Mouse Gesture Recognition for Human Computer Interaction

Krupalu Mehta, Meet Patel, Hardik Somaiya, Dhwanil Shah
Asst. Prof. Bhakti Palkar
Computer Department,
K. J. Somaiya College of Engineering,
Mumbai, India

Abstract: In the field of Computer Science and Information Technology, it goes without saying that the focus has been shifted from the System Oriented software to the User Oriented software. Naturally, for such software applications, the importance of User Experience has gained a paradigm shift. The paper highlights the significance of a seamless interaction between the user and the computer by proposing a reliable algorithm for performing basic operations by drawing gestures with a mouse. It aims to embrace simplicity and quick access using gestures and providing effortless interaction for the uniquely abled users. The core of the algorithm comes from the Hidden Markov Model, which is emblematic of a probabilistic approach for gesture recognition.

Keywords: Hidden Markov Model, Mouse Gesture Recognition.

*****

I. INTRODUCTION

This paper is a study and implementation of how the Hidden Markov Model can be used to build a reliable gesture recognition system.

1.1 MOTIVATION

Improvising the interface between human and computer is one of the major motivations for building such type of a system. Providing quick access for the often-used tasks, enabling user to customize gestures and receive real-time feedback after making the gestures are some traits that drove us to implement this system, which aims to bridge the gap of visual communication between the user and the computer.

1.2 OVERVIEW

In the background section, the reader is introduced with Pattern recognition and Hidden Markov Models. In the Gesture Recognition topic, fundamentals of the Gesture Recognition process are discussed including the previous work done on it. The Parameter reestimation and the recognition algorithms are discussed thereafter. Implementation of a Mouse Gesture Recognition made in Java is elaborated in the final section.

II. BACKGROUND

Before discussing how the gestures are recognized using a Hidden Markov Model, it is necessary to learn about the basics of Pattern Recognition, Gesture Recognition and Hidden Markov Model.

2.1 PATTERN RECOGNITION

Mathematically, pattern recognition is a meticulous process of classifying objects into a number of classes. There are usually two types of patterns, supervised and unsupervised pattern recognition. Supervised pattern recognition is the one in which the classes of the example patterns are already known. In unsupervised classification, the example patterns or the classes are not known on a priori.

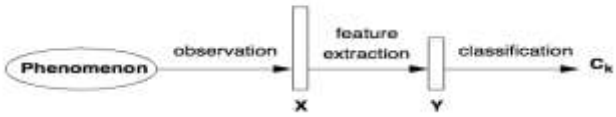

Fig. 1

In the Fig. 1, the phenomenon to be classified is observed heedfully and the result is n number of observations, each represented by *x* (observation vector). Features extracted from the vector x, are represented by *y* (feature vector). Classification can be done either by using the K-Means or any other reliable algorithm. The classified data is nothing but the generated classes, or in the context of gesture recognition, the Gesture Classes. Generally, in pattern recognition, recognition is always unsupervised, as example classes may vary highly from the user input data.

2.2 HIDDEN MARKOV MODEL

Probability is a strong element while designing systems for gesture recognition. The foundation of the Hidden Markov Model (HMM) is the identification of the transition probability from one state to another. A *Hidden Markov Model* (denoted λ) is a doubly stochastic process. The first stochastic layer is the underlying first-order Markov process, represented by the state transition diagram of Fig. 2 Each state is a possible observation of the Markov process, and a transition probability from state *A* to state *B* is $P(s_{t+1} = B \; s_t = A)$, the probability of

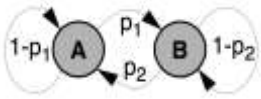

Fig. 2

going to state *B* at time *t*+1 given that the state at time *t* is *A*. The second stochastic layer of the HMM is the set of output probabilities

for each state. [1]

Training algorithms are used so that the HMMs can be trained such that, the wrapper is lifted from the good probability which leads to identification of the actual sequence of states *s* from the observation vector *x*. Hence, for the training purpose, we use the Baum-Welch algorithm and for the testing and recognition, the Viterbi algorithm.

III. GESTURE RECOGNITION

Gesture recognition is our final goal; hence we first define it and then discuss the previous work done.

