Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Master Thesis

A CHnMM Approach for Multi Stroke Gesture Recognition on Touch Input Devices

Author:

Umesh Bastola

August 1, 2018

Advisors:

Dr. Ing. Claudia Krull

Tim Dittmar

Department of Computer Science

Institute for Simulation and Graphics

Bastola, Umesh:
A CHnMM Approach for Multi Stroke Gesture Recognition on Touch Input Devices
Master Thesis, Otto-von-Guericke University Magdeburg, 2018.

Abstract

A modeling approach for spatio-temporal movement trajectories that is based on the stochastic model class called conversive hidden non-Markovian models (CHnMMs) has already been implemented for single stroke gesture recognition. This thesis extends the previous work by applying the state of art model to work with multi-stroke gestures. We define a set of fourteen gestures which differ in terms of number of strokes and speed of execution. Two separate approaches for recognition of those gestures, one that creates a single model per stroke and another that creates a single model per gesture are presented. With these models, we analyze the shape of the gesture's motion trajectories and the temporal and spatial relations between trajectories in order to characterize a multi-stroke gesture. To set-up, train and validate our systems, we built a multi-stroke gesture dataset formed by 2870 gestures collected from 11 volunteers.

Experimentation are carried out to identify the trustworthiness and weaknesses of the procedures. Both approaches are evaluated using recognition of touch gesture experiments and are compared with $N multi-stroke recognizer results. The results show better recognition rate for single model per gesture approach compared to both single model per stroke and $N multi-stroke gesture recognizer. Based on these results we concluded that CHnMM is promising and can recognize multi-stroke gesture when both shape and time dimension are taken into account. Also, as a possible use-case of this model, a prototypical gesture authentication system is implemented on a web based platform.

Acknowledgements

I would like to thank all the people who have helped and inspired me during the pursuit of my masters degree. First, I would like to thank my supervisors, Dr. Ing. Claudia Krull and Tim Dittmar, for their guidance during my research and study at the Otto-von-Guericke University Magdeburg. Both of my supervisors were always accessible and willing to help with my research making my research experience a smooth and a rewarding one. I also would like to thank my friends Ajay, Aiham, Jesse, Manisha, Rita, Sandeep, Sanjay, Sudeep, Sujan and Sumit for helping me with gestures required for this thesis. If it were not for them, I don't think that I would have been able to collect proper data that was a key component of my thesis. I, especially, would like to thank Sudeep for providing his iPad in the early phase of my thesis work and Sujan and Rita for their incredible support and encouragement throughout my masters degree. My deepest gratitude and appreciation to my family for their unyielding love and support throughout my life; this thesis would have simply been impossible without their help. I am completely indebted to my parents, for their overwhelming care and encouragement. No words can express my appreciation to them for their everlasting love and her constant support whenever I encountered difficulties. Above all, thanks be to God for all the blessings in my life and for giving me this opportunity to achieve a higher level of education. My life has become more bountiful.

Contents

List of Figures

List of Acronyms

CHnMM Conversive Hidden non Markovian Model

DTW Dynamic Time Warping

EER Equal Error Rate

FAR False Acceptance Rate

FRR False Rejection Rate

HMM Hidden Markov Model

HnMM Hidden non Markovian Model

IoT Internet of Things

RNN Recurrent Neural Networks

STMM Spatio Temporal Movement Model

TAR True Acceptance Rate

1. Introduction

1.1 Motivation

With the advancement of technology, many Information Technology related companies such as mobile and laptop companies for example, are globally cooperating or competing to bridge the gap between humans and computers. Multi-touch screens are gaining popularity and replacing peripheral devices like pen, mouse and keyboard as they support natural human behavior such as tap, touch, grab, drag, wave etc. Computers are designed to react to these gestures accordingly. Therefore, human gestures are considered to be one of the many successful bridging components. These gesture triggers have eased the use of computers and devices not only to the differently abled personalities but also able-bodied humans in various ways.

A gesture may have various meanings depending on where and at what speed the gesture is executed. For example, the hand gestures of a conductor at the opera. The musicians react with respect to the movement and speed of the conductor's hand. This means we can now devise that; a gesture is not just the trajectory of the moving object but it also encompasses the time value. It will be of significant value if we could develop a model that will be able to represent both of these aspects of a gesture.

The Hidden Markovian Model(HMM)[10] has proved to be quite a successful model for gesture and speech recognition techniques[11, 13]. However, the extension of Hidden non Markov Models(HnMM)[Krull and Horton], Conversive Hidden non Markovian Models(CHnMM) [4] have proved to be more promising than HMM for gesture[6] and pattern[3] recognition in terms of recognition accuracy when the shape of the gestures is not only the discriminating factor but its temporal dynamics as well. Tim Dittmar[6] has already implemented this CHnMM model for gesture recognition of single stroke gestures. We find the room for improvement to this model by extending its usability to work with multiple finger strokes. A successful implementation of the model covers a vast scope. For example it can be implemented to build a system that recognizes

the hand movement of the opera conductor in the real time environment and play various notes according to the interpreted tempo. Similarly, the aimed model can be implemented into a full fledged authentication system that uses user-specific gesture as password instead of conventional text-based password. These sparks of possibilities make this thesis more interesting.

1.2 Aim of the Thesis

Abundant research has been carried out in the field of gesture recognition. Researches have gained huge success in this field and are constantly looking forward to strengthen the recognition process and increase the trustworthiness of the system. With the same hope, in Chapter 3, we try to find out if the time dimension can be added as a feature of multi-stroke gesture recognition or not. Also, we hope to figure out the possibility of using CHnMM for multi-stroke gesture recognition. To reach these goals, the following tasks were identified:

- create a gesture set comprising of heterogeneous gestures,
- create gesture dataset from volunteers,
- propose and implement possible multi-stroke gesture recognition techniques that can extend existing single STMM per trajectory,
- compare and evaluate results of proposed approaches and decide if goals have been reached.

From our research, if we can derive that time dimension can be added to multi-stroke gesture recognition with the help of CHnMM, we will be able to differentiate among the multi-stroke gestures which have similar spatial properties but differ in terms of execution speed. This in turn, will increase the trustworthiness of gesture recognition process.

1.3 Structure of the Thesis

This document is structured in mainly three parts, introduction, methods and evaluation. In the first part, Chapter 1 contains introduction to the thesis, motivation behind it along with discussions about the intended goals of this thesis. There have been years of studies in the field of gesture recognition. In Chapter 2, we introduce the state of art research works which are closely related to our topic. We explore the different types of underlying recognition methods for both single stroke and multi-stroke gesture, which gives the background and idea of the thesis. Also, we go through the previous work accomplished by Tim Dittmar which serves as the building ground for our research work. Finally in this part, the necessary theoretical background of the methods we use throughout the experiments and evaluation will be discussed here.

In the second part, beginning with Chapter 3, we present a concrete definition of the gestures we are going to use in this research and build a multi-stroke gesture dataset based on these definitions. As a concrete step towards our target, we propose single STMM(Spatio Temporal Movement Model) per stroke and single STMM per gesture approaches that can be used for multi-stroke gesture recognition. Later in Chapter 3 we will see how we implemented our method to fit into previous work and how we get the result from that.

In the last part of this thesis, we carry out recognition and verification experiments with both of the approaches introduced in Chapter 3. Comparisons between the results of these approaches and results of the $N gesture recognizer are done. The evaluations of these methods based on different metrics like FAR, FRR, EER etc. are carried out. Moreover, the possibility of using CHnMM based gesture recognition system for authentication process is discussed. The final chapter, Chapter 5 will have the summary of evaluation and discussion in the previous chapters. Also, the final chapter contains the future works that are relevant to the scope of this topic.

2. Related Work and Background

Several studies have been carried out in recognition of gestures; from facial gestures to hand gestures, from 2D to 3D gesture recognition processes, from touch based to touch less user interface systems for example. In our study, we are concerned about the gesture recognition methods that can handle multiple finger strokes at the same time. In this section we discuss some related works and present the previous work that is the basis for this thesis. Similarly, to clarify the term related to this domain, in section 2.2, we will see the formal definition of CHnMM.

2.1 Classification of Touch Gestures

Based on the various interaction contexts, touch gesture can be differentiated with respect to the number of strokes and number of simultaneous touches. Inspired by [26], we categorized the different type of touch gestures based on the number of strokes and touches which can be seen in Figure 2.1. Here, a stroke is regarded as the trajectory of a touch (finger, stylus, etc.) during uninterrupted contact on the sensing surface. The touch number indicates the number of concurrent contacts involved. Single-stroke gestures are the gestures that consists of only one touch and only one stroke is drawn. For example, the swipe image feature in most of the touch enabled mobile devices is a single stroke gesture manipulation task. Multi-stroke gestures are gestures which contain at least two strokes and can be generated with sequences of single touches or by multiple fingers at the same time. For example, the pinch zoom feature in touch enabled devices is a multi-stroke gesture with multi touch. The sequential multi-stroke is a more complex gesture such that more than one concurrent touch is involved, which incorporates at least two subsequent strokes and simultaneous touches. In this thesis, we deal with multi-stroke gestures with more than one touch according to this classification. For the ease of readers, we will refer to this as multi-touch gestures throughout this paper.

		Number of strokes per touch	
		1	>1
Number of touch involved	1	Single stroke	Multi-stroke (Single touch)
	>1	Multi-stroke (Multi touch)	Sequential multi-stroke

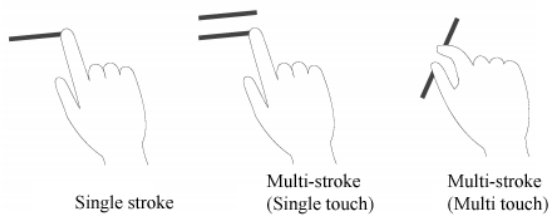


Figure 2.1: Gesture classification based on the number of strokes and touches, different type of gestures

2.2 Related work

HMM based applications are a common research topic for 3D gesture recognition, especially for body parts movement [18, 19, 31, 32]. For example, in [12], authors claim to recognize 20 different dynamic hand gestures with the recognition accuracy of 89.25%. However, the training process for HMM requires a significant amount of time to generate the model and requires higher number of training examples compared to other techniques such as DTW(Dynamic Time Warping) to produce models of good quality-Fierrez et al. [9]. Despite these and many other studies, discriminating similar shaped trajectories by their temporal dynamics has never been thoroughly stated in these researches.

In [3], authors have shown that Hidden non-Markovian Models are applicable in the field of gesture recognition and under certain circumstances, non-Markovian models provide a better distinction of gestures with different execution speeds than Markovian models do. This paper proved the feasibility of time dependent state transitions to distinguish similar signal sequences by their output speed. Best of our interest, HnMMs therefore, are able not only to decide whether an arbitrary pattern was executed correctly, but also whether it was executed at the right speed. However, the manual training process(development of the basic model and the computation of state change distributions and output probabilities) used was not proven to be robust and was a tedious process. In the following, we will see some thesis relevant gesture recognition studies that are chosen as representative of the gesture recognition researches in the past.

2.2.1 Single Stroke Gesture Recognition

As a technique to recognize the multi-stroke gestures, recognizing individual strokes within these gestures and later combining the results into one is also proposed. Therefore, recognition of single stroke gestures are also of equal interest to this thesis.

Sequence matching and statistical recognition based on global features are commonly used as recognition strategies for single touch gestures. In [30], authors introduce a com-

petitive sequence matching algorithm named $1 recognizer for single stroke recognition. This algorithm stores a set of gestures as template. When it comes to recognition, the candidate gesture as well as original stroke is re-sampled into a fixed number of points and a candidate stroke is compared to each stored template to find the average distance between corresponding points. A point to point matching equation is then implemented to get this distance. The template with the smallest distance to the candidate is declared as the result of the recognition process. This recognizer is claimed to be light weight and accurate with only a few loaded templates on a small dataset. Similar distance matching technique can be implemented for our spatial model but this will significantly increase the computation time since each stroke of gesture has to compared with every template in the database. Though sequence matching algorithms are effective for small sets of simple gestures, their efficiency degrades with high variation in gesture set and variation in gesture drawing patterns. Improvement from $1 to $P[28] have been made to handle single stroke and multi-stroke gestures, however this change does not represent gesture as ordered points, which is of main importance for us. Therefore it is not applicable in our system.

A non-linear sequence matching algorithm, dynamic time warping(DTW) matching is studied in [8, 14, 22] for classifying time series and employed for gesture recognition. For example, Niels and Vuurpijl [21] makes use of DTW for recognition of Tamil characters. Local features of each point such as co-ordinate values, velocity, curvature, pressure, acceleration etc. can also be extracted as part of DTW match features but this study uses just x and y co-ordinate values. The DTW is used to calculate the distance between two sequences according to the point to point matching. This process is described with the help of Figure 2.2.

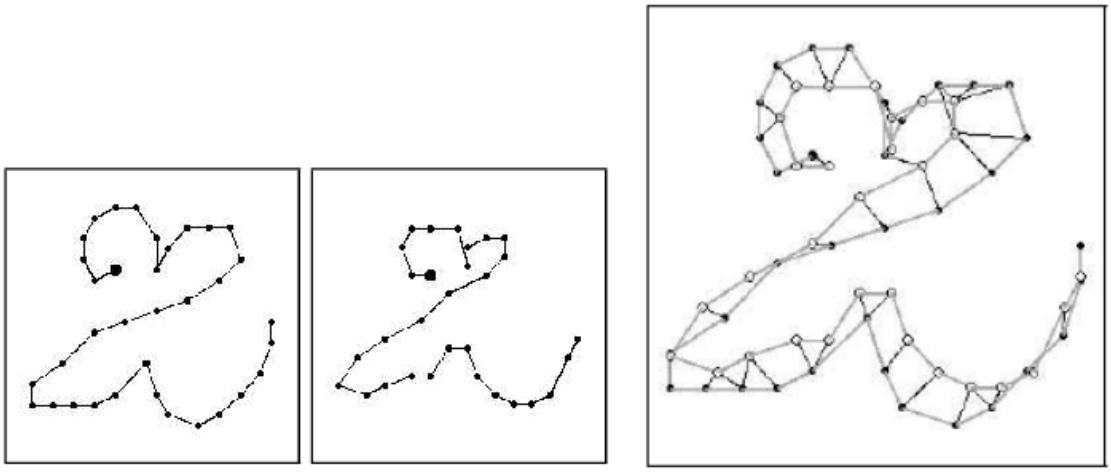


Figure 2.2: DTW character matching example

In the figure, two sample series that are similar to each other are available. An optimal global alignment between the two series are made with the help of temporal distortion between them. One point for example may be used to compare more than one points in another series to find optimal global alignment as seen in the DTW matching of sample series.

Even though DTW is able to match time series, it does not use actual time-stamp information in its calculation. Our desired system must represent temporal dynamics of a gesture. We found no studies that makes clear statement about usability of DTW for these kinds of tasks where time information is used. Therefore, this does not make DTW a good choice for our task. Even though we want to continue with DTW, there is another shortcoming. Our training set will have a large number of templates, DTW takes an unlabeled input stream of templates and calculate the minimum distance between the input and each template stream to classify input as whichever template stream it matches most closely. This makes the use of DTW inconsiderably expensive.

Another single stroke recognition study[24] makes use of statistical recognition based on the global features of a single stroke gesture. The global features are selected in such a way that, each feature is meaningful so that it can be used in gesture semantics as well as for recognition. And the feature set consists of enough features so that it can provide differentiation between all gestures and also not too many that it reduces efficiency. From the features they are using, the features that interest us are maximum speed of the gesture and the duration of the gesture. Training set determine the weights of these features and when a new gesture is to be classified, the feature vector is computed for an input gesture. Associated with each gesture class is a linear evaluation function over the features and the class which gives maximum weight for the features is declared as the winner. The main advantage of this approach is that, one can add as many features as required to distinctly recognize the gesture. The same feature also becomes a disadvantage since the addition of features make the system more complex and computation even costly. This feature based approach fails when there is no global feature difference between two gestures, for example, to discriminate between a circle drawn in clockwise direction and another in anti-clockwise direction.

In spite of being simple and fast, these methods have proved to be efficient and accurate for single stroke touch gesture recognition. But when it comes to multi-stroke gestures, where there might be higher level of variations, they are not as successful. The fact of variation in shape, drawing order, number of strokes, importance of time-stamp properties etc. play a decisive role and needs to be taken care of.

