USING CONVERSIVE HIDDEN NON-MARKOVIAN MODELS FOR MULTI-TOUCH GESTURE RECOGNITION

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

[(a)(b)(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
With the current boom of multi-touch devices the recognition of multi-touch gestures is becoming an important field of research. Performing such gestures can be seen as a stochastic process, as there can be many little differences between each execution. Therefore stochastic models like Hidden Markov Models have been already utilized for gesture recognition. Although the modelling possibilities of Hidden Markov Models are limited, they achieve an acceptable recognition quality. But they have never been tested with gestures that only differ in execution speed. Therefore we propose to use Conversive Hidden non-Markovian Models for multi-touch gesture recognition. This extension of Hidden Markov Models enhances the modelling possibilities and adds timing features. In this work two multi-touch gesture recognition systems were developed and implemented based on these two model types. Experiments with a set of similar gestures show that the proposed model is a good and competitive alternative and can even be better than Hidden Markov Models.

Keywords: HMM, CHnMM, gesture, recognition

1. INTRODUCTION

1.1. Background

Due to the big success of smartphones and tablets a ubiquitous presence of multi-touch devices is establishing itself around the world. While this multi-touch input method offers manifold possibilities to control these devices, almost all of them are usually controlled by using a fixed set of simple gestures like tap, drag and pinch. Such systems can be realized quite easily using heuristics but they are not very flexible.

To create a gesture recognition system using heuristics means that the system has to have all of the gestures previously implemented. Adding a new gesture afterwards could make code adaptions of the previously implemented gestures necessary, especially when the gestures are very similar. In order to create a more flexible gesture recognition system other methods need to be used that define a gesture by providing some performed examples. This way the user of a touch device could define the gestures that suit him the most for certain actions.

An existing flexible gesture recognition system for multi-touch devices is presented in the work of Damaraju and Kerne (2008). The system is based on Hidden Markov Models (HMM) and creates one HMM for each gesture according to sample inputs.

While there are other pattern recognition methods that are deployed for multi-touch gesture recognition, the current work focuses on HMM and on a quite new model class: Conversive Hidden non-Markovian Models (CHnMM). With this work we want to evaluate whether CHnMMs are applicable for multi-touch gesture recognition and how they perform in comparison to HMMs.

1.2. Motivation and Goals

The idea to use non-Markovian Models for gesture recognition was first brought into consideration by Bosse et al. (2011). The goal was to show that a system based on Hidden non-Markovian Models (HnMM) could distinguish gestures that are similar in shape but differ in execution speed. This differentiation has not been considered for HMM gesture recognition systems before. For that reason an HMM- and an HnMM-system were developed to recognize gestures performed with a Nintendo Wiimote and both systems were compared in their recognition quality.

Inspired by this, we want to apply a similar approach to multi-touch devices. Furthermore CHnMMs were used instead of HnMMs whose properties and differences are explained in Section 2.2 in more detail.

The general goal of this paper is to find out whether CHnMMs are applicable for multi-touch gesture recognition. Therefore the CHnMM-system needs to reach similar or better recognition rates than the HMM-system. Additionally the time needed for recognizing the gesture has to be competitive, so that the system could be used in real-time scenarios.

2. BACKGROUND INFORMATION

2.1. Hidden Models

In the context of this work the term Hidden Models stands for all model classes that are based on the concept that was introduced with Hidden Markov Models: inferring conclusions from a list of observations of a hidden system, a so called partially observable discrete stochastic (PODS) system (Buchholz 2012). The stochastic behaviour of such a system is known but its discrete states cannot be directly observed. Instead only arbitrary "messages", so called symbols or signals, which are created by the system, can be recognized. Since the creation of these symbols is incorporated into the model, it is possible to infer conclusions about the state of the real system.