3.1 DEFINITION

Here, we only scientifically define the gesture. Within the system, the gestures are defined as a set of combination of ambit curves and lines, which are identified by extraction of features from the observation vector. In the above definition, there are a number of assumptions. The number of possible gestures is restricted to the feature space. Hence, the efficacy of the gestures is highly dependent on the features. Gestures can be also defined as a sequence of transitions between perceptual states (states defined by feature measurements).

3.2 PREVIOUS WORK

Earlier work related to HMMs was limited to handwriting recognition for biometric authentication purposes [4]. Latterly, Starner also used HMM speech recognition product, to build a real-time handwriting recognition system [5].

Yamato, *et al* [3] used discrete HMMs to successfully recognize six different tennis swings among three subjects. Though the system used only 25x25-pixel images as the feature vector, it could decently distinguish the different motions. However, in order to use the discrete HMM, the domain was constrained enough to avoid the effects of vector quantization distortion. Furthermore, the system performed only isolated gesture recognition, in which gestures have already been temporally segmented from video.

The system described by Cui, Swets, and Weng [7] was "a self-organizing framework... for learning and recognizing spatiotemporal events (or patterns) from intensity image sequences." They applied this system to hand sign recognition of a single hand. Though they wisely distinguished between the most expressive and most discriminating features, all features were essentially eigenvector coefficients. The example gestures were used to partition the visual space, where each partition had its own eigenvectors. The gesture was assumed to have been isolated in a temporal window from which 5 frames were sampled to represent the gesture. They claimed 96% recognition rate of a simple vocabulary; however, they also admitted to hand-tweaking a threshold to achieve this. Though it is not based on HMMs, it is instructive to see how much complexity HMMs incorporate by examining a system similar to this thesis that does not use them.

Two papers describe the research leading to the conception of this thesis. In the first paper, Bobick and Wilson [8] defined a gesture to be a sequence of states in a measurement space. Unlike HMMs, this model allowed the calculation of a prototype trajectory to represent the gesture. This prototype was used to align the state pdf's along the most-likely direction by explicitly defining two types of variance, the along-trajectory variance and the across-trajectory variance. This approach is significantly different from HMMs because the prototype allowed tracking of the gesture within a state, whereas HMMs cannot model the motion within a state.

In the second paper, Wilson and Bobick [8] used a state-membership measure to combine multiple representations at each state. Re-estimation of the HMM parameters via the Baum-Welch algorithm was interleaved with re-estimation of the representation parameters.

3.3 SUMMARY

Gesture recognition depends on the definition of gesture. In a scientific manner, this paper defines gesture solely on the basis of feature measurements. This causes system performance to rely heavily on the quality of the features extracted but also allows reduction of gesture recognition to a purely computational form.

IV. HMM ALGORITHMS

4.1 PARAMETER REESTIMATION

Given a set of examples X for a particular gesture, an ideal system will create the HMM λ that most likely (as opposed to other HMMs) generated those examples. In other words, it finds λ such that $P(\lambda \mid X)$ is maximized. Such a system seems intractable, however. Instead, given the examples and a particular HMM, the Baum-Welch algorithm attempts to change the HMM parameters so that the given HMM most likely generated the examples (as opposed to other examples), *i.e.*, it maximizes $P(X \mid \lambda)$. Although this procedure is far from ideal, it has been proven to converge to locally optimal parameters. Hence we use the extended Baum-Welch algorithm [9] to reestimate both the HMM and codebook parameters.

4.2 RECOGNITION ALGORITHM

While training the HMM is an involved process, performing recognition is much simpler. Given an observation, all HMMs are scored based on how well they describe the sequence. The HMM with the highest score is chosen as the likely generator of the observation, resulting in a label of that HMM's gesture.

The Viterbi algorithm finds the single best state sequence $S\hat{} = (s_1, s_2 \ldots s_T)$ for the observation sequence $X = (x_1, x_2 \ldots x_T)$. In this study, we only took the score of the state sequence rather than the actual state sequence.

V. IMPLEMENTATION

As beginners, we have tried to implement the Baum-Welch and the Viterbi algorithms as well as possible. It came to our notice that, without the probabilistic approach using the HMMs, it was not only difficult but also time consuming to develop a reliable yet robust system for gesture recognition. Choice was dependent on how

 ISSN: 2321-8169
2517 - 2519

independent was a language was, from different platforms and to some extent, one through which we could easily manifest our understandings of the multiple mathematical algorithms. We used Java to implement the system and the Java Swing classes for User Interface Design.

