AN IMPROVED CONVERSIVE HIDDEN NON-MARKOVIAN MODEL-BASED TOUCH GESTURE RECOGNITION SYSTEM WITH AUTOMATIC MODEL CREATION

Tim Dittmar[(a)], Claudia Krull[(b)], Graham Horton[(c)]

Otto-von-Guericke-University Magdeburg
P.O. Box 4120
39016 Magdeburg, Germany

[(a)]tim.dittmar@ovgu.de, [(b)]claudia.krull@ovgu.de, [(c)]graham.horton@ovgu.de

ABSTRACT
Mobile devices like smartphones and tablets that are controlled via a multi-touch interface have become ubiquitous. In previous work a touch gesture recognition system based on Conversive Hidden non-Markovian models has been proposed that is able to recognize similar gestures with different execution speeds based on recorded examples. With this work, we improved the system by eliminating the major drawback of manually and tediously creating models for every gesture from recorded training data. To achieve this, the gesture model design has been adapted to include an additional structure that represents a map of all known gesture examples. Experiments conducted on two different datasets show that the new system can distinguish gestures with different speeds with good accuracy and fast detection times. Ideas to further improve the system are discussed and we believe that such a system could be the basis for a new gesture authentication system in the future.

Keywords: touch, gesture recognition, CHnMM, mobile

1. INTRODUCTION

The presence of mobile touch devices has become ubiquitous, especially due to the success of smartphone and tablet devices, which are primarily operated via the multi-touch interface they provide. This way of interaction and these device classes became popular with the introduction of the Apple iPhone in 2007 and the Apple iPad in 2010. Due to this development touch devices and interaction play an important role nowadays. Two facts that reinforce this are that Microsoft optimized their Windows Desktop operating system for touch input and Google adapted their web page ranking algorithms to consider the mobile (touch) quality of websites.

Many internet services are accessed from these mobile touch devices and the user authentication almost always requires the user to enter his or her username and a password, which takes significantly more time on a glass surface than on a physical keyboard (Findlater 2011), especially when it is a long and secure password with capital letters and special characters. An alternative authentication method for touch devices that has been mentioned already is the verification of users by touch gestures (Sae-Bae 2011, Sherman 2014) which have the potential to be far more convenient to enter by the user. But to our knowledge these methods do not consider differences in temporal dynamics of a gesture execution and only validate the shape, although the discrimination of gestures with different temporal dynamics would add an additional factor of entropy and therefore be more secure.

With this work, a first step towards such a touch gesture authentication system that considers temporal dynamics is taken. Our previous experiments (Bosse 2011, Dittmar 2015) have shown that Hidden non-Markovian Models (HnMM) and their subclass Conversive Hidden non-Markovian Models (CHnMM) are able to distinguish similar gestures also by their temporal dynamics. But the models in these papers had to be manually and tediously created by an expert from a set of training examples, which render these approaches infeasible for applications in practice. A system that is feasible in practice would be required to automatically generate its gesture models from given example executions, which is currently not efficiently possible for HnMMs and CHnMMs. While it is possible with HMMs, these have another weakness. HMMs do not explicitly incorporate timing information, hence, they are dependent on a periodic symbol emission that implicitly adds this information, which is difficult to achieve if the system is required to support a large set of different touch devices.

Consequently, we propose a new CHnMM-based hybrid model for touch gestures that also considers temporal dynamics. By introducing a new data structure, the *StrokeMap*, an automatic creation of the CHnMM from a set of given examples is facilitated. Additionally, a prototype gesture recognition and authentication system is created to evaluate our proposal.

2. BACKGROUND

In the following sections our previous work, a formal CHnMM definition and related work are presented to

put this paper into context and to increase the understanding of its contents.

2.1. Previous Work

In this section a short overview of previous work that this paper relies on is presented to explain some basic methods and models needed to understand this paper and our approach. Furthermore, a formal definition of a CHnMM is given at the end of this section.

In Krull et al. (2009) an extension to the popular Hidden Markov Model (HMM) (e.g. used in speech-, gesture- and handwriting recognition (Fink 2014)), has been presented: the so-called Hidden non-Markovian models that are more powerful regarding their modelling capabilities. For example, they incorporate arbitrary distributions for state sojourn times instead of only geometric distributions like HMMs do implicitly by utilizing fixed probabilities and a fixed time step for state changes. Due to their complex modelling capabilities the solution algorithms of HnMMs are complicated and computationally very demanding and that is why the subclass of CHnMMs was defined and thoroughly analysed by Buchholz (2012) in his dissertation. This subclass only slightly limits the modelling capabilities of HnMMs by requiring an output symbol for every state change, but this small change allows much more efficient solution algorithms. These algorithms (one is utilized in this paper) employ the Proxel method, which is not further explained in this paper, but the dissertation by Lazarova-Molnar (2005) covers this method thoroughly.