There are some common problems that can be solved with Hidden Models and two of them are relevant to this paper: Evaluation and Training. Evaluation is the task of calculating the probability that a trace O (a sequence of observations) was produced by the hidden model λ, formally $P(O \mid \lambda)$. This probability is often used to find out which model of a set of models was most likely responsible for a certain trace. The Training task is used to find a model λ, so that the result of the Evaluation task reaches a maximum according to a given trace O or a set of traces respectively:

$$\arg\max_{\lambda} = P(O \mid \lambda) \quad (1)$$

Since it is very difficult to create the best model by only knowing the trace, the training is often done by improving an already given model so that $P(O \mid \lambda_{new}) > P(O \mid \lambda)$.

2.2. Related Work

In HMMs the hidden model is a simple discrete time Markov chain with additional information about symbol creation in each of its states. So with every discrete time step a certain symbol will be created depending on the current state. Each possible symbol in that state has a fixed probability to be created. This model class has an obvious limitation: only Markovian systems can be modelled. Most real systems are not Markovian though and therefore a lot of extensions to HMMs are being developed that try to circumvent its limitations.

One of the latest developments in this research area has been achieved by Krull et al. (2009) with Hidden non-Markovian Models (HnMM), a powerful model class that is able to represent non-Markovian discrete systems like stochastic petri nets and combine them with the abilities of Hidden Models. This modelling power comes at the cost of complexity which can be reduced by focusing on a subclass of HnMMs. Buchholz (2012) has researched such a subclass in his work and named it Conversive Hidden non-Markovian Models (CHnMM). In this subclass the modelling potential is slightly restricted but the efficiency of computing solutions increases. For example transitions are only allowed if they produce a symbol. For this reason every state change can be recognized and the calculation of the solution does not have to cope with hidden state changes.

CHnMMs have already been employed in Bosse et al.'s (2011) work, even if they were not explicitly named as such. Furthermore they could not utilize the findings of Buchholz (2012) because they were simply not known at that time. Since this work is inspired by the work of Bosse et al. (2011), we have chosen to use CHnMMs as well. But in contrast to them we can also rely on the optimized algorithms and definitions that were developed by Buchholz.

2.3. Environment

For the experiments, a multi-touch tabletop, provided by the "User Interface & Software Engineering" (UISE) working group (at the computer science department of the Otto-von-Guericke-University Magdeburg), was used which was based upon the FTIR principle (Han 2005) and provided the touch data via the TUIO-protocol. Also a Java class by the UISE group that merges the TUIO messages to three simple events (stroke{Started, Updated, Finished}) was available. The software that processes camera images to TUIO messages has a frequency of approximately 15ms.

3. IMPLEMENTATION

In this section important aspects are covered that explain how both recognition systems are implemented and how they differ in certain details.

3.1. Symbol Generation

Symbols are very important for hidden models, because traces of them are used to deduce conclusions such as whether the trace was created by a certain model (evaluation) or which path of states the real system had probably taken (decoding). The symbol set used in this work consists of eight discrete directions: Up, UpLeft, Left, DownLeft, Down, DownRight, Right, UpRight. Each symbol represents the current direction of movement of a certain finger that touches the device, which is inspired by the symbol set used in the last experiment of Bosse et al. (2011).

Although both systems use the same symbol set, the way the symbols are created is different. For HMMs the symbol emission is connected to states, whereas with CHnMMs it is connected to state transitions. Therefore the emission for HMMs needs to be periodic in order to get clues about the current state. In contrast, a CHnMM works with symbol emissions that occur when the state of the system changes.

As a result the generation of symbols for the HMM system is done as follows: every time a strokeUpdated event is received, the vector from the last known position to the current position is calculated and the direction of this vector is discretised to one of the eight directions defined in the symbol set. This direction is directly emitted as a symbol after calculation. It is done this way for every active finger separately, i.e. two

moving fingers would lead to the emission of two direction symbols per processing step. Timestamps of the emissions are not needed or cannot be used for HMMs although it seems to be an important source of information for gesture recognition with different execution speeds.