2.2.2 Multiple Stroke Gesture Recognition

Similar to one of our intended processes for dividing a gesture into individual strokes and working on them separately, feature based recognition research [5, 29] also segment the gesture into smaller parts and later merge the output. In [29], it uses a two-stream Recurrent Neural Networks (RNN) architecture to model temporal dynamics and spatial configurations for skeleton based action recognition. It segments the whole body

into five parts i.e., two arms, two legs and one trunk. It implements and tests two different structures for the temporal stream: stacked RNN and hierarchical RNN. Whereas for the spatial model, they convert the spatial graph into a sequence of joints coordinates and apply rotation and scaling transformation to transform the 3D coordinates of skeletons during training. Then it uses a corresponding RNN to model the temporal movement of each body part based on its concatenated coordinates of joints at each time step. And later it concatenate the outputs of the RNN of different parts and adapt another RNN to model the movement of the whole body for temporal behaviour.

The use of rotation transformation, scaling transformation and shear transformation as data augmentation techniques used in this paper gives an exact idea about how our gestures need to be generalized.

Similarly in [5], it analyzes the shape of the gesture's motion trajectories and the temporal and spatial relations between trajectories in order to characterize a multi-touch gesture. This paper proposes a graph modeling approach to characterize the motion features and developed a graph based analysis and recognition system. Two graph models are introduced to extract the spatial and temporal features, named "Graph Modeling with Allen's Relations" and "Graph Modeling with Motion Based Features". These models segment the gesture's stroke into smaller units(vertices) and convert them into a graph with motion relation between sub-strokes(edges). A graph matching algorithm is used, combined to a single vector and units are combined to train a classifier.

Both of these above mentioned approaches give a general idea about separating a gesture into segments and work individually. Even though RNN or graph based modeling allows to exhibit dynamic temporal behavior for a time sequence, both of these methods does not take account of exact time-stamp information.

Napa Sae-Bae, Kowsar Ahmed, Katherine Isbister and Nasir Memon[25] presented a multi-touch gesture-based authentication technique which can be seen as a use case of multi-stroke gesture recognition task. They use the multi-touch surface to combine bio-metric techniques with gestural input. In their study, they defined a set of five-finger touch gestures, based upon classifying movement characteristics of the center of the palm and fingertips, and tested them. Instead of our topic of interest; CHnMM model, they used DTW, and built a classifier to recognize unique bio-metric gesture characteristics of an individual. They claimed to achieve 90% accuracy rate with single gestures, and had a significant improvement when multiple gestures were performed in sequence. Some gestures defined in this paper can be taken as gesture input for our experimentation process.

In [2], authors extended the features of $1 to handle multi-stroke gestures. The main idea in this paper was to link up the strokes together forming one long stroke. Two consecutive strokes are linked together by the help of pseudo-trajectory starting from end of first stroke to the beginning of following stroke. Therefore resulting single stroke can be manipulated using any single stroke gesture recognition approach. In real implementations, the order in which the strokes may be introduced is not known and may differ in different executions. Therefore, the order of concatenation is also unknown.

To handle this problem, $N recognizer simply tries all the possible combinations for the given strokes. The combinations differ in the direction of stroke drawing and order of stroke hence creating 2N possible combination for N strokes. This means there will be an explosion of combinations as the number of strokes increase. A simple example of possible permutations of strokes and order is shown in Figure 2.3.

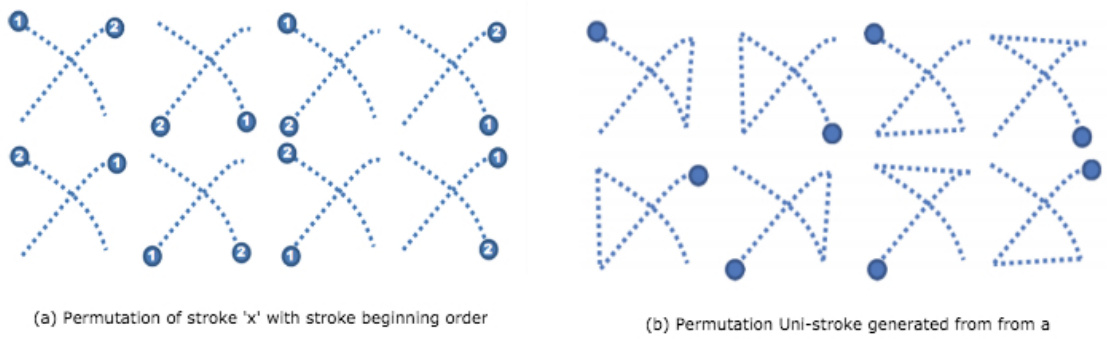


Figure 2.3: $N multi-stroke to uni-stroke transformation example

With a similar approach, our multi-stroke gesture can be converted into single stroke and dealt with single stroke recognizer. The only problem here is, we may have higher number of gestures and permuting is not the best option as it will increase the computation time. A proper way to find the best matching order needs to be devised. Also, we need to take care of an extra dimension: time associated with each stroke. Alongside the concatenation of spatial feature, time should also be meaningfully concatenated.

2.3 Previous Work

2.3.1 CHnMM

To master complex CHnMM, basics within the term has to be understood first. The most basic terminology that we need to understand CHnMM are stochastic processes, and are those processes having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely. For example the stock market and exchange rate fluctuations. On the next level, Markovian processes are the stochastic processes that satisfy Markovian property where a process is said to satisfy the Markovian property if its future can be predicted only based on the current state just as well as knowing the process's full history. Hidden Markov Model, for example selection of balls from an urn containing different color balls, is a statistical Markov model in which the system being modeled is assumed to be a Markovian process with hidden states. Therefore Hidden non-Markovian Models (HnMM) is a modeling paradigm based on Hidden Markov Models which extend the existing HMMs by changing the hidden model from a memoryless discrete-time Markov chain to a discrete stochastic model consisting of time dependent transition rates. This allows HnMM to analyze a system with hidden states only based on their interaction with the environment[16](More details about HnMM can be found in [17]).

HMMs are proved to achieve good results for gesture recognition in real-time with a low error rate[23]. At the same time, when it comes to the problem of representing time dependent processes, HMM are not as good as CHnMM[6].

Unlike HMM, HnMMs do not have the restriction of memory-less state transitions. If every state transition generates an output, then it is a CHnMM. CHnMM can represent stochastic systems that are continuous in time and may contain concurrent behavior with non-Markovian durations. According to[4], CHnMM has:

- a set of states S of size N
- a set of output symbols V of size M
- an initial probability vector $\Pi = (\Pi_1, \ldots, \Pi_N)$
- an NxN matrix A containing the state change behavior
- a set TR = tr_1, tr_2,..., tr_K of K transitions that define the model behavior.

Each transition tr_i is a tuple consisting of the following three elements:

- *dist* represents the continuous probability distribution that specifies the duration of the transition which causes a discrete state change on completion.
- *b(v)* is a function that returns the output probability of symbol v when the transition causes a state change. It is the semantic equivalent of the output probabilities in B for HMMs, but associated to transitions for CHnMMs instead of states as in HMMs.
- *aging* is a boolean value that determines if the time that the transition has been active for is saved (aging = true) or reset to 0 (aging = false) if there is a state change deactivating it caused by another transition, i.e. if the current active transition is interrupted by the triggering of another one.

All elements a_{ij} in A are either elements of TR or empty if no transition between states S_i and S_j exist. In conclusion, A CHnMM model Λ is defined as a tuple Λ = (S, V, A, TR, Π) that contains all previously described elements. In addition to the output symbol O_t of the observations, CHnMMs have a time property p_t, containing the point in time of symbol emission. This valuable time information is not needed for HMMs which only get it implicitly by a periodic symbol emission. An ordered sequence of these observations is called a trace O. A path Q contains a sequence of states q_t with associated timestamps that describe which state was active at the given time for a certain system.

2.3.2 A CHnMM Based Structure for Discriminating Spatio-temporal Movement Trajectories

In the beginning of this chapter, we discussed how HnMMs are applicable in the field of gesture recognition and can distinguish gestures in terms of the speed of execution[3]. In [6], a modeling approach for spatio-temporal movement trajectories using the special case of HnMM called CHnMM is presented. In this paper, "Spatio-Temporal Movement Model" (STMM) is implemented for single stroke gestures recognition and is the basis for this thesis. Hence, detail learning about the approach is important. The following paragraphs present the overview of developed CHnMM based structure for representing spatio-temporal movement trajectories.

This model approach employs a single model per movement trajectory describing the stochastic process that generates the trajectory of a consciously executed movement with a certain shape and certain temporal dynamics. This stochastic process is divided into its spatial and temporal stochastics. Splitting of stochastics facilitate the automatic model creation by utilizing the spatial information of the trajectories to define the CHnMM states and output symbols that together represent the spatial stochastics of the process, while the temporal stochastics are modelled by means of probability distributions in CHnMM transitions(*dist*). To represent the spatial stochastics of the process, *StrokeMap* was developed that describes the shape of the movement to be modeled. It defines areas that every trajectory path will reach successively and which are automatically determined from trajectory examples of the movement. These sequential areas*nAreas* segment a trajectory into phases, where each phase represents a state in CHnMM. In Figure 2.4, this concept is visualized with two exemplary trajectories that represent the movement or stochastic process respectively. The example trajectories are used to generate the *StrokeMap* first, which is also the basis for the layout of the CHnMM as each area of it corresponds to a state of the CHnMM. With the combination of *StrokeMap* and CHnMM and by using a left-to-right model topology, a supervised training is applied to estimate the probability distributions *dist* of the transitions of the CHnMM. For more details about the creation of spatial model and temporal model, please refer to [6]. However the important terms are also introduced here:

- *StrokeMap* : an ordered set of circular areas where a gesture stroke is expected to be after a certain distance of the gesture has been executed,
- *Area* : a circular area that represent the locations that every trajectory has to pass through successively,
- *minRadius* : a parameter determining the minimal radius of the created areas,
- *toleranceFactor* : a parameter determining the tolerance radius(by multiplying the factor with the original circle radius),
- *nArea* : a parameter stating the number of areas that will be formed in the trajectory path,

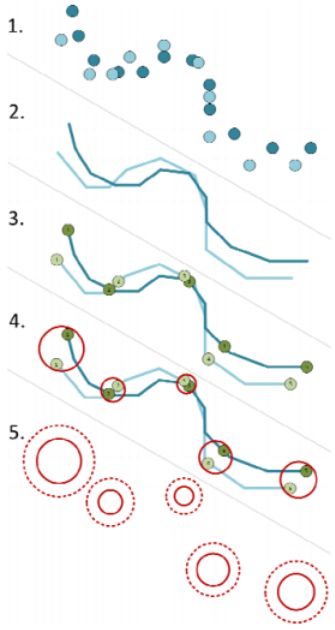


Figure 2.4: Visual StrokeMap generation process (fixed sampling)[6]

- *hitProbability* : specifies the probability that the *Ai-Hit* symbol is generated by a trajectory, where *Ai-Hit* is generated if the sampled point in trajectory lies within *minRadius* of circular area and *Ai-Tol* is generated if the point in trajectory lies within tolerance radius.
- *dist* : probability distribution that describes the time needed to travel from area A_i to area A_{i+1}

In [7], an online movement trajectory verification using this CHnMM is evaluated. The system is tested on signatures, doodles and pseudo-signatures for its verification performance. As we are also going to perform verification experiment in our research, the result from this paper were also of interest to us.

3. Defining Multi-stroke Gesture Recognition Models

In this chapter, we present our effort for multi-touch gesture recognition. In 3.1 we will outline our gesture and gesture set and finally take gesture data from our volunteers. We will pre-process the raw data and make it more meaningful to our desired system in section 3.2. Later in 3.3, two models will be introduced to extract the intra/inter stroke spatial and temporal features from the gestures.

3.1 Definition and Collection of Gestures

We found that there were not enough multi-touch gesture dataset available that could be used to create our intended system and for tests as well. Therefore we created our own gesture dataset containing different types of gestures varying in terms of number of strokes and speed of execution. In this section we will define the meaning of gesture in the scope of this thesis and also introduce our gesture set.

3.1.1 Defining Constraints of Gestures

A broad meaning of gesture is: a movement of part of the body, especially the hand or the head, to express an idea or meaning. To work with the CHnMM model, we had to constraint the meaning of gesture in this paper. So, we defined a gesture as just the hand movement in touch enabled devices. The following constraints define the gesture in the way we intend to recognize it.

1. A gesture can be formed by combining any number of strokes/fingers,
2. Fingers can be from different hands as well,
3. A stroke of gestures can start and end independent of each other,

4. Start of the gesture is when any finger touches the screen/touch pad,
5. Order of finger touching the touch enabled area for different stroke is vital,
6. Gestures strokes should be continuous (once the finger leaves screen/touch pad, finger down will be counted as new stroke),
7. In two separate executions of a gesture, any finger could be used to draw same stroke (for example, if you use index finger to draw a stroke in first execution, you are not bound to use index finger but you can use any other fingers for drawing that stroke in the next executions),
8. Gesture is said to be complete when user submits the execution to the system.

Any gesture that does not follow these constraints is not a valid gesture for this thesis.

3.1.2 Defining Gesture Sets

To experiment with varying number of fingers and hand movements, a gesture set consisting of fourteen different gestures were introduced. However, some gestures are inspired from other research and are cited for reference. Also, gestures have been created to evaluate the claimed main ability of the recognition system to discriminate trajectories of similar shape with different temporal dynamics. The gestures and their way of executing is defined below:

1. Close: All five fingers move toward the palm's center, in a closing motion[25].
2. FTCW: Thumb is fixed, other fingertips rotate around it in a clockwise direction [25]
3. Bolt fast: Excluding thumb finger, four fingers move from top to bottom forming an electric bolt like symbol in fast motion.
4. Bolt slow: Excluding thumb finger, four fingers move from top to bottom forming an electric bolt like symbol in slow motion.
5. Circle Similar fast: Using index fingers from two hands, two circles are drawn in the same direction(clockwise) in fast motion.
6. Circle Similar slow: Using index fingers from two hands, two circles are drawn in the same direction(clockwise) in slow motion.
7. Circle different fast: Using index fingers from two hands, two circles are drawn in different directions one clockwise and another anti clockwise in fast motion.
8. Circle different slow: Using index fingers from two hands, two circles are drawn in different directions one clockwise and another anti clockwise in slow motion.

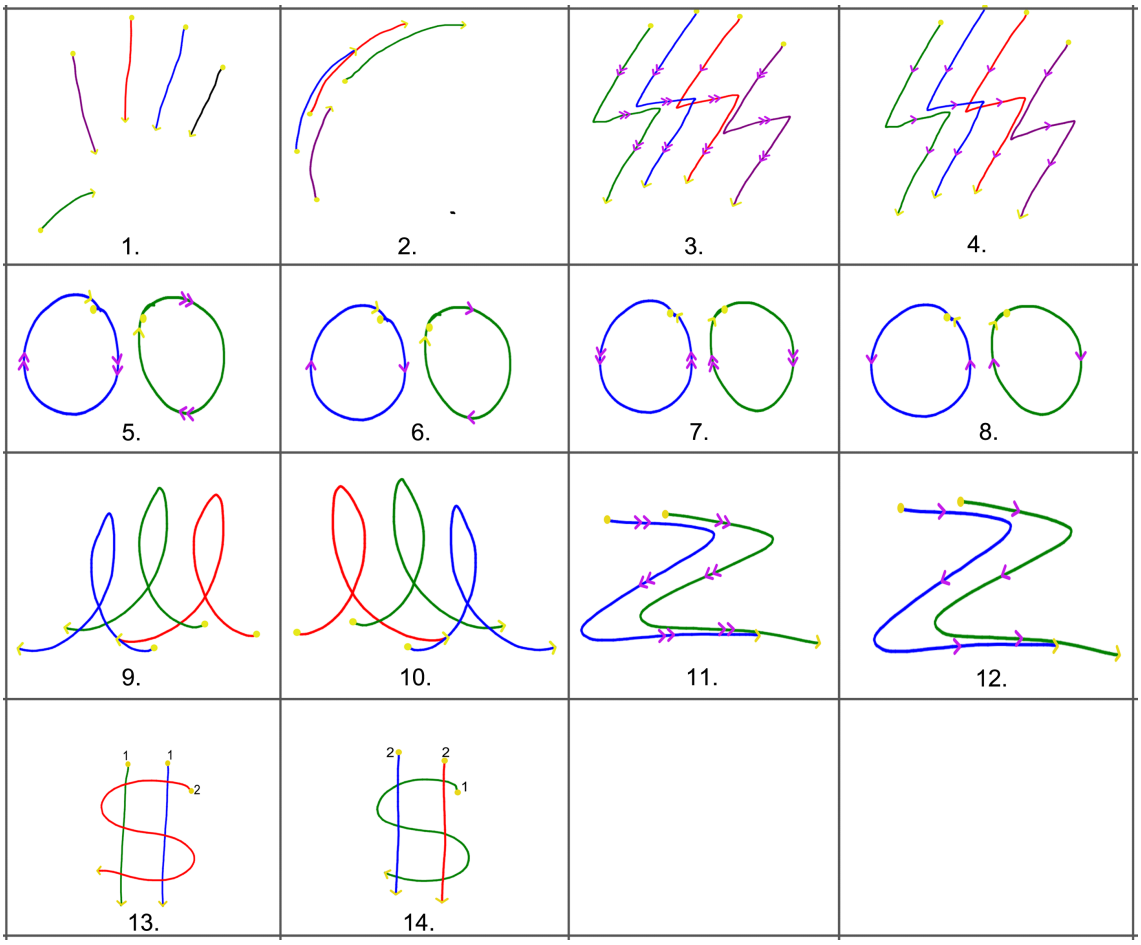


Figure 3.1: Gesture set

9. I right-left: using three fingers, cursive ”i” like shape is drawn from right to left of the screen.

10. I left-right: using three fingers, cursive ”i” like shape is drawn from left to right of the screen.

11. Erase fast: move two fingers making a ”Z” like shape in fast motion.

12. Erase slow: move two fingers making a ”Z” like shape in slow motion.

13. Dollar sign($) lines first: create a dollar sign starting from vertical lines and then shape S.

14. Dollar sign($) lines second: create a dollar sign starting from S shape and then moving to vertical lines.

In the Figure 3.1, yellow dots indicate the start of the stroke and yellow arrow indicates the direction of stroke execution. Speed of the executions of strokes are indicated by

purple arrow where the speed information is relevant(single arrow indicates slow and double arrow indicate fast execution speed). Also, for sub-figures 13 and 14, execution order are stated in ascending order.