In 2011, Bosse et al. (2011) successfully showed that HnMMs can be used to create a Wiimote gesture recognition system that is also able to distinguish between similar gestures with different execution speeds. Similarly, Dittmar et al. (2015) utilized CHnMMs to create a touch gesture recognition system for a multi-touch tabletop, and showed in experiments the ability of the recognition system to distinguish touch gestures by their temporal dynamics.

In both experiments the systems have been compared to an HMM-based recognition system. While these performed slightly worse than the HnMM- and CHnMM-system, they also were able to distinguish the gestures with different execution speeds in most cases, even though HMMs do not incorporate explicit timing information. The ability comes from the fact that implicit timing information is provided due to the periodic emission of symbols that is used for HMM-based systems.

Another problem of both approaches is the fact that the models for the gestures were created and parameterized manually. A circumstance that renders the system infeasible in practice, because no user would be able to create the models on their own and even an expert needs a long time to create it. With the new approach presented in this paper we especially want to eliminate this problem by trying to employ a supervised learning approach as proposed by Bosse (2012) for CHnMM-based gesture recognition systems.

2.2. CHnMM – Formal Definition

A CHnMM contains the following elements that are similar to the elements of HMMs:

- a set of states S of length N
- a set of output symbols V of length M
- an initial probability vector $\Pi=(\pi_1,\ldots,\pi_N)$
- an NxN matrix A containing the state change behaviour, but with more complex elements a_{ij}.

Additionally, a CHnMM contains the set $TR=\{tr_1,tr_2,\ldots,tr_K\}$ of K transitions that define the model behaviour. 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 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. A CHnMM model λ is fully defined as a tuple $\lambda=(S,V,A,TR,\Pi)$ 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. The path Q also contains a sequence of states q_t with associated time stamps.

2.3. Related Work

The idea to use gestures for authentication is not completely new. A basic and widespread approach that is similar is the so-called “Pattern Lock” used in the Android mobile operating system (Shabtai 2010). Further research that focuses on authentication for unlocking a certain device includes Sae-Bae et al. (2012) and Sherman et al. (2014), but none of them considers temporal dynamics.

The application of HMMs for gesture recognition in general is very common but quite rare for touch gesture recognition systems. Two systems that employ HMMs to learn touch gestures by example are presented by Damaraju et al. (2008) and Joselli et al. (2009). The

first system is able to learn and recognise multi-touch gestures performed on a tabletop, while the latter learns and detects single stroke gestures on mobile touch devices. But in both cases different execution speeds are not tested or even considered.

To our knowledge the recognition of similar gestures that differ only in execution speeds has never been further investigated (except in our previous work), although there exist many HMM-based gesture recognition systems that incorporate temporal features like the velocity into the output symbols, for example (Chen 2003). On the contrary, most papers concentrate on tolerating the time variances in gesture executions, for example (Hong 2000), and do not try to exploit them.

3. THE MODEL

Our new method employs a single model per gesture, which is also the case for most HMM-based gesture recognition systems (for example (Damaraju 2008)). Unlike our previous work (Dittmar 2015), we do not solely use a CHnMM to represent a gesture but combine it with what we call a *StrokeMap*, a structure that holds all the information about the shape of a gesture stroke. The creation of this structure based on example gestures is explained in detail in the following section.

The reason for introducing this new structure is to enable supervised learning by separating a gesture into known and clearly defined sections. It is also driven by the idea that for a gesture authentication system mainly the shape and the temporal dynamics of a gesture are relevant and the new structure helps to add valuable information about the shape to the output symbols.

The CHnMM represents the temporal dynamics of the gesture and is explained in Section 3.2 after the *StrokeMap*.

3.1. Creating the StrokeMap

As already mentioned and as the name suggests the *StrokeMap* holds all the information about the shape of a gesture stroke in the form of circular areas representing expected locations of successive points of the trace. It is created from all training examples of the gesture.

In Figure 1 all steps involved in creating a StrokeMap from training data are visualized with two example trials used as training data. The first part of the figure shows the recorded points of both examples (1.). In the first step of the *StrokeMap* generation process these points are interpolated linearly to get a continuous path (2.). The next step is to calculate *n* spatially equidistant points for each example gesture, i.e. a pair of adjacent points has the same length of interpolated stroke path between them. In the visualized example the value *n* is five but apart from that, *n* is a parameter or setting of the recognition system that is called *nArea* within this paper. In another step the calculated equidistant points are grouped together (all first, second, third, etc. points) to so called area points to create circle shapes around them containing all points of a group (4.). These circles represent the area where a gesture stroke is expected to be after a certain distance of the gesture has been executed. The centre of the circle is determined by determining the average of all points of a group while the radius is given by the distance from the circle centre to the point of the group with the maximum distance from the circle centre. As a result the created circle contains all the points of the group.