The calculation of symbols for the CHnMM-system is quite similar, but the symbols are only emitted if the current state or direction of the finger changes or if the finger touches the surface for the first time and therefore no previously moving state of the finger is known. This way a finger that moves in one direction should not emit any symbols until the direction changes. But in practice the data of a straightly moved finger can be noisy, so much so that symbols are emitted although the direction of the movement has not changed. To reduce this noise in the symbol stream, stroke updates are not considered for symbol emission if the calculated direction vector is shorter than an empirically defined threshold. For CHnMMs the timestamp is essential and it is taken from the last known point of a stroke that determines when this point was received as opposed to the time when the symbol was calculated.

Furthermore the two symbols GestureStart and GestureEnd were used for both systems to determine when the execution of a gesture starts and when it ends. For this work we determined that a gesture starts with the first touch of a finger and ends when the last finger is removed from the surface.

3.2. Modelling

A main aspect of both systems is the creation of models as this is what the recognition method is based on. There are quite a lot of possibilities to do the modelling for the recognition systems, e.g. the scope of what a model represents could be one of the following:

- A model represents the whole set of possible gestures
- A model only represents a single gesture
- Two or more models represent a single gesture

The last concept was used in the work of Bosse et al. (2011) where a single gesture was modelled with three models, one for each axis of the acceleration sensors. Our approach is to use one model per gesture and this is the case for both recognition systems.

The next step is to define what the states of the model will represent. Since symbols are used that give information about the direction of movement, it seems reasonable to separate a gesture into phases or states of movement directions. For example, a gesture consisting of a movement to the left and back to the right will be modelled with a single state for each of the movement phases left and right. While the HMM only consists of these movement states, the CHnMM is made of two additional states, namely the Start- and End-state. The Start-state is needed because CHnMMs are state change oriented, i.e. symbols are emitted when a state change occurs. If there were no state before the first movement phase, the first direction symbol could not be incorporated into the model, which would waste important evidence of the current state of the real system. The purpose of the End-state is mainly the ability of the model to refuse incomplete gestures. More details on that are given in Section 3.4.

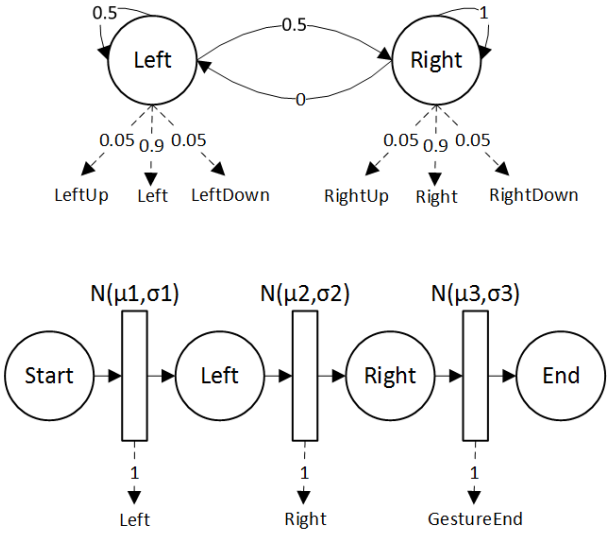

Figure 1: Example Models for a left-right Gesture (HMM: top, CHnMM: bottom)

Of course all of these aforementioned states need to be connected. For HMMs the connection is expressed with fixed probabilities that state how likely it is to stay in a state or to change to another one in every discrete time step, i.e. in this case with every periodic symbol emission. The CHnMMs however are connected with transitions and each of them is described by a certain probability distribution that determines when a state change is happening. We assume that the time for a certain state change between two movement phases is normal distributed, so each transition needs an expectation value and a standard deviation.