This heterogeneity of gestures in terms of number of strokes will help to test the system's ability to deal with the varying number of strokes. Likewise, the heterogeneity of gestures in terms of execution speed(for example, *bolt fast and bolt slow, erase fast and erase slow* will test the CHnMM's ability to distinguish gestures based on temporal differences. However, before we can accomplish this, we need to collect gestures and create a dataset.

3.1.3 Developing a Web Interface to Create Datasets

Freely available gesture/trajectory libraries were not able to fulfill the requirement of our test specification in previous section. Hence, to create a proper data set library with multi-stroke gestures, we created a web application using the Ruby on Rails web development platform. The application could be used from any multi-touch enabled device that could browse over the Internet. In the front end, jQuery library jQMulti-Touch development framework [20] was used to enhance the process. Whereas, in the back end, object-relational database management system (ORDBMS) PostgreSQL was used to store the user input gestures. Figure 3.2 visualizes the steps involved in taking gestures from logging in to plotting user data.

As we have a platform that can store and backup the gesture set required for our research, we can now collect data from our source.

3.1.4 Taking Gestures from Volunteers

In a monitored environment, we asked 11 volunteers to sign-up to our web application so that we could uniquely identify the gesture set from a particular volunteer. Volunteers were briefly trained to use the platform. After signing in to the application, users could see a touchable area with grids on it. For the ease of volunteers, examples were provided on the screen for each gesture as in Figure 3.2 sub-image 3. Then user were asked to select a gesture name from the drop-down at the right side of the screen. Users had to draw the selected gesture in the touchable area and had options to save it to the database or clear and redraw(e.g., Figure 3.2 sub-image 4). Volunteers were advised to perform and save a gesture 20 times for each gesture in the drop-down menu(e.g., result of 20 executions plotted in Figure 3.2 sub-image 5). Regular instruction regarding speed and direction of execution were provided throughout the process to ensure the consistency of the gesture sets. An average of half an hour was invested by every volunteer to complete their gesture set. We recorded roughly(three participants had 15 executions per gesture) 14 gestures* 20 executions *11 volunteers = 3080 gesture executions. For this task, all volunteers used the same device; Samsung Galaxy Tab S2.

In this dataset, all executions of one gesture from a user were supposed to have user specific properties that help to discriminate their gestures from the rest of their gestures along with gestures from other users. However, to get to this expected level, pre-processing of data is required which we will cover in the following section.

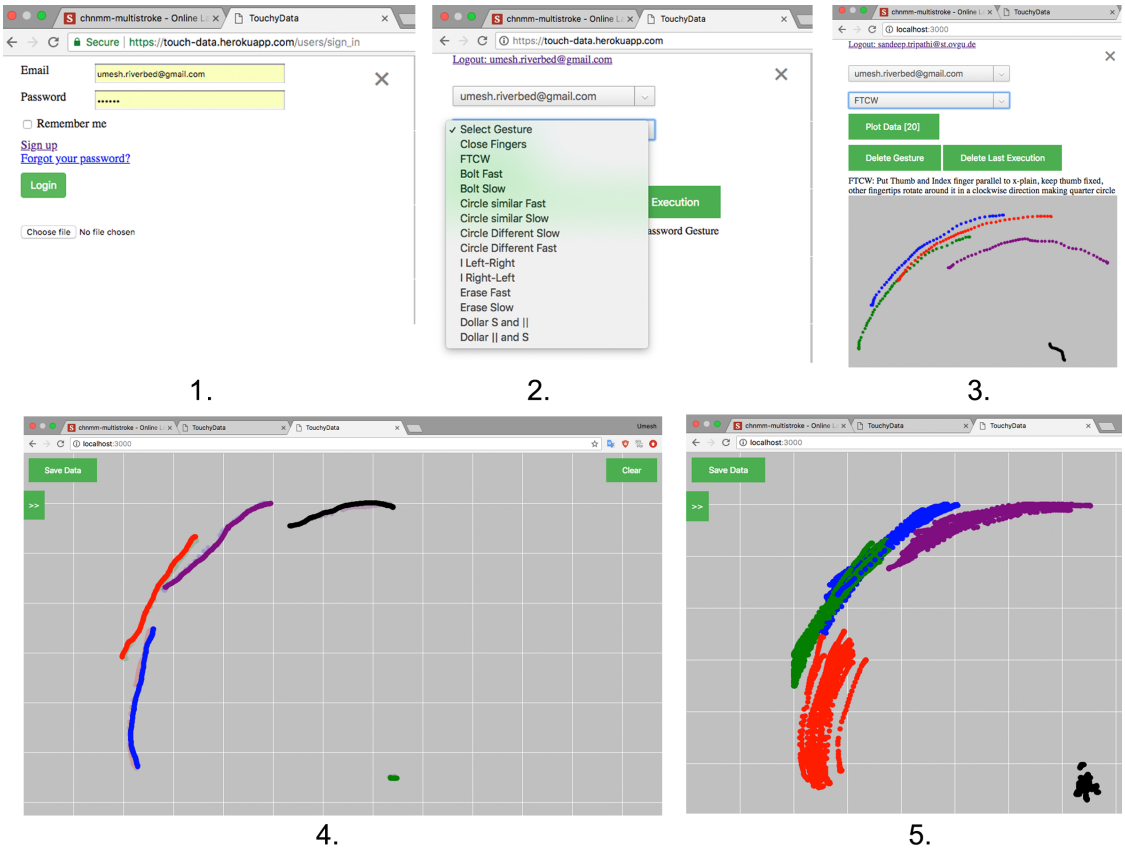


Figure 3.2: Web interface for gesture input

3.2 Data Preparation

Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors. Raw data is likely to contain many errors. Therefore, certain pre-processing has to be done so that it fits our desired criteria for further processing. In this section we perform data preprocessing. Mainly we will go through data transformation and data reduction steps.

3.2.1 Normalization of Time and Co-ordinate Components

In our data collection process, we did not constrain users from drawing gestures from a fixed point in the co-ordinate plane. Therefore two executions of same gesture can vary in terms of position in the screen. To make sure all gesture executions have more or less overlapping position at the end, we had to normalize the co-ordinate values. Similarly, the time component should also be normalized in terms of first touch of the gesture.

In the front end of the gesture input web application, a timer was started when first touch event was triggered and with each following touch entry, timer value at that instance was assigned to that touch registered. Hence, no matter how many finger strokes were initiated, all of them had a time-stamp with respect to the first touch

registered. This feature helped to connect different strokes of a gesture by facilitating a continuous probability distribution function.

After a complete gesture was submitted to save, in Rails end, the co-ordinates were subjected to normalization. One of the way to achieve that could be that, co-ordinates are normalized based on the first registered touch. But, if we look at the two executions of a multi-touch gesture, there might be always the case that the order of fingers touching the screen is different with the slightest of time difference. Moreover, this would result in a lot of negative co-ordinates if the first touch registered was at the right end of the screen. Hence, it would not be possible to generate a robust CHnMM model from those strokes. So a different approach was coined. The smallest x and y co-ordinate

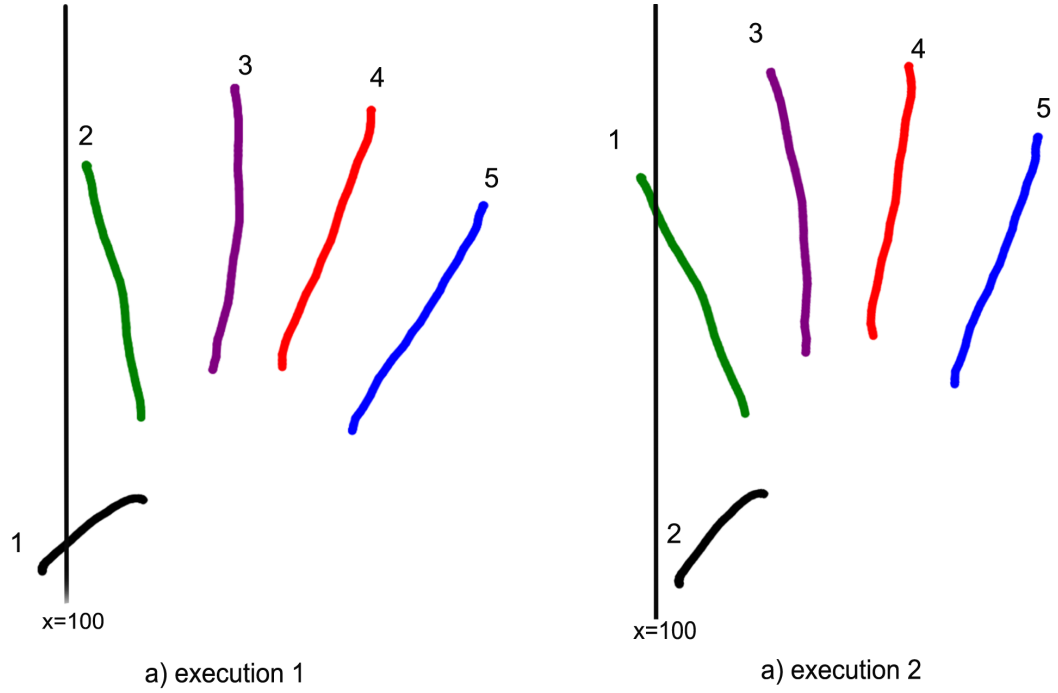


Figure 3.3: Stroke sequence number with reference to x-plane

values were extracted from gesture input, and each co-ordinate in gesture subtracted the smallest x and y values from their x and y co-ordinate values respectively. Hence two different executions of the same gesture would overlap even if they were drawn at different position in the screen. It allows to abstract a gesture identity from the specifics of the visual input, like relative positions in the screen, and is known as Translation Invariance.

As each stroke is going to be saved as a single entry in the database, the strokes need a stroke sequence number to define the position of stroke in gesture. Later, the strokes with same stroke sequence number will form a model. At the first iteration of development, the finger strokes were given an unique id based on their position in x-plane. The stroke with the smallest x co-ordinate value got the id 1 and with the

highest x co-ordinate value got largest id. This would work fine with the gestures that would not have conflict of x plane values, for example bolt or circle in our gesture set. But when it comes to the stroke which may start parallel to x-plane, assigned sequence number were not correct as seen in Figure 3.3. Hence a new algorithm for assigning sequence number to the gesture strokes was formulated.

To make sure that every finger stroke gets a correct sequence number, each stroke data was checked against the first gesture data and then, the most relevant stroke sequence number was assigned. The relevancy was calculated using the euclidean distance measure. Finally for each finger stroke, a dictionary was prepared to insert into the postgresql database.

Now we have resolved the inconsistencies in the data. Data with different spatial representations are put together and conflicts of stroke sequence within the gesture are resolved. Hence, data is normalized and generalized.

3.2.2 Dividing Data into Training Set and Test Set

While collecting gesture data from the volunteers, at the earlier executions of every gestures, volunteers needed time to get used to the process. Slowly after few executions, adaptation to gestures was seen and the speed of execution also changed. Which means, selecting the first half or second half of execution for training would not be representative of the data set as a whole. n Random executions could have been taken for training and the remaining for test but then there would have been overhead of tracking which executions were grouped where. Hence we decided to take alternate executions of gestures for training and test sets. Over fitting was not seen as two consecutive executions are also different in terms of trajectory and time components. At the end, 50% of gestures execution were taken for training the model and the remaining 50% for test.

3.3 Model Design

In this section, the spatial model and CHnMM collectively are called as STMM unless separately stated. Terms like *nArea, StrokeMap, areaPointDistance, tolerenceFactor, minRadius, hitProbability* are same for both of the approaches presented here and can be referenced to section 2.4.

Two variations of the model design are discussed and implemented here. These models differ in terms of number of STMMs generated. Suppose we have a gesture with N finger strokes, then the first modeling approach creates N number of STMMs, hence called as modeling approach with "single STMM per stroke". The second modeling approach generates a single STMM for every gesture, regardless of the number of strokes in the gesture. Therefore, it is termed here as "single STMM per gesture" approach.

3.3.1 Single STMM per Stroke

We already discussed in section 3.2 about the normalization of time and co-ordinate components and numbering of the strokes of a gesture. This means we have the information of which trajectories are going to form a STMM for a finger stroke. Also, as described in the state of art single trajectory CHnMM[6], model approach employs a single STMM per movement trajectory. Hence we can already create a STMM for separate finger strokes of a gesture. Therefore for a gesture with n-strokes, there will be n models in total. In the Figure 3.4, spatial model generation process is visualized. In the example figure, we can see that there is no link between two strokes of a gesture. Segment (a) in the Figure 3.4 shows two example gesture executions distinguished by colors purple and green. Whereas dark and light variation of colors represent different strokes of a gesture. Segment (b) shows a continuous trajectory of the movement and two finger strokes are separated by the dotted line to show that they are handled separately. A fixed number of interpolation within the strokes can be seen in segment (c). Segment (d) visualizes the minimum circle containing all the corresponding points of the example trajectory strokes. A dotted circle with *tolerenceRadius* is added at the last. At the end we can see two spatial models are generated, one per finger stroke.

A system trained with N STMMs, one per stroke, in the recognition phase will result in N individual recognition result. Therefore, those separate results have to be merged together and an ultimate recognition result must be presented. We applied three different merge techniques that ultimately helped to find the winner. For all techniques, the recognition result was only calculated if all strokes of a gesture were individually recognized as a particular gesture part. This process is visualized in Figure 3.5. To elaborate, there is an example gesture with three strokes which needs to be recognized. For each stroke, the recognition system outputs several possibilities of being a part of a certain gesture. In the figure, we can see there are competing gestures(gesture id 2 from user 1, user 2 and user 3) for individual strokes. If one of the strokes form the competing gestures have missing/zero recognition probability, then the gesture is already counted as an invalid competitor. In our example illustration, the second stroke does not have u1_ges2(user id 1, gesture id 2) as competitor even though the first and third strokes have. Therefore u1_ges2 is filtered out from the list of competing gestures already. If all strokes have non-zero recognition probability, then the stroke probabilities are forwarded for gesture recognition result merging. The merge and result techniques that we experimented are the maximum average of score winner, maximum multiplication result winner and highest score winner. These techniques are discussed in detail in section 4.1.