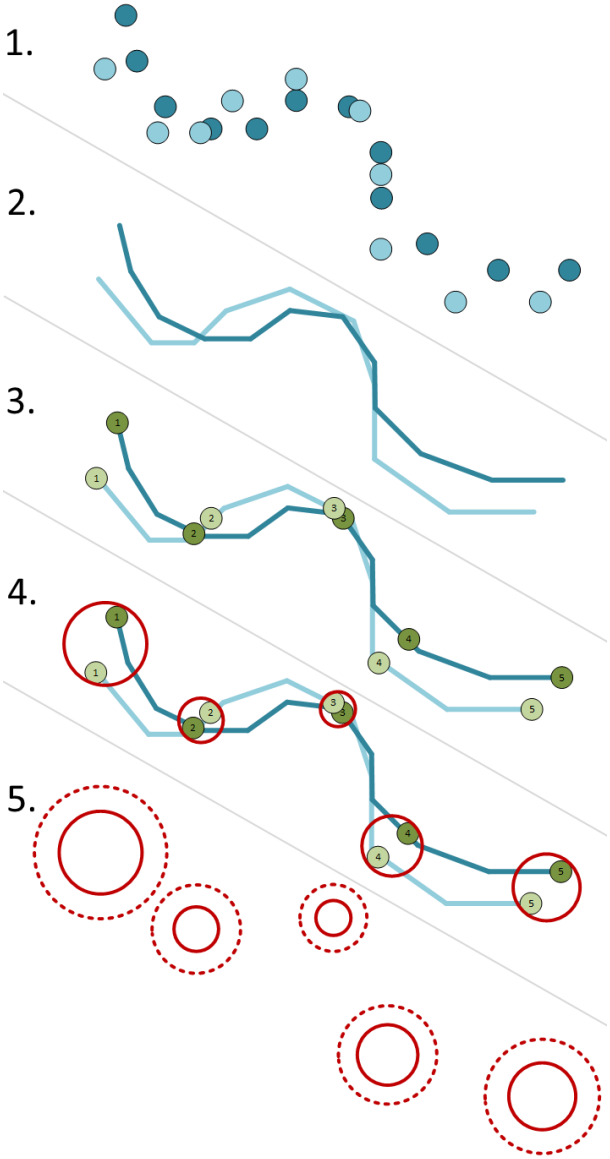

Figure 1: StrokeMap Generation Process Demonstration

As a last step, slightly larger circles are calculated for each circle representing a tolerance area, as it cannot be expected that new and unknown executions of the same gesture will go through the already calculated circles, especially if only a small number of example gestures is available. The radius of the tolerance circle is determined by multiplying the original circle radius with the factor *tolF* which is another parameter of the recognition system. Eventually, the final *StrokeMap* consists of the ordered set of circles and their tolerance radii (5.).

3.2. Creating the CHnMM

The design of the CHnMM for a gesture is closely connected to the *StrokeMap*. Each area (a circle and its tolerance circle are considered one area) of the *StrokeMap* is represented by a state of the CHnMM. Furthermore, a start state is added, hence the set of states is $S=\{Start,A1,A2,\ldots,An\}$. These states are linearly connected with transitions $TR=\{T1,T2,\ldots,Tn\}$ resulting in a layout that resembles the linear topology known from HMMs (Fink 2014). Consequently, the elements a_{ij} where $i=j+1$ are mapped to the elements of set *TR*, all other matrix elements are empty. A graphical representation of this design is given in Figure 2. Further visualized elements are the output symbols $V=\{A1\_Hit,A1\_Tol,\ldots,A2\_Hit,A2\_Tol,An\_Hit,An\_Tol\}$ and their output probabilities . In this first iteration of our new approach $b(A_i\_Hit)$ is set to 0.9 and $b(A_i\_Tol)$ is set to 0.1 to inflict a penalty for gestures that only go through the tolerance areas of the *StrokeMap*. Further details on the symbols and their creation and meaning are given in Section 3.3.