Figure 1 shows how an HMM and a CHnMM for a left-right gesture could look like. The dashed arrows mark possible symbol emissions and their probability. It can also be seen that the HMM only has fixed probabilities for state transitions whereas the CHnMM uses normal distributions in this case. Therefore an HMM is dependent on the discrete time step it was created or trained with and as a result will not work reliable with other devices that provide data with a different time step. Note that the HMM in Figure 1 tolerates noisy symbols by giving them a low probability to occur. For the CHnMM case the filtering is not done on model level but on symbol generation level as it was explained in Section 3.1.

It should be understandable now, how a certain gesture can be modelled with either an HMM or CHnMM. The last step to get a usable model is the training, which will be discussed in the next section.

3.3. Training

After a model for a gesture has been developed the parameters like state change and symbol emission probabilities need to be adjusted, so that they fit the finger movement of a user performing the gesture. This section describes how this was done for HMMs and CHnMMs.

The first step is to collect example data of the gesture, so it is performed twenty times with the HMM symbol generation and another twenty times with the CHnMM symbol generation method. Since a Java framework for HMMs is used, namely jahmm (François 2009), the training is simply done by creating a base model of the gesture (as it was described in the previous section) and executing the provided training algorithm with the example traces. The framework uses the Baum-Welch algorithm, a standard training algorithm for HMMs. With every iteration of this algorithm the model approaches a local optimum. That is why it is important to have a good base model to start with.

However the training for CHnMMs is more complex and cannot be done with the usual Baum-Welch algorithm. Buchholz (2012) developed an adapted version for CHnMMs that is nearly as powerful but sometimes creates even worse models. We therefore decided to train the CHnMMs manually similarly to how Bosse et al. (2011) proceeded in their works. This meant that sample traces were put into an excel sheet, movement phases or states were respectively identified and the expectation value and standard deviation for the normal distribution were calculated.

Eventually the process of creating ready-to-use HMMs and CHnMMs was complete.

3.4. Classifying the executed gesture

In this section the process of classifying an executed gesture are elaborated. This is the last part of the implementation before the experiments can be conducted.

The classification with HMMs is done by calculating the probability that the trace of the executed gesture O was created by the gesture model λ (evaluation) for each known gesture model. There is a function in the jahmm framework that does the evaluation based on the forward algorithm. The gesture, whose model created the highest probability, was returned as the classification result.

For the CHnMM system a similar approach was used but the fact that there is an End-state in each model is exploited. So instead of calculating $P(O|\lambda)$ the forward probability of the End-state at the point of time of the last symbol emission T is taken as a measurement for the likeliness of a gesture. Formally this forward probability is defined as

$$\alpha_T(s_{end}) = P(O \cap q_T = s_{end} \mid \lambda) \qquad (2)$$

with s_{end} being the End-state and q_T being the state of the model at time T.

With this approach only traces are considered that reached the End-state and therefore are likely to be complete gestures whereas the evaluation approach would also accept incomplete gestures. For example a trace of a movement to the left could be classified as a left-right gesture, because its model can create such a trace.

This technique could be easily used with HMMs too, but the jahmm framework is not suited for this technique. However, for the goal of this paper this circumstance is not a problem, because no incomplete gestures were executed in the experiments.

4. EXPERIMENTS AND RESULTS

Both implemented systems need to be evaluated in order to compare their recognition quality and performance. Since the recognition systems are classifiers, the commonly used Precision/Recall metric can and will be used, although the classifiers are not binary. The work of Sun and Lim (2001) shows how this metric is used for n-ary classifiers.

The precision of gesture G equals the number of gestures correctly classified as G divided by the number of gestures classified as G. Whenever the classifier returns G the precision value of this gesture states how trustworthy this result is. The number of gestures correctly classified as G divided by the number of performed G gestures is the recall of gesture G. It is a measurement of how complete the classification is.

For the experiments also a fixed set of gestures is needed. Figure 2 shows all six gestures that were chosen. They can be split into two groups of similar gestures: three DownUp and three circular gesture variants. These gestures are special, because they are similar in shape but different in execution speed.