3.3.2 Single STMM per Gesture

An alternative to single model per stroke is single model per gesture. It is rational that, creating one model per gesture will reduce the overhead of combining the results of separate strokes into one. But the challenges of combining the strokes as a gesture before the creation of the model still remains. In this approach, strokes of a gesture are

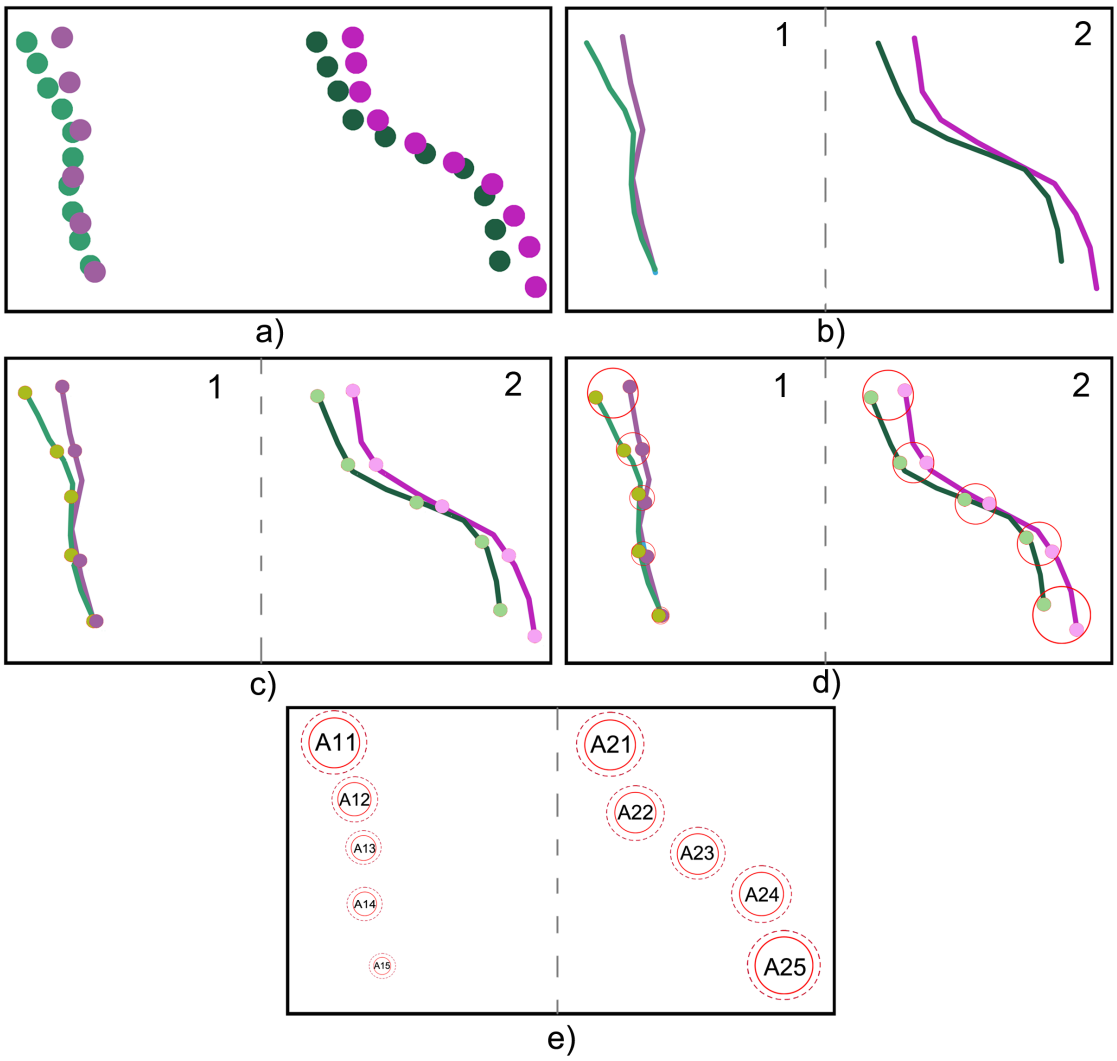


Figure 3.4: N Number of StrokeMap generation: Single model per stroke

linked by a virtual connecting line running from the last point of n^{th} stroke to the first point of $(n+1)^{th}$ stroke.

This model approach employs a single STMM per gesture and as mentioned previously, consists of spatial and temporal models. The model describes the stochastic process that generates the fixed number of trajectories with homogeneous shape and temporal dynamics. Similar to the single STMM approach, the stochastic process is split into its spatial and temporal stochastics. Spatial stochastics is represented with the *StrokeMap*. Each area in *StrokeMap* segments a trajectory into phases that maps to a corresponding state in CHnMM. Likewise, the temporal stochastics are modelled by means of probability distributions in CHnMM transitions *(dist)*.

The task of combining the strokes as a gesture is facilitated by the known order of strokes. The trajectory information of all the strokes are tailed to first stroke in sequence

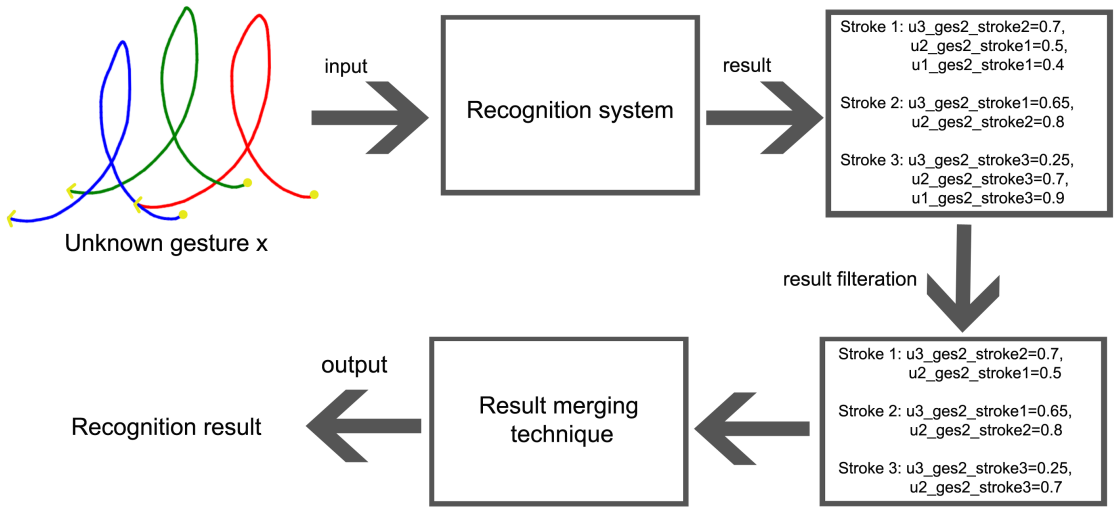


Figure 3.5: Single STMM per stroke: Recognition process

where every point of the trajectory has its stroke id and time information re-configured with respect to the first point of first stroke. Hence maintaining spatial and temporal dynamics of the gesture as a whole. Therefore n-trajectories are converted to one trajectory. Interpolating on these final trajectory points for fixed number of areas would create a link and successively create areas between the end of n^{th} finger stroke and start of $(n+1)^{th}$ finger stroke as seen on Figure 3.7. To avoid *StrokeMap* areas in-between of two consecutive finger data, interpolation is carried out within the trajectory points of each stroke and fixed number of areas are created for those interpolated points.

Alongside, as the strokes are being merged, the time dimension is also normalized with respect to the first point of first trajectory. Hence, all the points in $(n+1)^{th}$ stroke added time stamp from last point of nth stroke to their time value. This at the end, results to a long trajectory which can be trained using single trajectory CHnMM and also carried information of separate strokes. The cost of using this technique is the carryover of time error from previous strokes. For example, the time inconsistency in first stroke will add the time in all the following strokes even-though the later strokes were consistent within themselves.The visual representation of steps involved can be seen in Figure 3.6.

Segment (a) in the Figure 3.6 shows two example gestures executions distinguished by colors purple and green. Whereas dark and light variation of colors represent different strokes of a gesture. Dotted lines running in between the strokes show the virtual link between the strokes of a gesture. Segment (b) shows a continuous trajectory of the movement. A fixed number interpolation within the strokes can be seen in segment (c). Segment (d) visualizes the minimum circle containing all the corresponding points of the example trajectory strokes. A dotted circle with *tolerenceRadius* is added at the last.

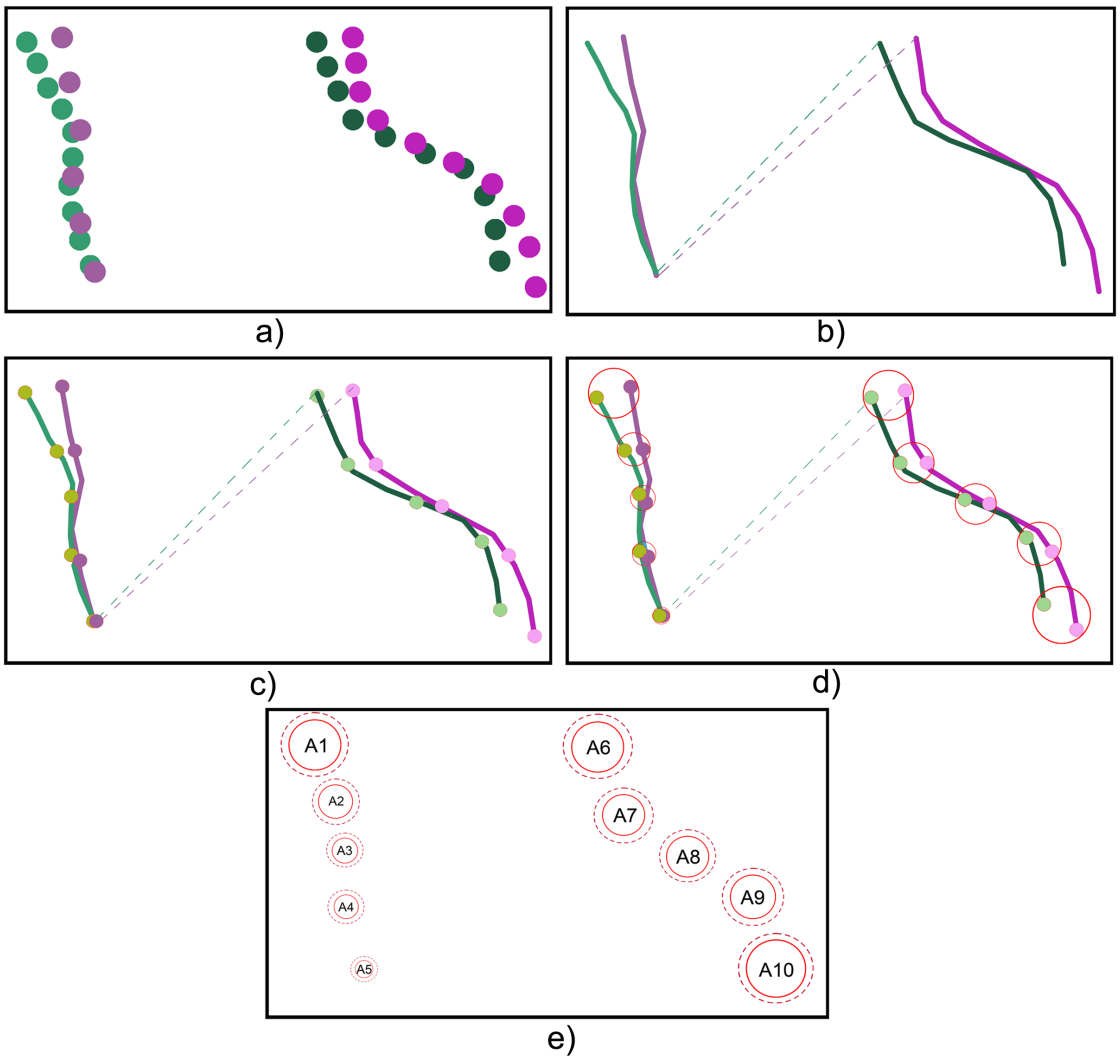


Figure 3.6: Single strokeMap generation: Single STMM per gesture

In the recognition phase, we already have the gesture models and we are looking for the best matching model for candidate execution of a gesture. As the order of strokes is undefined, a candidate gesture with n strokes can have at most n! possible combinations of strokes that need to be checked for. Therefore the complexity of recognizing the gesture increases with the increment of the number of strokes. To reduce the complexity of recognition process, incremental spatial matching is employed. At first, every stroke checks for shape match with the already learned gesture models. Matching strokes are then assigned with stroke number and then recursively, other strokes are added and checked for match improvement. With every improvement, strokes will get their best fitting position. At the end, time values are adjusted according to the match sequence and can be sent for recognition task.

As a result of recognition, it is possible that two or more combinations of strokes are competing for different gesture model matches with consecutive probabilities. In that

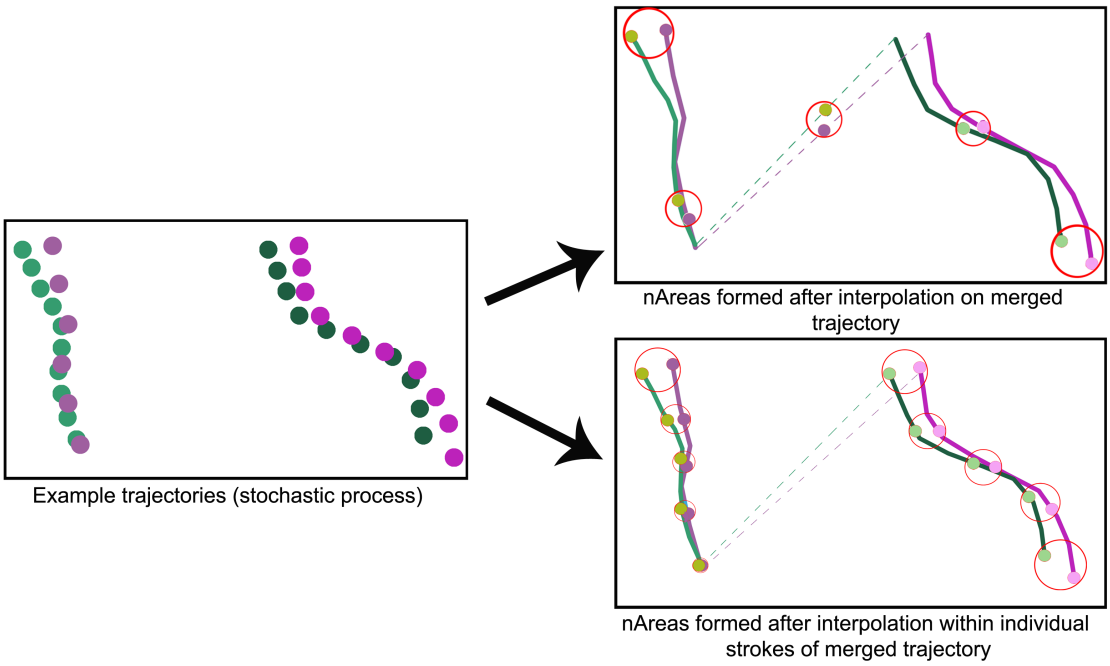
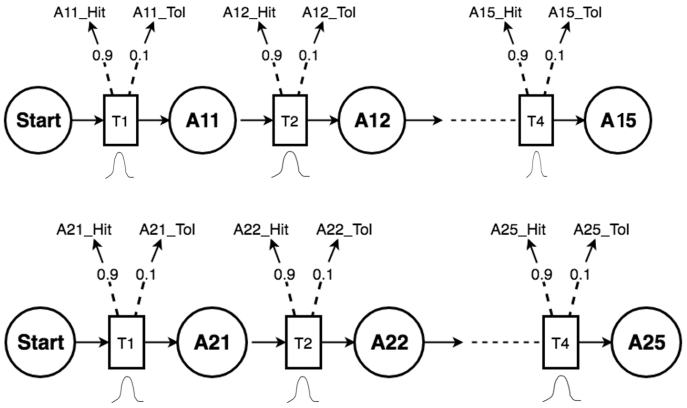


Figure 3.7: nAreas generated with fixed number of nAreas = 5

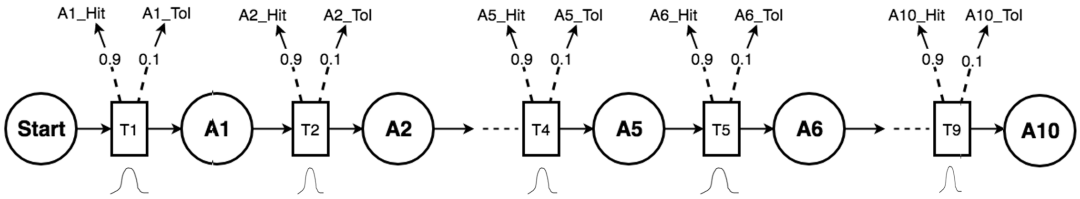
case, the model with the highest probability wins. The temporal models generated from these two approaches are illustrated in Figure 3.8. In single STMM per stroke approach, 2 temporal models are generated for the given example in Figure 3.4. The system parameter fixed number area, *nArea*, decides the number of states in each ChnMM. Therefore for each strokes of a gesture in Figure 3.4, CHnMM with 5 states is generated. Whereas, only one temporal model is generated for the single STMM per gesture approach. Here, the system parameter *nArea* times number of strokes in gesture defines the number of states in CHnMM. Therefore for our example in Figure 3.6, a temporal model with 10 states is created.