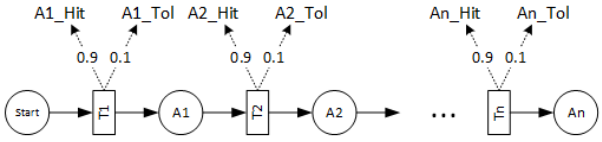

Figure 2: CHnMM Design for a Gesture

All the aforementioned CHnMM elements are already determined by the connected *StrokeMap*. Incidentally, the *aging* element of the transitions is not relevant for this design, because there are no competing transitions. The remaining *dist* element for each transition of the CHnMM is the part of the model that represents the temporal dynamics of the gesture. For *T1* the distribution is always the same as it is deterministic because the first symbol always occurs at the very beginning, hence, the state change from *Start* to *A1* happens in an instant. However, it is included in the model to incorporate the information of the first symbol which would be just a dummy symbol otherwise. For the remaining transitions, the calculated area points (3. section in Figure 1) are utilized to calculate the required time from area A_i to A_{i+1} for each training gesture, which is possible because every area point also contains a time value which is the linearly interpolated timestamp. As a result, a list of sample times is created for every transition (except *T1*) that can be utilized to estimate a probability distribution. In this paper, a uniform distribution was chosen using the minimum and maximum of the collected samples as parameters.

This new process makes it possible to automatically create a CHnMM-based gesture model from a set of examples, which is the main goal of this research.

3.3. Symbol Creation

The symbol creation process transforms the data of an executed gesture to a symbol trace *O*, which is used to classify the gesture. For the proposed system, the output symbols are connected to the *StrokeMap* and the chosen symbols (see Section 3.2) contain information about the shape of the gesture due to this connection. In detail, a given executed gesture is processed in an analogous manner to the process described in Section 3.1. Thus, area points are determined for the given single gesture execution and checked against the corresponding areas in the *StrokeMap*. If an area point lies within the inner circle area, the symbol A_i_Hit is emitted, if it lies within the tolerance circle it is symbol A_i_Tol and if it lies outside of the tolerance circle the trace generation is cancelled, as this already indicates that the gesture does not fit the *StrokeMap*. This way the processing of gestures that do not fit the gesture model shape-wise can be cancelled early.

3.4. Gesture Authentication and Classification

In a *gesture authentication scenario* or to be more precise in a gesture verification scenario, a user would identify himself first, for example with a unique username. The recognition system knows the gesture model (*StrokeMap* and CHnMM) that was created for this username at registration and to be authenticated, the user needs to recreate the gesture. For the inputted gesture the symbol creation process is conducted, and if all area points at least lie within their respective tolerance areas of the *StrokeMap* a symbol trace *O* is generated as described above. If no trace could be created the authentication fails. Otherwise, the so called evaluation task is performed on the CHnMM using the trace *O* to calculate $P(O|\lambda)$, employing the Proxel method. With an evaluation value greater than 0 the authentication succeeds for a given gesture and username.

For a *gesture recognition scenario* an executed gesture needs to be classified as a gesture from the set of trained or known gestures respectively. In this case, the newly developed system in this paper creates a trace *O* from the given gesture for each known gesture model. This is different to our previous work and most HMM-based systems, where only one trace is created. For each trace O that could be created, which should be the case for similarly shaped gestures only, the evaluation task is conducted and the gesture model that generated the highest value is used as classification result. In case that no trace could be generated or that no evaluation value was higher than 0, no classification result is returned and the inputted gesture is considered to be different from all known ones. This represents an important difference to most existing gesture recognition systems, where every input is classified to the best fitting gesture even if they are shaped completely different.

4. EXPERIMENTS & RESULTS

To evaluate the abilities and quality of our system different experiments have been conducted. These experiments are based on two different datasets.

4.1. Description of the Datasets

Dataset1 consists of touch gesture data from ten different persons. Each participant was told to think of a gesture that he or she could use as a gesture password and to perform it twenty times while attempting to use the same position, shape and speed. In Figure 3 these gestures are visualized by an example of each user.

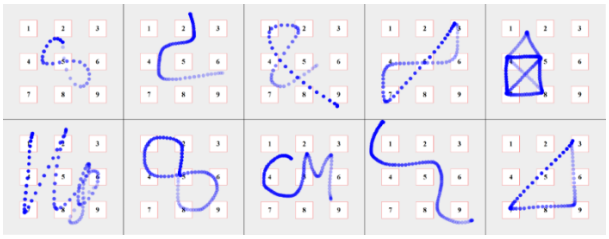

Figure 3: Gestures of Dataset1

Dataset2 has been created to evaluate the ability of the recognition system to distinguish gestures with different execution speeds and is illustrated in Figure 4. It consists of three different shapes namely a circle (C), a shape formed like the letter D (D), and a triangle (T), which are all performed counter-clockwise and starting at the top. These shapes are performed at two different speeds, fast (f) and slow (s) and consequently the dataset consists of six different gestures each performed thirty times. The similarity in shape C and D adds an additional challenge for the recognition system.

