oriented gesture recognition," in Proc. International conference on Pattern Recognition, 2002.

[39] J. Alon, V. Athitsos, and S. Sclaroff, "Accurate and efficient gesture spotting via pruning and subgesture reasoning," in Proc. IEEE ICCV Workshop on Human Computer Interaction, 2005, pp. 189–198.

[40] H. Lee and J. Kim, "An HMM-based threshold model approach for gesture recognition," Pattern Analysis and Machine Intelligence, vol. 21, no. 10, pp. 961–973, October 1999.

[41] P. Morguet and M. Lang, "Spotting dynamic hand gestures in video image sequences using Hidden Markov Models," in Proc. IEEE International Conference on Image Processing, 1998, pp. 193–197.

[42] R. Oka, "Spotting method for classification of real world data," The Computer Journal, vol. 41, no. 8, pp. 559–565, July 1998.

[43] H. Yoon, J. Soh, Y. Bae, and H. Yang, "Hand gesture recognition using combined features of location, angle and velocity," Pattern Recognition, vol. 34, no. 7, pp. 1491–1501, July 2001.

[44] Y. Zhu, G. Xu, and D. Kriegman, "A real-time approach to the spotting, representation, and recognition of hand gestures for human-computer interaction," Computer Vision and Image Understanding, vol. 85, no. 3, pp. 189–208, March 2002.

[45] H. Kang, C. Lee, and K. Jung, "Recognition-based gesture spotting in video games," Pattern Recognition Letters, vol. 25, no. 15, pp. 1701–1714, November 2004.

[46] K. Kahol, P. Tripathi, and S. Panchanathan, "Automated gesture segmentation from dance sequences," in Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, pp. 883–888.

[47] T. Darrell, I. Essa, and A. Pentland, "Task-specific gesture analysis in real-time using interpolated views," Pattern Analysis and Machine Intelligence, vol. 18, no. 12, pp. 1236–1242, 1996.

[48] J. B. Kruskall and M. Liberman, “The symmetric time warping algorithm: From continuous to discrete,” in Time Warps, String Edits and Macromolecules, J. B. Kruskal and D. Sankoff, Eds. Addison-Wesley, 1983, pp. 125–162.

[49] M. Brand, N. Oliver, and A. Pentland, “Coupled Hidden Markov Models for complex action recognition,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 994–999.

[50] C. Vogler and D. Metaxas, “Parallel Hidden Markov Models for American Sign Language recognition,” in Proc. IEEE International Conference on Computer Vision, 1999, pp. 116–122.

[51] A. Wilson and A. Bobick, “Parametric Hidden Markov Models for gesture recognition,” Pattern Analysis and Machine Intelligence, vol. 21, no. 9, pp. 884–900, September 1999.

[52] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random17 fields: Probabilistic models for segmenting and labeling sequence data,” in Proc. 18th International Conference on Machine Learning. Morgan Kaufmann, San Francisco, CA, 2001, pp. 282–289.

[53] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, “Hidden conditional random fields,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 10, pp. 1848–1852, 2007.

[54] F. Chen, C. Fu, and C. Huang, “Hand gesture recognition using a real-time tracking method and Hidden Markov Models,” Image and Video Computing, vol. 21, no. 8, pp. 745–758, August 2003

[55] Zhong Zhang, Rommel Alonzo, and Vassilis Athitsos. Experiments with Computer Vision Methods for Hand Detection. *Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, May 2011.

[56] J. Alon, “Spatiotemporal gesture segmentation,” Ph.D. dissertation, Dept. of Computer

Science, Boston University, Technical Report BU-CS-2006-024, 2006

[57] Digit Dataset. http://vlm1.uta.edu/~athitsos/projects/digits

[58] G. Williams, “A study of the evaluation of confidence measures in automatic speech recognition,” University of Sheffield, Tech. Rep. CS-98-02, 1998

[59] Microsoft VC++ – http://msdn.microsoft.com/en-us/vstudio

[60] MATLAB – www.mathworks.com

[61] OpenCV – http://opencv.willowgarage.com/documentation/

[62] Paul Doliotis, Alexandra Stefan, Chris Mcmurrough, David Eckhard, and Vassilis Athitsos. Comparing Gesture Recognition Accuracy Using Color and Depth Information. *Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, May 2011.

BIOGRAPHICAL INFORMATION

Ameya Kulkarni was born in India in 1987. He completed his Bachelor of Engineering in Electronics and Telecommunication from Vishwakarma Institute of Technology (affiliated to University of Pune), Pune. He obtained his Master of Science degree from University of Texas at Arlington in December 2011. He is also a member of UTA chapter of Tau Beta Pi Engineering Honor Society. He worked for Research In Motion as an OS Embedded Software Engineer – Student from January 2011 – December 2011. His current research interests include computer vision, image processing and embedded systems.