In the beginning of this chapter, we defined some constraints that applies to a valid gesture for this research. Fulfilling these constraints, we defined a gesture set and finally created a gesture dataset. The primary focus of this chapter was to define an applicable gesture recognition model. Two gesture recognition model approaches, *Single STMM per Stroke* and *Single STMM per Gesture*, were presented and with this model, we were finally able to get gesture recognition result.

Being able to use STMM for gesture recognition process and to get recognition output is one major step towards our goal, which is to find out if the time dimension can be added to multi-stroke gesture recognition with the help of CHnMM or not. But, before we can answer this, we have to test the credibility of the system. In the following chapter, we will conduct appropriate experiments and based on the result of these experiments, we will be able to find out if our goal has been truly achieved. Also in Chapter 4, we will see which approach defined here will perform better in terms of recognition and verification tasks.



(a). Single STMM per stroke



(b). Single STMM per gesture

Figure 3.8: Single STMM per stroke vs Single STMM per gesture: Temporal model(CHnMM) generated

4. Experiments and Evaluation

Any bio-metric system can work in two distinct modes, which must be distinguished during evaluation: verification and recognition. Suppose an university student wants to use his/her library card. In verification mode, student presents your identity and the bio-metric device verifies that the identity matches (student present his/her ID with face image, the system verifies student's real face and the image on ID).

Quite opposite in recognition mode, no assumption of identity is made in the beginning and comparison to all templates has to be made. Suppose a CCTV(Closed Circuit Television) captured a crime scene and police are trying to reveal the convicted person's identity using the footage. The police authority identifies the convict using the face recognition application. No assumption is made, they just use the acquired face image and compare it to all the faces in their database.

Both of these modes rely on feature similarity, but similarity is a relative measure. Therefore, as per the requirement, one needs to define a threshold of acceptance. For example in gesture based authentication, the system will use this threshold value to accept or reject the access attempt. Access to the system is granted only, if the score for a trained person (recognition) or the person that the pattern is verified against (verification) is higher than a certain threshold. In the best possible case, a true user's scores (scores of patterns from persons known by the system) is higher than the scores of impostors, therefore the single threshold can separate true users and impostors. However, this assumption does not hold true for real bio-metric applications. In some cases impostor patterns can have scores that are higher than the scores of some true user's patterns and opposite. Therefore, error is nearly certain.

In our experiments, the result is influenced by the parameter configuration of the system. The configurations varied in terms of type of distribution estimator(*dist*), number of areas formed per stroke(*nAreas*) and *toleranceFactor*. The distribution estimator(*dist*) describes the time needed to travel from area A_i to area A_{i+1} and to travel a certain distance. Whereas, *toleranceFactor* is the measure to determine a tolerance radius of

the Areas by multiplying the factor with the original circle radius. Where original circle is formed as a minimal circle that contains all the points of a set S_i containing sample points from every example trajectory in the corresponding region.

The distribution estimator(*dist* could be one of the normal and naive uniform distribution estimators, *nAreas* could be one of the numbers from set 10,15,20 and *toleranceFactor* could be one of the numbers from set 1.1,1.3,1.5,1.7,1.9,2.1,2.3,2.5. The combination of all these parameters resulted in 48 different parameter settings.

From this discussion above, we learned that we need a dataset to carry out the desired experiments. As the model development process also needed the experiment datasets, we have already defined them in section 3.1 and 3.2. In this chapter we see how recognition and verification experiments were carried out, we compare the results of different models approaches and results of $N recognizer. We also will evaluate the result of the experiments and try to figure out which setting of parameters resulted in a better output.

4.1 Single STMM per Stroke: Combining Results

A system trained with N STMMs, one per stroke, in recognition phase will result in N individual recognition result. Therefore to get the recognition result of a gesture as a whole, those separate results have to be merged together. We chose three different merge techniques given below. An example result of recognition from Figure 3.5 is shown in Figure 4.1.

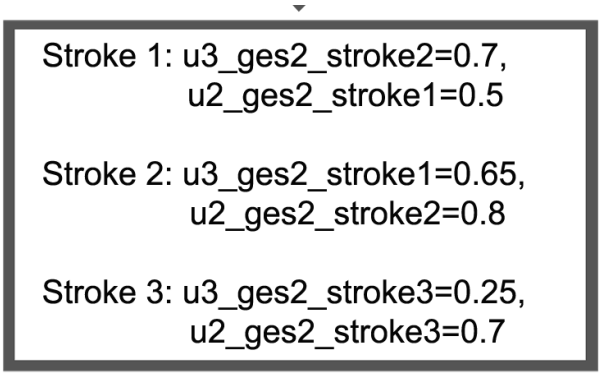


Figure 4.1: Result of recognition for example gesture from Figure 3.5

4.1.1 Maximum Average of Scores Winner

In this technique, all the individual stroke resulting probabilities of a candidate gesture are summed up, and the average is stored as a resulting score for that candidate gesture. When scores for all the candidate gestures are calculated, the one with the highest score wins. For example, in Figure 4.1, the sum of scores for candidate gestures u2_ges2 and u3_ges2 are 2.2 and 1.3 respectively. With this merge and result technique, u2_ges2 is the winner.

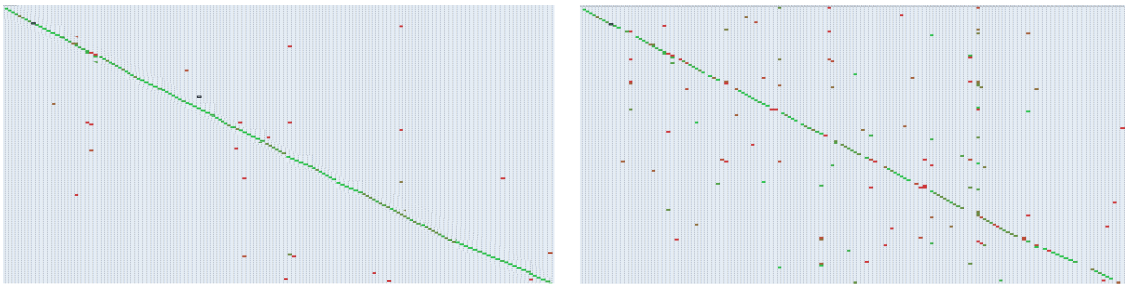
4.1.2 Maximum Multiplication Result Winner

In this technique, all the individual stroke results of a candidate gesture are multiplied, and is stored as the resulting score for that candidate gesture. When the scores for all the candidate gestures are calculated, the one with the highest score wins. For example, in Figure 4.1, multiplication of scores for candidate gestures u2_ges2 and u3_ges2 are 0.7*0.8*0.7=0.392 and 0.7*0.65*0.25=0.113 respectively.

4.1.3 Highest Score Winner

In this technique, for each stroke, the candidate with the highest probability wins that stroke. At the end, whichever candidate has the highest number of wins, is the final winner. From our example in Figure 3.5, using this technique, u2_ges2 has two wins and u3_ges2 has one win. Therefore the gesture is recognized as u2_ges2.

With this technique, if the score for two or more candidate gestures are equal, the ultimate winner is not clear. Therefore, an extra effort has to be introduced to distinguish the winner. The combination of these techniques can be employed at an extra level of decision process but the theoretical possibility of two candidate gestures having the same score in the next level always exists. In this single STMM per stroke approach, finding a reliable merging technique is an area where further work can flourish. Maximum weighted sum technique is another possible way of merging but, since our strokes do not have weights associated with them, this technique is not applicable here. An example confusion matrix of results from our recognition system using the maximum average of score winner and maximum multiplication result winner are shown in Figure 4.2. Both of the results are for the same parameter settings(*nArea = 10, toleranceFactor = 1.7, dist = uniform*). In the figure, all marks other than in diagonals are errors. Also, the discontinuity in diagonal represents the error. Looking at these two figures, we can clearly say that, the maximum average of scores winner performs better. Using this technique, the best recognition rate of 85.8% was achieved for our dataset.



(a). Maximum average of score winner (b). Maximum multiplication result winner

Figure 4.2: Result of recognition for our dataset using different merging techniques

4.2 Gesture Recognition

For this experiment, 50% of all(8 users*14 gestures*20 executions + 3 users*14 gestures*15 executions = 2870 executions) gesture executions are used to train the STMM. For the remaining 50% of the execution, without gesture and user information, the system is asked to return the best fitting STMM for that execution. The system should correctly recognize its owner and also recognize the exact gesture that the execution belongs to. Parameter variations and cross validation are done.

For the evaluation of our recognition experiment, first of all we can categorize recognition attempts into recognized and non-recognized gestures according to the results. With this classification, the *Non Recognition Rate*, can be can be calculated as:

NRR = *Number of non-recognized gestures / Total number of recognition attempts*
In our case:
Total number of recognition attempts = 1435 gesture executions

Likewise, all recognized gestures can be further classified as *False Recognized* and *Correctly Recognized*. Where, false recognition is the condition where a gesture execution is identified as someone else's gesture to which it does not belong to or is identified as different gesture from same person. And, correct recognition is the condition where a gesture execution is identified as the user's to which it really belongs to. Hence, *False Recognition Rate* and *True Recognition Rate*, can be obtained from this information.

False Recognition Rate = Number of false recognized gestures / Total number of recognition attempts

True Recognition Rate = Number of correctly recognized gestures / Total number of recognition attempts

The accuracy of a system is given by the *True Recognition Rate* whereas the reliability of the system(precision) can be derived as:

Precision = Number of correctly recognized gestures / Total number of correctly and falsely recognized gestures

The result of this experiment for single STMM per gesture approach and single STMM per stroke approach are shown by Figure 4.3 - Figure 4.9. FRR for both approaches show similar characteristics. For both, FRR increases with the increase in tolerance factor. The increment in FRR is steep for single STMM per stroke approach reaching up to 12%, whereas increment is steady and reaches a maximum of 7% for single STMM per gesture. However, normal distribution performs better for both approaches. The parameter setting *nArea = 10*, *dist = normal* and single STMM per gesture(Figure 4.3, shows a peculiar nature of the system. With the increase of *toleranceFactor*, it forms a curve similar to sine curve.

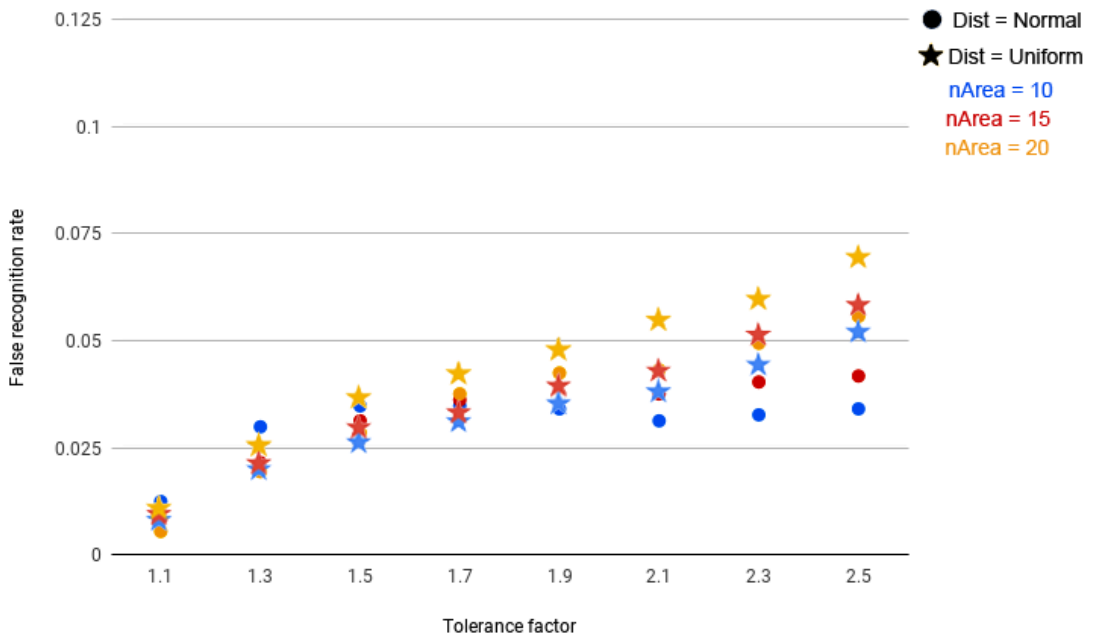


Figure 4.3: *False Recognition Rate*: Single STMM per gesture

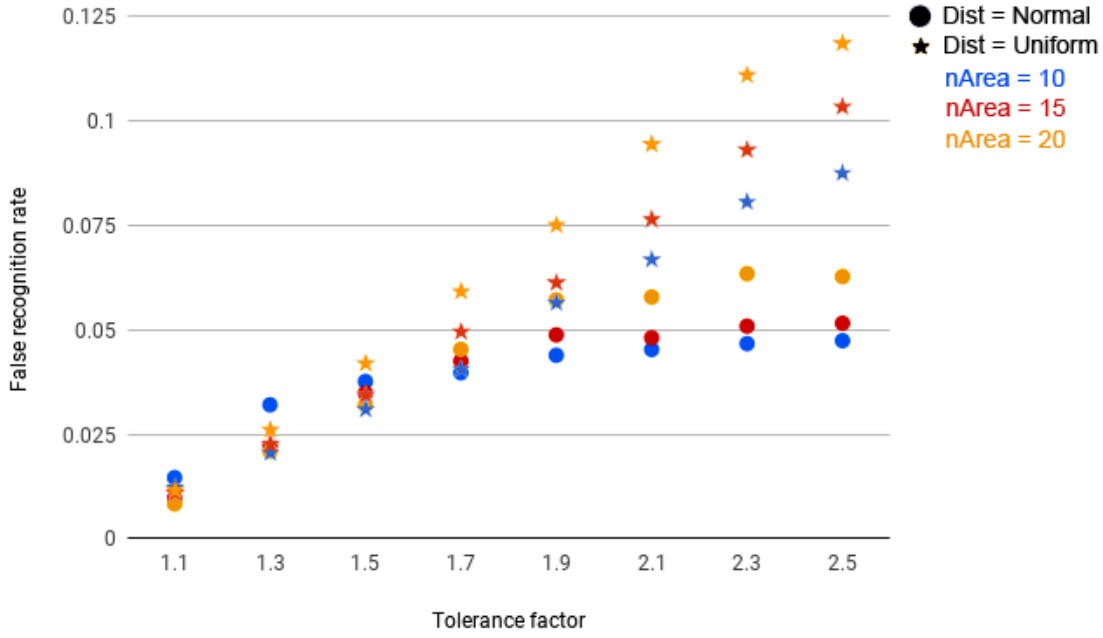


Figure 4.4: *False Recognition Rate*: Single STMM per stroke

In Figure 4.5 and Figure 4.6, the *True Recognition Rate* linearly increases with the increase in tolerance factor while using normal distribution compared to uniform dis-