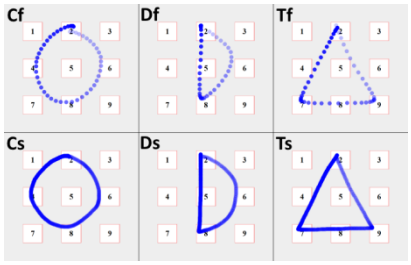

Figure 4: Gestures of Dataset2

Both datasets have been recorded on an iPad using the same user interface where the touch area has a numbered grid in the background (as seen in Figure 3 and Figure 4) to aid the user in performing the gestures at the same position. Furthermore, the touch data has been retrieved in the browser via JavaScript instead of an application to demonstrate that the recognition system could be employed in the web.

To increase the expressiveness of our experiments a cross validation approach is utilized, i.e. the dataset is split into *k* subsets of the same size and each subset is used as a test set while the *k-1* remaining subsets are used to train the recognition system. As a result, *k* tests can be conducted with one dataset and the test data will always be different from the training data.

Furthermore, a parameter variation is performed on each experiment to analyse the behaviour of the system for different parameter sets. The parameters and their ranges are as follows:

- *nArea* – The number of areas used in the StrokeMap (range: 10–20, step: 2)
- *tolF* – The tolerance factor used to determine the size of the tolerance area (range: 1.1–2.1, step: 0.1)
- *minRadius* – The minimal radius of each area (range: 0.01–0.19, step: 0.2)

As a result, 660 different parameter sets are created within the parameter variation. The unit used for the *minRadius* parameter is related to the employed interface and its coordinate system. The touch area is 600x600 pixels in size while the x and y values for the point coordinates range from zero to one. Thus, a *minRadius* value of 0.5 would create circles that are as wide as the touch area (diameter of 1.0).

The recognition system and the experiments were implemented in C# and all experiments were processed on a usual laptop with an Intel Core i5 processor (2410M @ 2.3GHz) and 6GB RAM.

4.2. Basic Gesture Recognition

First, a gesture recognition experiment was conducted to see if our new recognition system is able to classify different gestures based on training data. We conducted a parameter variation and cross validation with four subsets on Dataset1 and recorded the result of each gesture classification. With four subsets of twenty gestures, every gesture is trained with fifteen and tested with five other gesture examples per subset, resulting in twenty classification results per parameter set and gesture. With 660 parameter sets, ten gestures and 20 tests for each gesture, a total of 132000 classifications have to be processed, including 2640 times of training the recognition system.

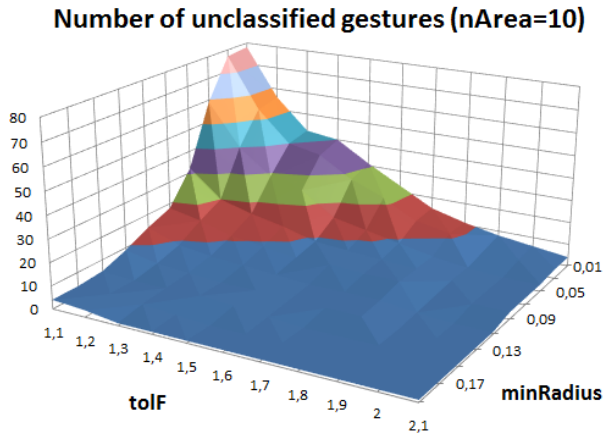

Figure 5: Number of Unclassified Gestures for Different Parameter Sets (*nArea*=10) using Dataset1 and 4 Subsets

The results of this experiment are very promising as they prove that our new approach does not fail the basic task of recognizing different gestures that were created from examples. The data reveals that not a single gesture has been classified as another gesture, but

as Figure 5 shows there are cases where the tested gesture could not be classified at all. This number of unclassified gestures increases with stricter and more intolerant parameters and peaks at 78 unclassified gestures (of 200 executed gestures). But a nearly perfect result with only one unclassified gesture can be achieved by setting the parameters more tolerantly. Consequently, a good setting for this kind of gesture set would be *nArea*=10, *tolF* = 1.6 and *minRadius* = 0.17 which only had one unclassified gesture, probably because of a badly executed gesture example, as even more tolerant settings cannot avoid it. In Figure 6 the results for *nArea*=20 are visualized which suggest that this parameter has only little influence to the general behaviour but slightly decreases the tolerance as it raises the number of unclassified gestures which peak at 83 in this case.

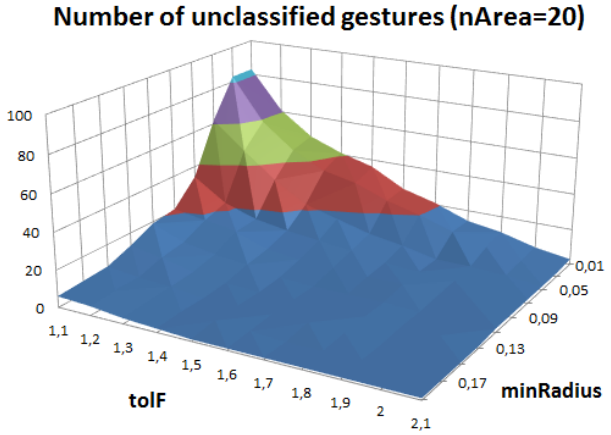

Figure 6: Number of Unclassified Gestures for different Parameter sets (*nArea*=20) using Dataset1 and four Subsets

The processing of this experiment (without writing results to file) was finished after 31 s, proving that the classification and even training is very fast, because the average classification time for a gesture is less than 0.24ms.