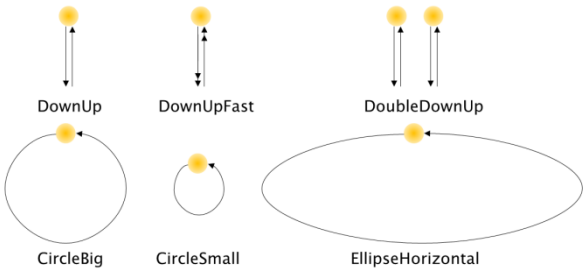

Figure 2: Gesture Set used for the Experiments

The expectation for the experiments is that the HMM-system will have problems to distinguish these similar gestures, whereas the CHnMM-system will have acceptable error rates. The needed computation time is expected to be quite similar. However the CHnMM-system might be even faster, because the number of computation steps is smaller due to the lower number of symbols created with the CHnMM-system.

4.1. First experiment

For the first experiment, each gesture was performed twenty times for each system and the symbol traces were recorded to produce training data for each gesture

and system. With this data HMMs and HnMMs were created as described in Section 3 for each gesture.

The result is an HMM and a CHnMM recognition system for the defined gesture set. Again each gesture was performed twenty times for each system and the classified (most likely) gesture was recorded. In the case that no most likely gesture is present (all probabilities are zero) this attempt is recorded as not classified.

Table 1 presents the results obtained for the HMM-system that shows an overall precision of 0.737 and a recall of 0.725. These values are worse than usually measured with HMM gesture recognition systems. Especially the DownUp and DoubleDownUp gestures were problematic and could not be distinguished properly. Also the EllipseHorizontal gesture was classified as CircleBig five times. Other gestures, however, reached acceptable recognition rates. In only two cases no classification could be made, probably due to symbol traces that did not occur while training. This phenomenon is also known as overfitting.

Table 1: Results with HMM-System

Gesture	Correctly classified	Not classified	Pre-cision	Re-call
DU	10	-	0.526	0.500
DUF	17	-	1.000	0.850
DDU	9	2	0.409	0.450
CB	19	-	0.704	0.950
CS	17	-	0.944	0.850
EH	15	-	1.000	0.750
	87	2	**0.737**	**0.725**

The results of the CHnMM-system are presented in Table 2 and show a precision of 0.955 and a recall of 0.883, which are clearly better than the ones of the HMM-system. The DownUpFast gesture has the worst recall, because it was classified four times as a DownUp gesture which is why its precision is lower than the rest. The reason for this is probably that the gesture was performed faster for training than it was performed for classification.

Table 2: Results with CHnMM-System

Gesture	Correctly classified	Not classified	Pre-cision	Re-call
DU	20	-	0.833	1.000
DUF	15	1	1.000	0.750
DDU	19	1	1.000	0.950
CB	19	1	0.950	0.950
CS	16	4	1.000	0.800
EH	17	2	1.000	0.850
	106	9	**0.955**	**0.883**

Another observation is that the general precision is better than the recall. This basically means that you can have a good trust in the classification results but that not all gestures were classified. For real applications where a certain recognized gesture activates a certain command or behaviour the precision needs to be especially high. Otherwise a gesture is performed and the wrong command or behaviour is executed which will lead to a bad user experience.

4.2. Experiment with Training Data

As mentioned in the section before, a possible source of error is the discrepancy of gestures performed during training and classification. To avoid this, the experiment was reconducted, but instead of performing gestures the training data was used as input.

The results that can be seen in Table 3 and Table 4 show much better precision and recall values for both systems as expected. The CHnMM-system even reached a perfect precision; its recall though is slightly lower than the one of the HMM-system. This is because the gesture models of the CHnMM-system were modelled quite strictly, i.e. they did not take noisy data, which occurred a few times in the training, into account. This way the time consuming manual training was performed a little bit easier.