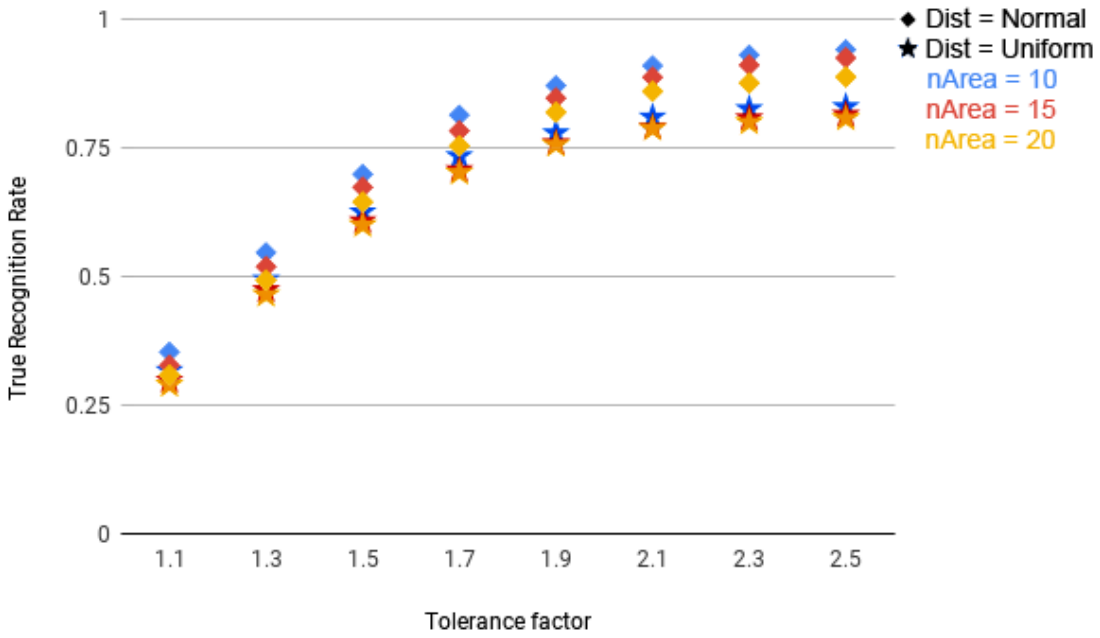


Figure 4.5: *True Recognition Rate*: Single STMM per gesture

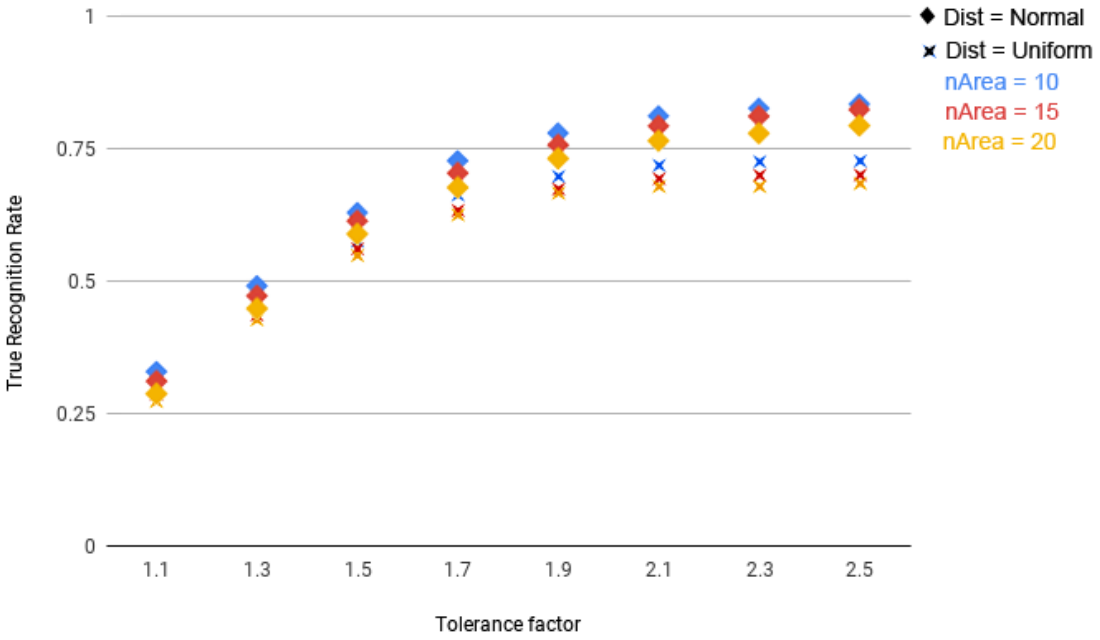


Figure 4.6: *True Recognition Rate*: Single STMM per stroke

tribution. This is because of the fact that normal distribution has infinite support and has a continuous bell shaped density function where as uniform distribution has limited

support and has a piecewise constant density function. Also, we are taking the highest probability model as a result model and the probability of a true user's gestures also increase when tolerance is increased. Therefore, the number of possible candidate models increase but the true model's probability is the highest.

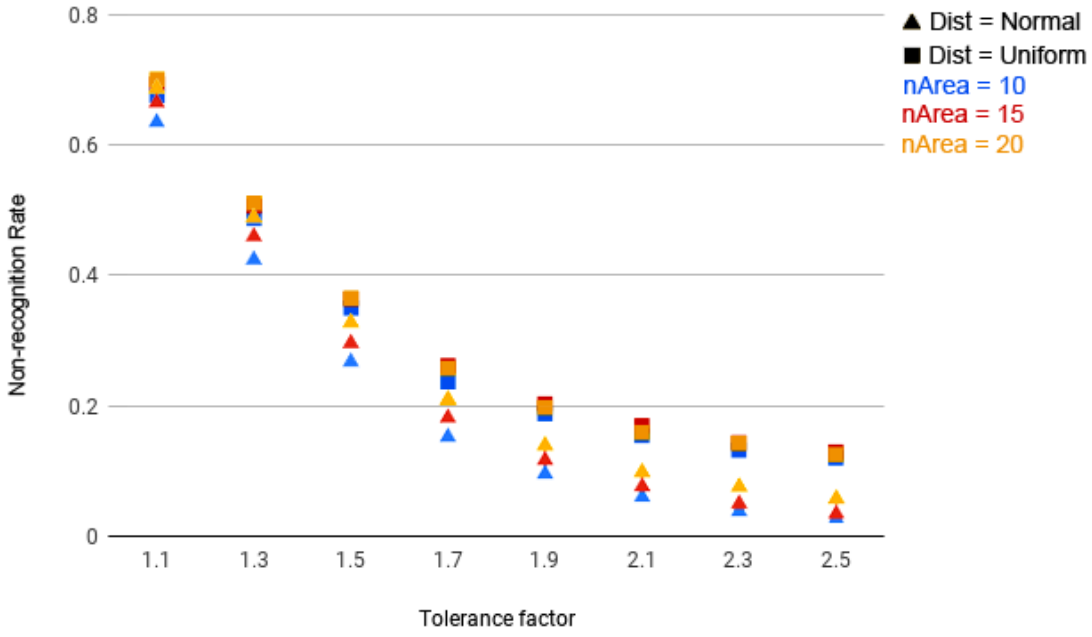


Figure 4.7: *Non-recognition Rate*: Single STMM per gesture

As expected, the *Non-recognition Rate*, in Figure 4.7 and Figure 4.8 abruptly declines with the increase of tolerance factor since the system gets less strict. Non-recognition rate, while using normal distribution is seen more or less as same for all variations in *nArea* for both approaches.

If we look at both *False Recognition Rate* and *True Recognition Rate*, the parameter result starts from the worst outcome at 1.1 and eventually performs best at 2.1 before increasing again. With both approaches, the distinction between the uniform and normal distribution can be clearly observed. The strictness of uniform distribution over normal distribution can be observed in all of the figures.

Considering all the positive and negative indicators of the recognition process, the system outputs best result while using single STMM per gesture and parameter set *toleranceFactor* = 2.3, *dist* = normal distribution and *nArea* = 10. The *True Recognition Rate* is 92.9%, *False Recognition Rate* is 3.27% and *Non-recognition Rate* 3.83%. On the contrary, single STMM per stroke gave its best result, *True Recognition Rate* of 82.22% and *False Recognition Rate* of 4.68% and *Non-recognition Rate* 13.1% with the same parameter configuration. The true recognition rate is actually quite low for single STMM per stroke when compared to single stroke gesture recognition using CHnMM

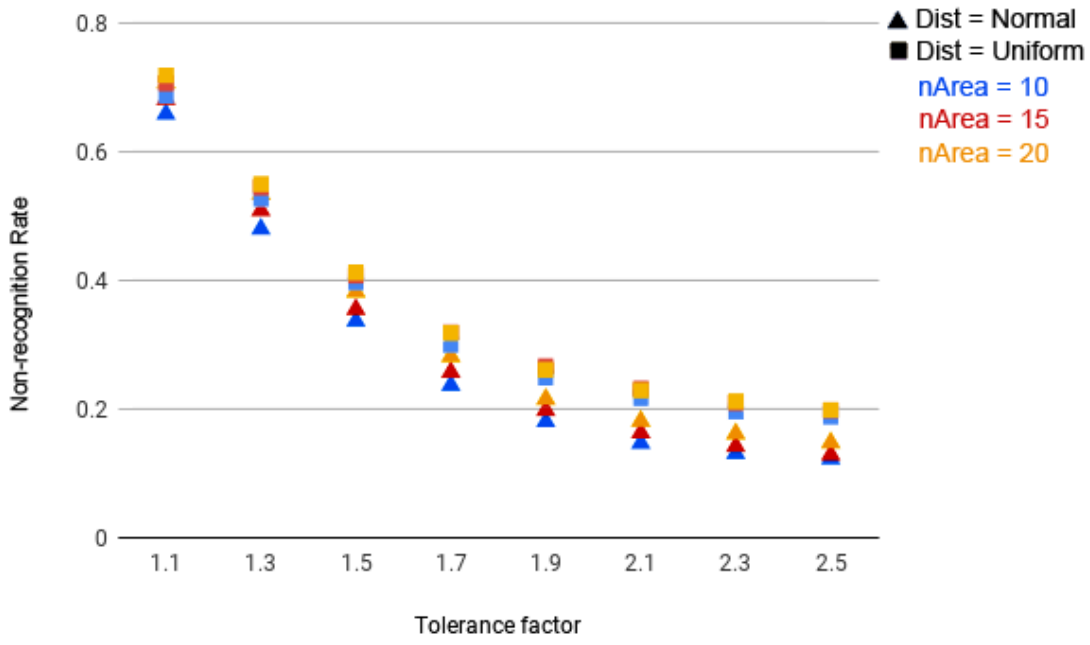


Figure 4.8: *Non-recognition Rate*: Single STMM per stroke

[6] which has more than 99%. Though the test was carried out on completely different data, both of them share exactly the same recognition process. This could be explained by the merging technique being less reliable in single STMM per stroke technique.

The precision of the recognition hovers around 97% with all the parameter sets(Figure 4.9 for our best performing parameter setting.

Interestingly, a rapid prototype recognizer, $N achieved the *True Recognition Rate* of 62.1% despite the fact that there are many similar gestures in terms of their spatial properties in our gesture set. The low result is due to the fact this recognizer only checks for the spatial match. Therefore, it is hard to distinguish same gesture from different users when there is only a spatial property as reference. As this recognizer does not reject any gesture, the rejection rate is zero which makes the *False Recognition Rate* 37.9%.

In a different setting, when $N recognizer was tested to recognize the gesture ID regardless of the user's identity, it resulted in the *True Recognition Rate* of 84.5%.

In a nutshell, single STMM per gesture is seen as a better option for the recognition task and can correctly identify the source most of the times. The best performing parameters result in 92.9% accuracy and has the precision of 96.6%.

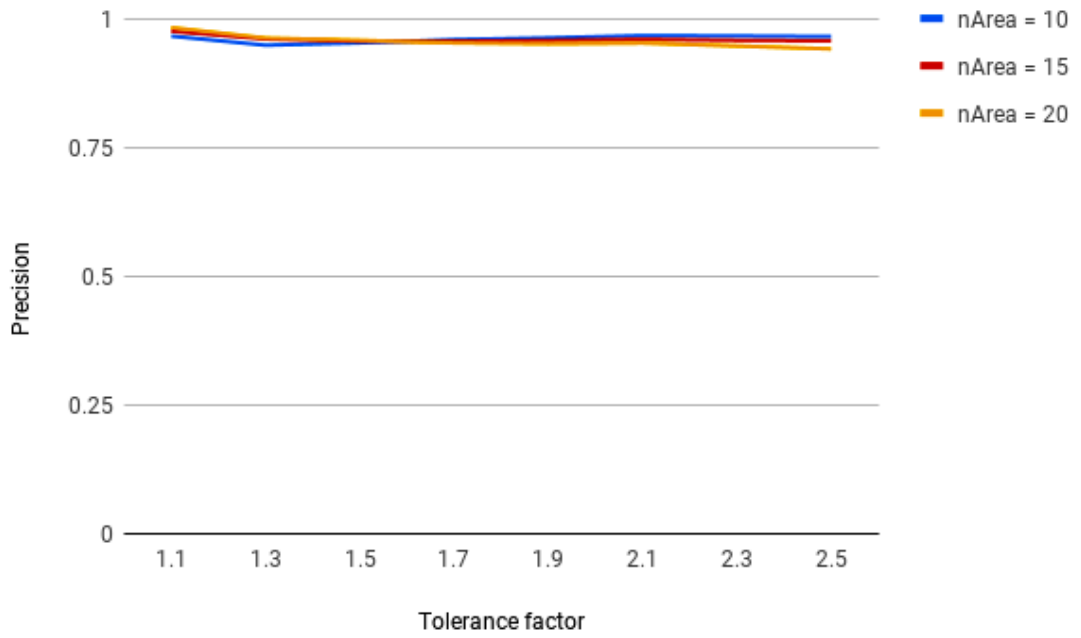


Figure 4.9: *Recognition Precision*: Single STMM per gesture with normal distribution

4.3 Gesture Verification

As a proposed application of this model, verification of users is also presented in this paper. In order to evaluate the possibilities for this, the verification experiment has been carried out. Similar to the earlier experiment, this experiment also employs parameter variations and cross validation. For this experiment, a gesture from particular user is considered as the genuine password and that can be used to verify that user. The same gesture form all remaining users are considered as forged passwords. Therefore, for each users, there are (8 users*14 gestures*20 executions + 3 users* 14 gestures* 15 executions = 2870 executions) that are tested and cross referenced for true verification. All genuine gestures should be accepted by the system as they represent the trained gesture and all forgeries should be clearly rejected. With this approach, it is possible to calculate a false acceptance rate (FAR) and a false rejection rate (FRR) for each parameter set, user and gesture sets.

FAR measures and evaluates the efficiency and accuracy of a bio-metric system by determining the rate at which unauthorized or illegitimate users are verified on a particular system. FAR is also known as false acceptance ratio or type II error rate/ratio. FAR is calculated by dividing the number of false acceptances by the number of identification attempts with forged data. Here, False acceptance is the condition where the system verifies someone else's gesture as a true user's gesture. So,

$$FAR = \frac{FA}{TFA}$$

Where:
FA = Number of False Acceptances
TFA = Total Number of Forged Attempts

False rejection is the condition where system fails to verify or identify an authorized person. Also referred to as a type I error, a false rejection does not necessarily indicate a flaw in the bio-metric system; for example, in a fingerprint-based system, an incorrectly aligned finger on the scanner or dirt on the scanner can result in the scanner misreading the fingerprint, causing a false rejection of the authorized user. A system's FRR typically is stated as the ratio of the number of false rejections divided by the number of identification attempts with a true user's data. So,

$$FRR = \frac{FR}{TGA}$$

Where:
FR = Number of False Rejections
TGA = Total Number of Genuine Attempts

The result of verification experiments for single STMM per gesture approach and single STMM per stroke approach are shown by Figure 4.10 - Figure 4.17. As expected, when the tolerance is increased, the system accepts higher amount of gestures(both genuine and forge) hence increasing FAR and decreasing FRR.

FAR is achieved lower when using normal distribution for single STMM per gestureFigure 4.10 where as it is opposite in the case of single STMM per strokeFigure 4.11. Another interesting nature seen here is that, with the increase in *nArea*, FAR increases for normal distribution and decreases for uniform. This phenomenon might be described by the fact that the increase in number of area increases the regions to be matched. And uniform distribution is more strict therefore acceptances of gestures decrease(both false acceptance and true acceptance). Whereas, for normal distribution, though the regions that need to be matched increase, the strictness also gets less.

Figure 4.12 and Figure 4.13 show the FRR for our dataset. FRR is achieved less with normal distribution for both single STMM per gesture and single STMM per stroke approaches for gesture verification. At best case, FRR tends to minimum, below 5% for single STMM per gesture whereas it stops around 14% for single STMM per stroke approach.