4.3. Temporal Gesture Recognition

For Dataset2 the same experiment approach as in the previous experiment has been used to evaluate the gesture recognition quality of the system if the gestures have different execution speeds. In this case, the dataset consists of six different gestures whose thirty examples are divided into five subsets, resulting in a total of 118800 classifications to be processed. Consequently, each gesture is trained with 24 training and tested with six gesture examples for each parameter set.

The calculation of the results took 44 s, which is around 13 seconds more than the first experiment, although fewer classifications had to be performed. The reason for this is probably the fact that more training processes had to be conducted due to the greater number of subsets. Additionally, the recognition system presumably cannot benefit from early cancelation as much as in the first experiment due to the similarity of most gestures. However, it is still very fast with less than 0.38ms per gesture on average.

For the recognition quality, the results for one parameter set (*nArea*=10, *tolF*=1.7, *minRadius*=0.01) are shown in Table 1, where a decent recognition quality could be achieved. Only one fast triangle execution was wrongly classified as slow and four executed gestures were not classified, leaving 175 out of 180 gestures that were correctly recognized. In Table 2 the same parameter set has been used but the number of subsets was reduced to two, resulting in a smaller training set of 15 gestures instead of 24, and a larger test set of 15 gestures instead of 6.

Interestingly, no gesture was wrongly classified this time, but the number of not classified gestures increased notably to 18 which is especially due to 12 slow circle gesture executions that were not classified. It is hard to tell why this particular gesture performed so much worse than the others, but it could be that there is more variation in the execution due to its length. Future investigations are necessary to verify this.

Table 1: Experiment Results with *nArea*=10, *tolF*=1.7, *minRadius*=0.01 and five Subsets

	Classified Gesture						
Executed Gesture	Cf	Cs	Df	Ds	Tf	Ts	None
Cf	30	-	-	-	-	-	-
Cs	-	29	-	-	-	-	1
Df	-	-	29	-	-	-	1
Ds	-	-	-	28	-	-	2
Tf	-	-	-	-	29	1	-
Ts	-	-	-	-	-	30	-

Table 2: Experiment Results with *nArea*=10, *tolF*=1.7, *minRadius*=0.01 and two Subsets

	Classified Gesture						
Executed Gesture	Cf	Cs	Df	Ds	Tf	Ts	None
Cf	29	-	-	-	-	-	1
Cs	-	18	-	-	-	-	12
Df	-	-	29	-	-	-	1
Ds	-	-	-	28	-	-	2
Tf	-	-	-	-	28	-	2
Ts	-	-	-	-	-	30	-

Another interesting fact is that these results could only be achieved using a *minRadius* of 0.01, because higher values were causing significantly more wrong classifications, as seen in Table 3 where *minRadius* is set to 0.05. Some fast gestures were falsely classified as slow while however the number of not classified gestures decreased to 0. The results suggest that the recognition system seems to prefer the slow variants of a gesture in some cases. The *tolF* parameter has a

similar influence as in the first experiment as it slightly reduces the number of not classified gestures the higher its value and additionally has a slight influence on the number of false classifications, which is, however, minor compared to the influence of *minRadius*.

Table 3: Experiment Results with *nArea*=10, *tolF*=1.7, *minRadius*=0.05 and five subsets

	Classified Gesture						
Executed Gesture	Cf	Cs	Df	Ds	Tf	Ts	None
Cf	25	5	-	-	-	-	-
Cs	-	30	-	-	-	-	-
Df	-	-	27	3	-	-	-
Ds	-	-	-	30	-	-	-
Tf	-	-	-	-	23	7	-
Ts	-	-	-	-	-	30	-

Based on this experiment, we can conclude that the system can differentiate gestures that only differ in execution speed, at least with a *minRadius* of 0.01, as with higher values the slow gesture variants seem to be preferred as a classification result.

4.4. Authentication

Since this paper is motivated by the idea of a gesture authentication system, a suitable experiment for evaluating the authentication quality has been conducted. The approach is similar to the previous experiments, employing parameter variation and cross validation. For each gesture a training set is used to create the gesture model. The test set contains genuine gestures that should be accepted by the system as they represent the trained gesture. Furthermore the examples of all other gestures of the dataset are tested against the created model to analyse whether the system correctly rejects them. With this approach it is possible to calculate a False Acceptance Rate (FAR) and a False Rejection Rate (FRR) for each parameter set, gesture and subset.