Table 3: Results with HMM-System and Training Data

Gesture	Correctly classified	Pre-cision	Re-call
DU	20	0.741	1.000
DUF	20	1.000	1.000
DDU	13	1.000	0.650
CB	20	0.952	1.000
CS	19	1.000	0.950
EH	20	1.000	1.000
	112	**0.933**	**0.933**

Table 4: Results with CHnMM and Training Data

Gesture	Correctly classified	Not classified	Pre-cision	Re-call
DU	20	-	1.000	1.000
DUF	20	-	1.000	1.000
DDU	18	2	1.000	0.900
CB	20	-	1.000	1.000
CS	16	4	1.000	0.800
EH	16	4	1.000	0.800
	110	10	**1.000**	**0.917**

Besides the good overall precision and recall values, the HMM-system had huge problems recognizing the DoubleDownUp-gesture. The reason seems to be that the HMMs of DownUp and DoubleDownUp are nearly the same although DoubleDownUp gesture emits quite exactly the doubled number of symbols. However, a HMM cannot express this movement because the probability to change from Down-state to Up-state is approximately the same in

both gestures. This also explains the problems with this gesture in the first experiment and also shows a weakness of HMMs, even if it recognized the other gestures better than expected.

In both experiments the time from finishing the gesture to getting the classification result was measured. This time was always 0ms if the input data could not be classified. Otherwise the values were mostly 31 or 32ms and sometimes 47ms for both recognition systems. Since the reaction times appear fixed, we assume that this is because of the scheduling side-effects of the operating system (Windows) and that the pure calculation time of the recognition systems is even lower.

5. CONCLUSION

The goal of the paper was to show that CHnMMs can be used in the field of multi-touch gesture recognition and that they are an alternative to HMM recognition systems. The results of the experiments show that the CHnMM-system reached better, if not the same level of recognition quality as the HMM. Additionally both systems are on par regarding recognition speed, which makes CHnMM-systems also usable for real-time scenarios. Therefore all goals of this work have been achieved.

Further observations suggest that HMMs are not suited to distinguish certain gestures apart from others, as was seen with the DownUp and DoubleDownUp gestures, where both are described by nearly the same model. Nevertheless the HMM-system worked better than expected. It can be assumed that although HMMs are timeless, they get fed with timing information through the periodic emission of symbols (approx. every 15ms in our HMM experiments). This way they are able to even distinguish similar gestures with different execution speeds. Systems that do not emit symbols periodically will have more problems with that when using HMMs.

However a big advantage of HMMs is that they can be trained automatically with the Baum-Welch algorithm, making them easier to implement in comparison to manual training. For CHnMMs an efficient training algorithm was developed based on the Baum-Welch one by Buchholz (2012). The quality of this training algorithm in the field of multi-touch gesture recognition needs to be evaluated in future work though.

Further experiments and extensions to this work could be:

- a bigger gesture set,
- using other symbol generation methods,
- more complex multi-touch gestures,
- comparison to other gesture recognition methods (Artificial Neural Networks, Dynamic Time Warping, Support Vector Machines etc.).

As a consequence, CHnMMs could be used to develop flexible multi-touch recognition systems that are independent of the frequency of symbol emission and therefore are independent of the device. Furthermore there are less symbols generated resulting in lesser calculations than HMM or similar systems. This and the time step independency make a CHnMM recognition system suitable for mobile multi-touch devices.

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)*, 12th-14th September. Rome, Italy.

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

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

François, J.-M., 2009. *jahmm.* Google Code. Available from: http://code.google.com/p/jahmm/ [29. September 2012]

Han, J. Y., 2005. Low-cost multi-touch sensing through frustrated total internal reflection. *Proceedings of the 18th annual ACM symposium on User interface software and technology (UIST '05)*, pp. 115-118, New York, USA.

Krull, C., Horton, G., 2009. HIDDEN NON-MARKOVIAN MODELS: Formalization and solution approaches, *Proceedings MATHMOD 09 Vienna*. Wien.

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

Sun, A., Lim, E.-P., 2001. Hierarchical Text Classification and Evaluation, *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM 2001)*, pp. 521-528, California, USA.