In Figure 4.14, the achieved Equal Error Rate(EER) from different parameter variations is presented. The values describe the balance between FAR and FRR results that occurred in the parameter variation. The EER of 4.5% is achieved at tolerance factor

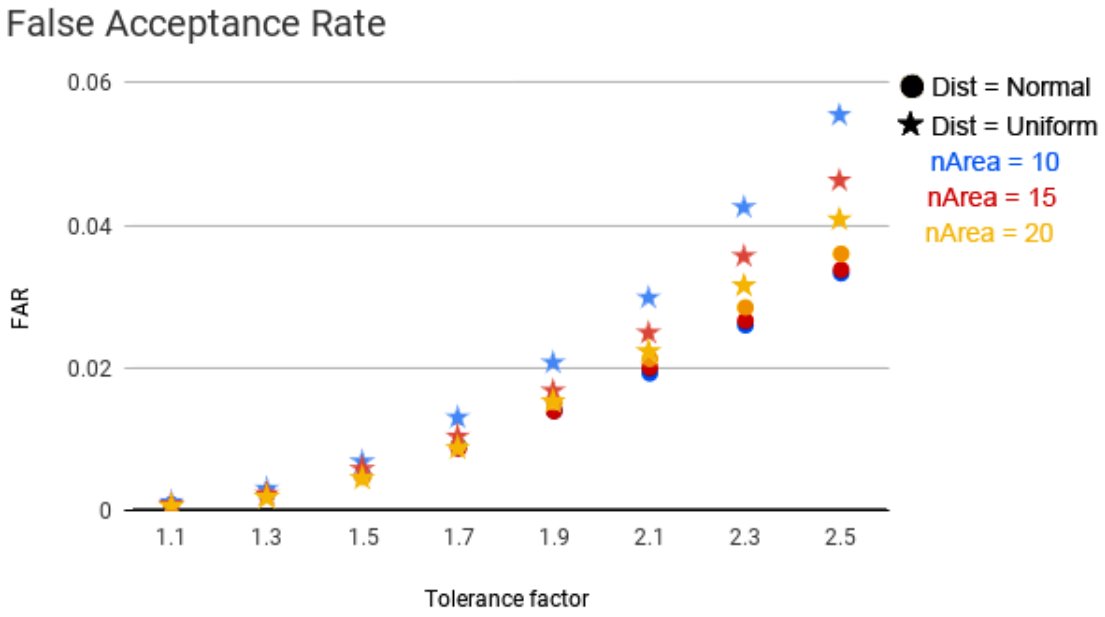


Figure 4.10: FAR: Gesture verification with single STMM per gesture

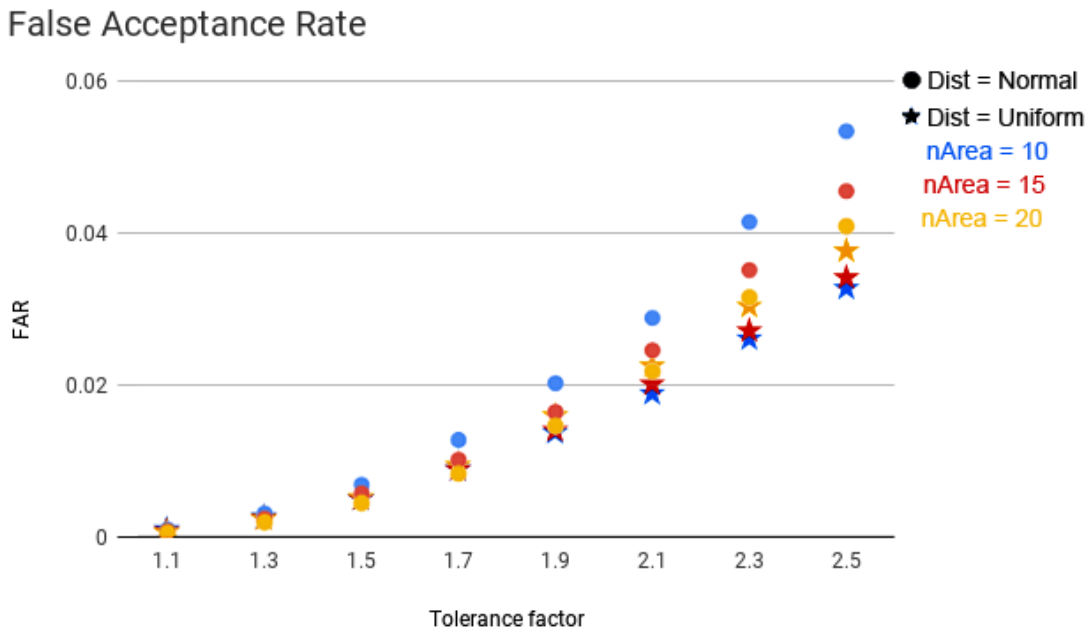


Figure 4.11: FAR: Gesture verification with single STMM per stroke

2.5 as seen. The achieved EER value shows the possibility of using the system in practice as it suggests that most of the time gestures can be distinguished well enough.

This result can also be compared to the results from other verification methods for cross referencing. However, for a fair comparison, a system that considers both time and space values of a gesture need to be referenced. $N recognizer currently does not implement the verification of the gesture therefore comparison is not possible unless it is tuned for verification on our own.

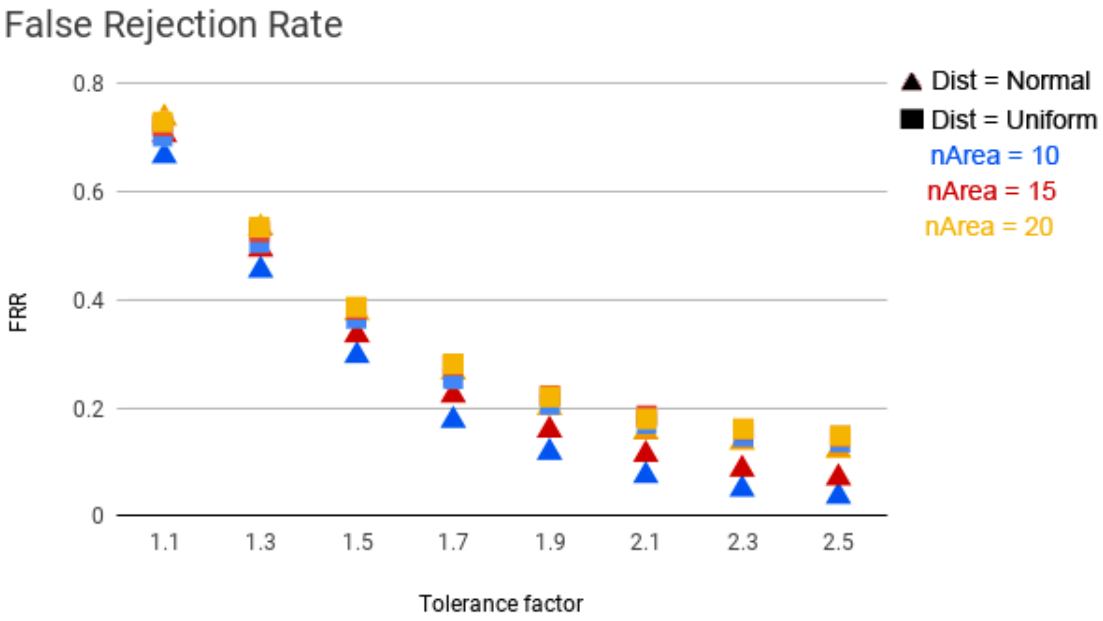


Figure 4.12: FRR: Gesture verification with single STMM per gesture

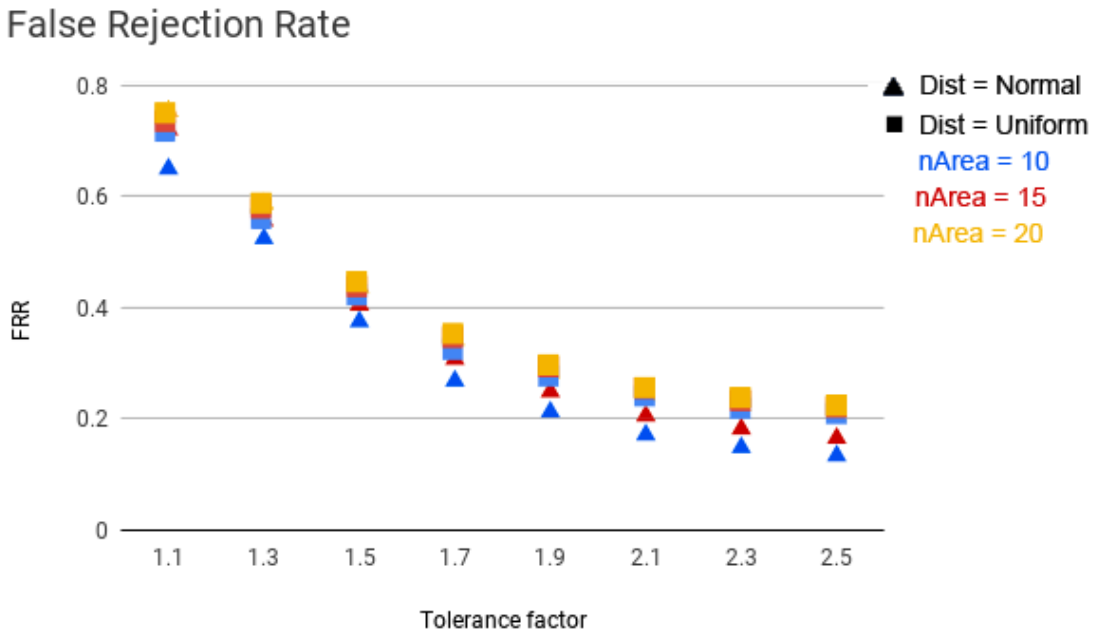


Figure 4.13: FRR: Gesture verification with single STMM per stroke

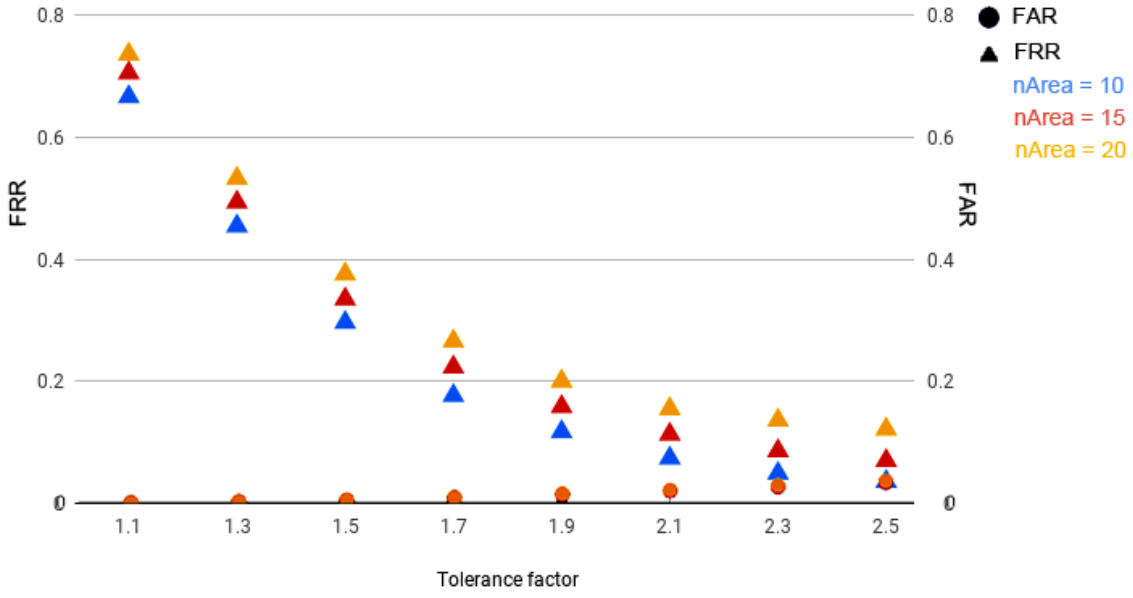


Figure 4.14: FAR vs FRR: Single STMM per gesture, normal distribution

A graph showing the *True Acceptance Rate* for the single STMM per gesture while using the normal distribution is presented in Figure 4.15. The graph shows that, *nArea = 10* performs best among the competitors when it comes to accepting a true user's gestures. The distribution of probabilities is similar for all the tolerance factors.

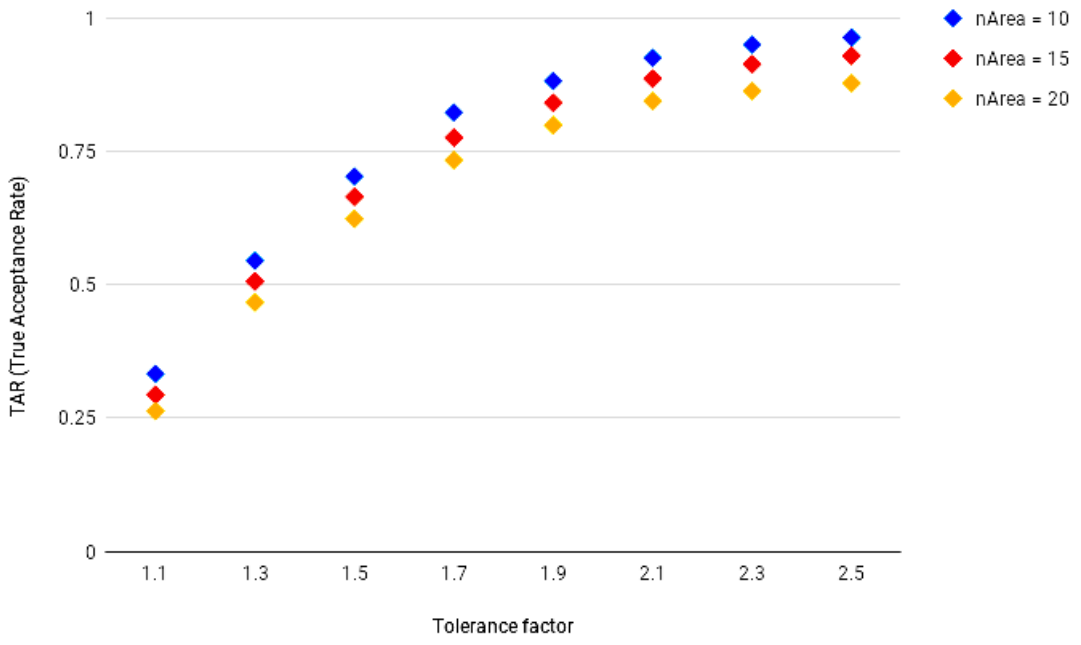


Figure 4.15: True Acceptance Rate: Single STMM per gesture, normal distribution

We also calculated the conditions where the spatial match was confirmed but the gestures differ in terms of temporal factor. This distinction is shown by the gesture mismatch(spatial match temporal mismatch) in Figure 4.16, Figure 4.17. Two variations, true rejection due to temporal mismatch and false rejection due to temporal mismatch are presented. True rejection due to time is desired outcome and is the condition where the gesture is from a different user but still it matches the spatial model, however, fails to match the temporal model. Hence is rejected. False rejection due to time is not a desired outcome and is the condition where the gesture is from a true user, it matches the spatial model, but it fails to match the temporal model. Hence is rejected. As we can see in both of the graphs(Figure 4.16, Figure 4.17), true mismatch increases rapidly with the increase of tolerance factor but false mismatch remains relatively similar. This shows that, if only spatial model was employed for the verification, then FAR would have rocketed but STMM can also discriminate the temporal difference between the gestures from different users.

Note: The maximum scale in bar graphs are not the total number of gestures used to carry out the experiment.

From all the resulting graphs, it is clearly noticeable that Single STMM per gesture simply outperforms single STMM per stroke in every evaluation criteria. Also, for every indicators(FAR, TAR, FRR and ERR), normal distribution outputs better result. As we discussed in the recognition experiment, this is because the normal distribution approach is more forgiving than the uniform one if unknown trajectories do not perfectly fit the trained time window.