The results of this experiment for Dataset1 are rather uninteresting as it performs analogously to the first experiment and therefore achieves an average FAR of 0 and an average FRR of 0.05 for the same parameter set as in the first experiment. Presumably, the FRR is again caused by the badly performed gesture.

The results for Dataset2 are shown in Figure 7 where five subsets were used for the cross validation. Consequently, 3088800 gesture examples were authenticated and 19800 times a gesture model has been generated from 24 examples, and the process finished after 68 s. Thus, the average time to process a gesture is below 0.02ms. However, it has to be noted that the majority of the authentication attempts were fraudulent ones that benefit from early cancellation.

A quite good parameter set (*nArea*=10, *tolF*=1.5, *minRadius*=0.03) achieved an average FAR of 2% and an average FRR of 4%, which is marked with a red circle in Figure 7. This is a very good result for a prototype but especially the FAR value could not be accepted in real world applications where sensitive data and information need to be protected from unauthorized persons. The influence of the parameters is as expected. A higher tolerance (*tolF* and/or *minRadius* is increased) causes improved (smaller) FRR values, because more variants of a gesture are accepted. Simultaneously, the FAR values get worse (increase), because also fraudulent inputs are more likely to be accepted. The reason that the FAR and FRR values are worse than for Dataset1 is due to the fact that Dataset2 has gestures that only differ in their execution speeds and their discrimination only seems to work well for rather intolerant parameter sets.

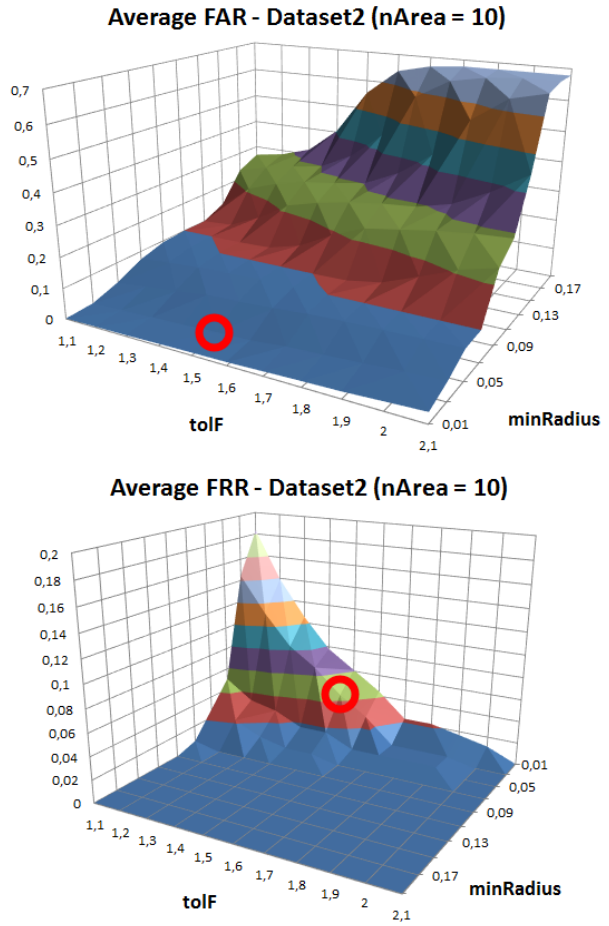

Figure 7: Average FAR & FRR for Dataset2 with Different Parameter Sets (NOTE: the *minRadius* scale has been reversed for a better visualization in the lower image)

The results show that the system can recognize different gestures by different users. The worse performance in the second experiment is due to the gesture examples being very simple (and similar) and therefore not a good choice for authentication gestures. This suggests that there should be rules for the choosing of secure gestures, as there are for secure passwords,

such as for example a minimum length of the gesture stroke.

5. CONCLUSION

With this paper we presented a prototype of a new CHnMM-based gesture recognition system that unlike our previous systems is able to automatically generate touch gesture models from given gesture examples and that is able to perform gesture recognition and authentication tasks. Hence, the goal of removing the tedious and difficult manual creation and parameterization process for CHnMM based systems could be achieved. The results show that the system works very well in recognizing different gestures by different users. Even the discrimination of gestures that only differ in execution speed achieved respectable results.