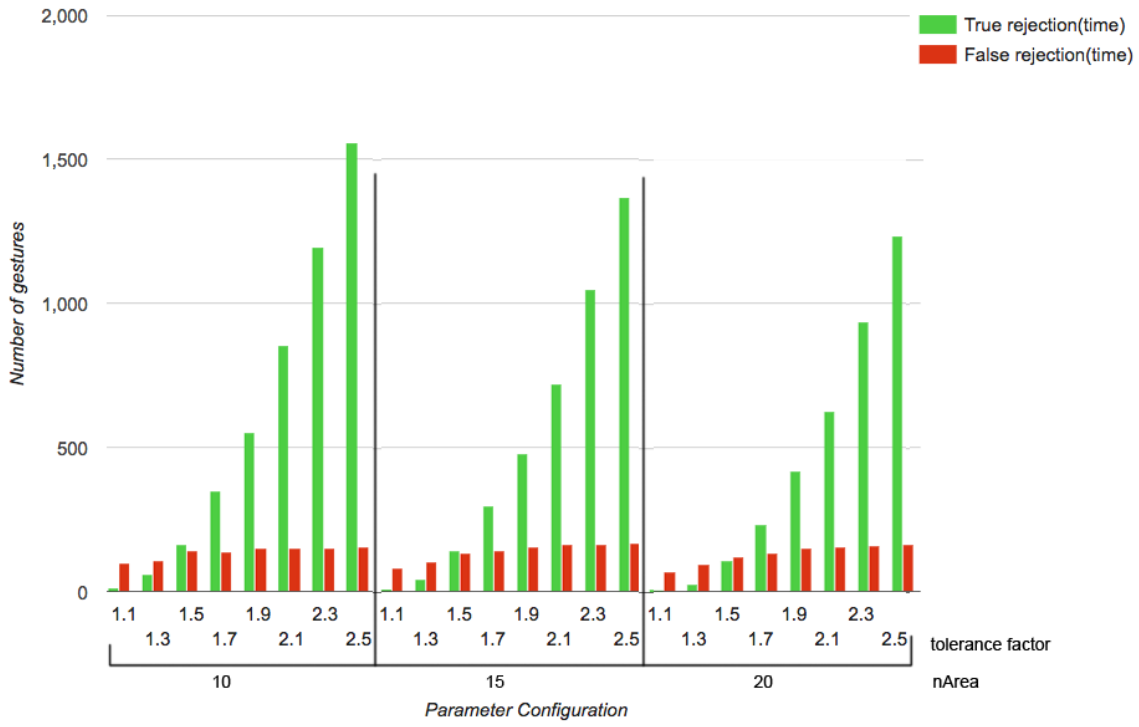


Figure 4.16: True rejection due to temporal mismatch: Uniform distribution

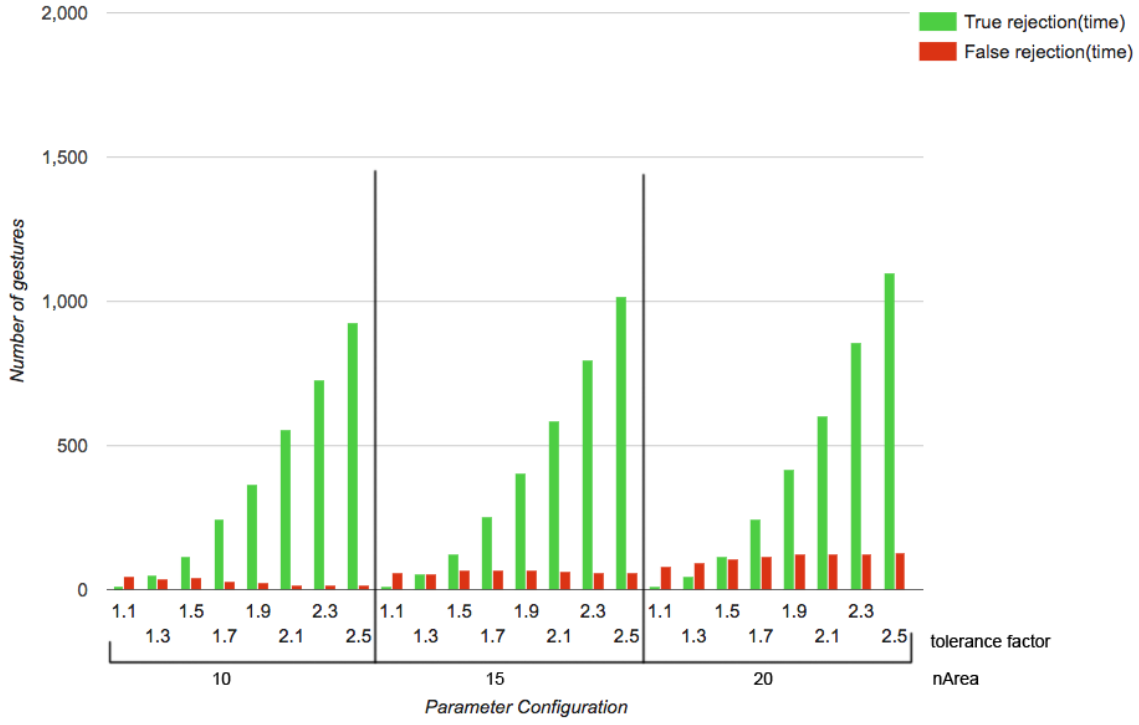


Figure 4.17: True rejection due to temporal mismatch: Normal distribution

Considering all the positive and negative indicators of the verification process, the results show best while using parameter set *toleranceFactor* = 2.3, *dist* = normal distribution and *nArea* = 10. The *True Acceptance Rate* is 95.05%, *FAR* is 4.21% and *FRR* is 4.94%. A fairly noticeable difference to the results from Single stroke gesture recognition in [6] is that, the best performing *toleranceFactor* has moved up from 1.7 to 2.3. The increase in number of strokes and their co-ordination to form the gesture describes the reason.

4.3.1 STMM for Authentication

With the remote accessibility of IoT, a trustworthy process of confirming the identity of a person/source for providing access to a system is one of the most discussed and researched topic. Authentication does not necessarily mean identity verification. Identity verification will produce a token that tells the system either the user is verified or not. With that token, an authentication system will allow the user to enroll into the system service. According to [1], "Authentication" is an online activity performed when the database is approached by a user or by an agent on their behalf, claiming that they are someone by presenting their identifier. Authentication involves verifying the identifier by using the pre-arranged secrets. A typical method of determining the identity of a true user is verification with a combination of username and password. More advanced mechanisms such as drawing patterns on touch enabled devices and biometrics are also used. Majority of the pattern based authentication processes available today, such as Pattern Lock in Android OS[27], are not able to consider temporal dynamics as a feature of the authentication process.

As we saw in our experiment that STMM for multi-stroke gesture recognition has possibilities to identify or distinguish the characteristics of a user, we implemented the prototypical authentication of users using STMM in a web based application. We created a two way verification system, one with username and text password and another with username and user specific gesture. We expect a user to be able to log-in to the web application by drawing a gesture instead of using text based password. At the beginning, a user would log-in with text password and once inside, create a gesture password. For easy remembrance and consistency of trajectory, the user could upload their reference image of their preference if they want to. The user then has to draw this gesture 7 to 8 times so that a model could be learned. After saving the gesture, when logged out, the user could choose to log-in with his/her specific gesture. As the results from the verification experiment above suggested, most of the time, user were able to log-in to their account. This experiment was carried out to show that our system can be used for authentication purpose.

In this chapter, we focused on interpreting our gesture recognition models based on their results from the recognition and verification experiments. Based on these experiments, Single STMM per gesture seems to be better than single STMM per stroke with wide difference in recognition rate and verification rate. The system achieved 92.9% accuracy with 3.27% *False Recognition Rate*. Similarly, a promising result is seen from the verification experiment and its implementation.

The results from our experiments suggest that CHnMM is useful for multi-stroke gesture recognition to model the temporal dynamics. Gesture recognition systems implementing time dimension as an extra feature of gesture modeling and recognition(like ours) can outperform simple gesture recognition systems that use just spatial properties(for example $N). Accuracy and reliability of the system can be increased by improving this work for real-life usage.

5. Conclusion

Lately, multi-touch devices are gaining popularity in every sector ranging from home appliances to rocket science, personal use devices to security based systems. As of today, multi-stroke gestures are mostly used for simple manipulation tasks such as zooming, dragging objects in the screen etc. In this thesis, we expanded the usage of multi-stroke gestures and opened new doors by characterizing the user specific features with multi-stroke gesture models.

A CHnMM based structure for representing spatio-temporal movement trajectories for single stroke gestures was presented in the previous work and the main goal of this thesis was to find out if CHnMM can be used for multi-stroke gesture recognition or not. We found that there were not enough multi-touch gesture dataset available that could be used to build and test our intended features. Therefore we created our own gesture dataset containing different types of gestures varying in terms of number of strokes and speed of execution. To reach to our goal, two different approaches to model multi-stroke gesture were proposed and implemented. We showed that single STMM per gesture approach was also built on the core of multiple STMMs per gesture. With the dataset, we carried out recognition and validation experiments for both of the approaches and additionally tested our dataset against $N, a rapid prototyping recognizer. Both of the approaches were able to recognize the gestures correctly to some extend.

5.1 Conclusion

From the result of our experiments, we found the approach that creates single STMMs per gesture performs better compared to the one creating multiple STMM per gesture. This was as predicted to some degree since the number of strokes specify the number of STMMs and with the increase in STMMs, the underlying error also increases resulting to weak recognizer. The main source of error in our case was merging the recognition results from multiple STMMs into one. Nevertheless, both of our model approaches

outperformed the $N recognizer. The main reason behind this was found to be the use of CHnMM, which has the ability to model temporal dynamics more detailed by using arbitrary probability distributions for state duration. Hence, the best recognition rates were achieved compared to $N.

The outcome of the conducted experiments shows that our goal has been achieved and suggest that, CHnMM can distinguish between similar multi-stroke gestures with spatial match but with temporal differences. Hence, can be employed to add time feature to the gesture recognition system. In this paper, also found out that, due to the available method parameters, it is possible to adjust the system behavior to the needs of the application. For example, increasing *toleranceFactor* increases the FAR but reduces FRR and vice versa.

This research aids to characterize partially observable discrete stochastic gestures in continuous time with concurrent non-Markovian behavior. They therefore enable us to gain insights into the unobserved behavior of 2D gestures. This can be further extended to 3D gestures.

5.2 Limitations and Future Work

With our research, we achieved promising results, but we still have a long way to go to use it in real life. Of course, there are noticeable areas for improvement in this thesis. As of this state, our system hugely relies on stroke sequence numbers that were correctly assigned while creating gesture dataset. If we were to train our model with the trajectories where stroke sequences are not pre-specified, then our system would fail to recognize the gestures correctly. To handle these kinds of gestures, a pre-recognizer needs to be introduced that will allocate proper sequence number to the candidate gesture strokes.

In the single STMM per gesture approach, trajectory information of all the strokes are tailed to first stroke in sequence and the time information re-configured with respect to the first point of first stroke hence carrying over the time inconsistency of earlier strokes. For example, the time inconsistency in first stroke will add the time in all following strokes even-though the later strokes were consistent within themselves. A different technique to extract time features and movement features between pair pieces of a gesture can be worked out. Similarly the result of multiple STMMs per gesture can also be improved by introducing a more reliable merging technique in contrast to the simple addition that is currently being used.

There is a possibility of developing a whole new approach for modeling of gesture other than these two approaches presented here. As of now, strokes within a gesture are ordered one after another, therefore a STMM just represents a part of gesture at certain instance. But, STMM that can represent complete gesture instance at certain time can also be developed. This approach was also considered as possible modeling approach but was not studied in detail. However, if this modeling approach is studied

and implemented, there is high chance of getting better results. This for example will eliminate the temporal error discussed in earlier paragraph.

Even though the base single trajectory model[6] can handle both 2D and 3D gestures, the current system works fine with 2D gestures but is not compatible with 3D gestures. Hence, it can be improved to handle 3D gesture as well and can be generalized for sign language interpretations. Also, $N recognizer uses rotation invariant to check for best possible matches but this paper lacks this feature. We have shown prototypically, how this system can work on web based platforms. Lastly, a more robust iteration of this system can be implemented for real world authentication systems in future.

Bibliography

[1] Agrawal, S., Banerjee, S., and Sharma, S. (2017). Privacy and security of aadhaar: A computer science perspective. 52:93–102.

[2] Anthony, L. and Wobbrock, J. O. (2010). A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*, GI '10, pages 245–252, Toronto, Ont., Canada, Canada. Canadian Information Processing Society.

[3] Bosse, S., Krull, C., and Horton, G. (2011). Modeling of gestures with differing execution speeds: Are hidden non-markovian models applicable for gesture recognition.

[4] Buchholz, R. (2018). Conversive hidden non-markovian models.

[5] Chen, Z. (2017). *Recognition and interpretation of multi-touch gesture interaction*. Theses, INSA de Rennes.

[6] Dittmar, T., Krull, C., and Horton, G. (2017a). A conversive hidden non-markovian model based structure for discriminating spatio-temporal movement trajectories. *International Journal of Simulation and Process Modelling*, 12(3-4):274–286.

[7] Dittmar, T., Krull, C., and Horton, G. (2017b). Evaluating a new conversive hidden non-markovian model approach for online movement trajectory verification. pages 249–258.

[8] Faundez-Zanuy, M. (2007). On-line signature recognition based on vq-dtw. *Pattern Recogn.*, 40(3):981–992.

[9] Fierrez, J., Ortega-Garcia, J., Ramos, D., and Gonzalez-Rodriguez, J. (2007). Hmm-based on-line signature verification: Feature extraction and signature modeling. 28:2325–2334.

[10] Fink, G. A. (2014). *Markov Models for Pattern Recognition - From Theory to Applications*. Springer Science and Business Media, Berlin Heidelberg.

[11] Furui, S. (2018). Digital speech processing, synthesis, and recognition / sadaoki furui.

[12] Ghotkar, A., Vidap, P., and Deo, K. (2016). Dynamic hand gesture recognition using hidden markov model by microsoft kinect sensor. 150:5–9.

[13] Hwang Juang, B. and R. Rabiner, L. (1991). Hidden markov models for speech recognition. 33:251–272.

[14] Kholmatov, A. and Yanikoglu, B. (2005). Identity authentication using improved online signature verification method. *Pattern Recogn. Lett.*, 26(15):2400–2408.

[Krull and Horton] Krull, C. and Horton, G. Hidden non-markovian models : Formalization and solution approaches.

[16] Krull, C. and Horton, G. (2008). The effect of rare events on the evaluation and decoding of hidden non-markovian models.

[17] Krull, C. and Horton, G. (2009). Solving hidden non-markovian models: How to compute conditional state change probabilities.

[18] Min, B.-W., Yoon, H.-S., Soh, J., Yang, Y.-M., and Ejima, T. (1997). Hand gesture recognition using hidden markov models. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 5, pages 4232–4235 vol.5.

[19] Moni, M. A. and Ali, A. B. M. S. (2009). Hmm based hand gesture recognition: A review on techniques and approaches. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 433–437.

[20] Nebeling, M. and Norrie, M. (2012). jqmultitouch: Lightweight toolkit and development framework for multi-touch/multi-device web interfaces. In *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '12, pages 61–70, New York, NY, USA. ACM.

[21] Niels, R. and Vuurpijl, L. (2005). Dynamic time warping applied to tamil character recognition. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 730–734 Vol. 2.

[22] Plouffe, G. and Cretu, A. M. (2016). Static and dynamic hand gesture recognition in depth data using dynamic time warping. *IEEE Transactions on Instrumentation and Measurement*, 65(2):305–316.

[23] Rigoll, G., Kosmala, A., and Eickeler, S. (1998). High performance real-time gesture recognition using hidden markov models. In Wachsmuth, I. and Fröhlich, M., editors, *Gesture and Sign Language in Human-Computer Interaction*, pages 69–80, Berlin, Heidelberg. Springer Berlin Heidelberg.

[24] Rubine, D. (1991). Specifying gestures by example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 329–337, New York, NY, USA. ACM.

[25] Sae-Bae, N., Ahmed, K., Isbister, K., and Memon, N. (2012). Biometric-rich gestures: A novel approach to authentication on multi-touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 977–986, New York, NY, USA. ACM.

[26] Schmidt, M. and Weber, G. (2013). Template based classification of multi-touch gestures. *Pattern Recognition*, 46(9):2487 – 2496.

[27] Shabtai, A., Fledel, Y., Kanonov, U., Elovici, Y., Dolev, S., and Glezer, C. (2010). Google android: A comprehensive security assessment. *IEEE Security Privacy*, 8(2):35–44.

[28] Vatavu, R.-D., Anthony, L., and Wobbrock, J. O. (2012). Gestures as point clouds: A $p recognizer for user interface prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pages 273–280, New York, NY, USA. ACM.

[29] Wang, H. and Wang, L. (2017). Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. *CoRR*, abs/1704.02581.

[30] Wobbrock, J. O., Wilson, A. D., and Li, Y. (2007). Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 159–168, New York, NY, USA. ACM.

[31] Wu, D. and Shao, L. (2014). Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–731.

[32] Xia, L., Chen, C. C., and Aggarwal, J. K. (2012). View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27.

STATEMENT OF AUTHORSHIP

Thesis: A CHnMM Approach for Multi Stroke Gesture Recognition on Touch Input Devices

Name: Umesh | Surname: Bastola

Date of birth: 01.11.1989 | Matriculation no.: 213653

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning.

I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Magdeburg, August 1, 2018

_____

Signature