As a summary, the following list gives an overview of some special properties of the new recognition system that could be advantageous for different application:

- Gestures are defined by examples
- The tolerance and accuracy of the recognition system is configurable with parameters
- The training and recognition processes are computationally very fast
- The gesture model creation and the recognition is independent of the touch data frequency and therefore independent of the device and platform used (unlike HMM-based systems)
- The system does not attempt to always classify an executed gesture, hence, only gestures that really are represented by a gesture model are detected
- Due to the early cancelation abilities of the system, it has the potential to work with good performance even on very large gesture sets

Of course, there are also some current limitations to the system, for example the current prototype is not translation, rotation or scaling invariant, although translation invariance could be easily achieved by using coordinates relative to the start of the gesture. However, it is not clear if these invariances are desirable in authentication scenarios.

Since the implementation of the system is only a prototype there are still many aspects that need to be investigated, for example:

- More sophisticated approaches to define the size of the tolerance areas should be employed that also consider the number of examples (more examples → smaller tolerance area)
- The circle generation could calculate the smallest circle enclosing all area points
- Different area shapes like a polygon could be employed
- More specific probability distributions depending on the use case
- Instead of a fixed number of areas, an area point could be generated every time a certain distance of the gesture trace is reached, to better cope with different lengths of the example gestures
- The output probabilities for symbols could be adapted according to the number of example gestures

The proposed idea of using a *StrokeMap* and a CHnMM to model touch gestures, which is presented in this paper, could be easily extended to a more general concept where paths and trajectories are modelled that are subject to variations in their execution and can differ in their temporal dynamics, which is the case for many human movements. Therefore, the presented approach could also be used for gestures that are performed with a stylus device or, in combination with image recognition techniques, it could also be applied for gesture recognition from camera-recorded movements. The concept is also easily extendable to three dimensional paths and trajectories (e.g. for Wiimote or Kinect gestures) and also multi-path abilities are a subject of future research, hence the applications could be manifold.

REFERENCES

Bosse S., Krull C., Horton G., 2011. MODELING OF GESTURES WITH DIFFERING EXECUTION SPEEDS: Are Hidden non-Markovian Models Applicable for Gesture Recognition. Proceedings of the 10th International Conference on Modelling & Applied Simulation (MAS). 189–194. 12th-14th September. Rome, Italy.

Bosse S., Krull C., Horton G., 2012. SUPERVISED TRAINING OF CONVERSIVE HIDDEN NON-MARKOVIAN MODELS: increasing usability for gesture recognition. The 11th International Conference on Modeling and Applied Simulation, 106–111, Vienna

Buchholz R., 2012. Conversive Hidden non-Markovian Models. Dissertation. Otto-von-Guericke-Universität Magdeburg.

Chen F., Fu C., Huang C., 2003. Hand gesture recognition using a real-time tracking method and hidden Markov models. Image and Vision Computing, Volume 21. 745–758

Damaraju S., Kerne A., 2008. Multitouch Gesture Learning and Recognition System. Interface Ecology Lab at Texas A&M University.

Dittmar T., Krull C., Horton G., 2015. A new approach for touch gesture recognition: Conversive Hidden non-Markovian Models. Journal of Computational Science. Available from: http://dx.doi.org/10.1016/j.jocs.2015.03.002 [accessed 16 March 2015

Findlater L., Wobbrock J. O., Wigdor D., 2011. Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. In Proceedings of the

2011 annual conference on Human factors in computing systems. 2453–2462. New York

Fink G. A., 2014. Hidden Markov Models. In: Markov Models for Pattern Recognition. Springer, 71–106

Hong P., Turk M., Huang T. S., 2000. Gesture modeling and recognition using finite state machines. In Automatic face and gesture recognition, proceedings. fourth ieee international conference on. 410–415

Joselli M., Clua E., 2009. gRmobile: A Framework for Touch and Accelerometer Gesture Recognition for Mobile Games. Games and Digital Entertainment (SBGAMES), 2009 VIII Brazilian Symposium on. 141–150. Rio de Janeiro

Krull C., Horton G., 2009. Hidden non-Markovian Models: Formalization and solution approaches. Proceedings of 6th Vienna International Conference on Mathematical Modelling. Vienna.

Lazarova-Molnar S., 2005. The Proxel-Based Method: Formalisation, Analysis and Applications. Dissertation. Otto-von-Guericke-Universität Magdeburg.

Sae-Bae N., Ahmed K., Isbister K., Memon N., 2012. Biometric-Rich Gestures: A Novel Approach to Authentication on Multi-touch Devices. Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems – CHI. New York

Shabtai A., Fledel Y., Kanonov U., Elovici Y., Dolev S., Glezer C., 2010. Google Android: A Comprehensive Security Assessment. IEEE Security and Privacy 8: 35–44

Sherman M., Clark G., Yang Y., Sugrim S., Modig A., Lindqvist J., Oulasvirta A., Roos T., 2014. User-generated free-form gestures for authentication: Security and memorability. Proceedings of the 12th annual international conference on Mobile systems, applications, and services. 176–189