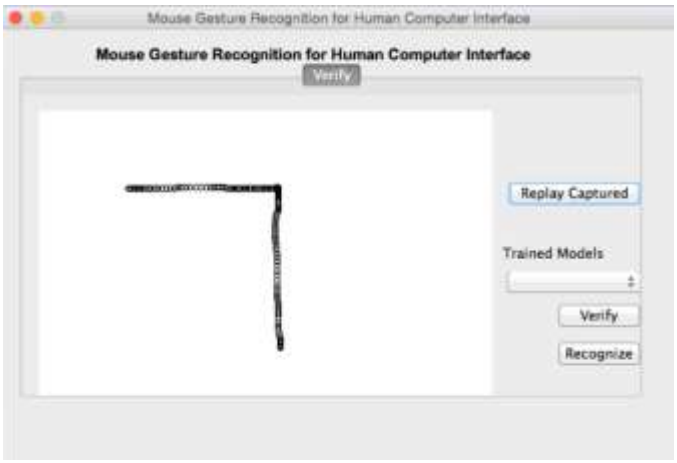

Fig. 3: Snapshot of the implemented system.

The bulk of the system relies on two portable libraries. The Matrix library provides a useful matrix abstraction for all normal types (char, int, float, double). It has been optimized for both matrix operations and memory management. Where possible, matrix operations are performed with highly efficient pointer arithmetic. In a program with many allocations and deallocations of large objects, the program's memory segment becomes fragmented, causing significant execution time to be spent on copying the objects to fit compactly in memory. In the Matrix class, normal allocation and deallocation of matrices has been overridden to allow reuse of pointers to large matrix objects, easing the memory management task of the operating system. Both optimizations have demonstrated noticeable improvement in execution speeds.

The second portable library is the MotifApp library. Built on top of the X and Motif libraries, the MotifApp library provides many classes for user interface components. For example, the OptionMenu class allows for easy construction and use of an option menu. The MotifApp library has proven to greatly simplify the complicated task of constructing a user interface. The Mouse Gesture Creator has been implemented using this library.

Verification and Recognition modules use the Viterbi algorithm to record the features of the drawn gesture in real-time. Each gesture model is stored with a ' .model ' extension, and can be trained using the Train HMM module. Generating codebooks is possible using the provided module. Replay Captured module animates the gesture exactly as done by the user.

VI. CONCLUSION

A system for gesture recognition has been designed and implemented. The initial observations of using HMMs for gesture recognition seemed noticeably difficult, but after careful implementation of the HMM algorithms, the results started improving. Thus allowed us to learn gradually how such a system can be developed using multiple mathematical algorithms having a probabilistic approach rather than going for image synthesis.

VII. ACKNOWLEGMENT

The authors gratefully acknowledge the contributions of Prof. Bhakti Palkar and thank her for her endless support and motivation, the college for providing the necessary infrastructure and platform for doing this research.

REFERENCES

[1] Donald O Tanguay, Jr., "*Hidden Markov Model Gesture Recognition*", S.B. Computer Science Massachusetts Institute of Technology, Cambridge, MA June 1993.

[2] Anis Elbahi, Mohamed Ali Mahjoub, Mohamed Nazih Omri, "Hidden Markov Model for Inferring User Task Using Mouse Movement", Research Unit MARS, Transactions on Pattern Analysis and Machine Intelligence archive, vol. 33, Issue 4, pp. 741-753, April 2013.

[3] Bassam Sayed, Issa Traore´, Isaac Woungang, and Mohammad S. Obaidat, "Biometric Authentication Using Mouse Gesture Dynamics", IEEE SYSTEMS JOURNAL, VOL. 7, NO. 2, JUNE 2013.

[4] Nikos G. Tsagarakis and Darwin G. Caldwell, "Improving Mouse-Based Computer Interaction in Users With Weak Upper Limb Motion Control Using a Haptic Assistive System", IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS, VOL. 43, NO. 2, MARCH 2013.

[5] R. Nag, K. H. Wong, and F. Fallside. "Script recognition using hidden markov models." In ICASSP 86, 1986.

[6] T. E. Starner and A. Pentland. "Visual recognition of American Sign Language using hidden markov models". In Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition, Zurich, 1995.

[7] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov models. In Proc. 1992 IEEE Conf. on Computer Vision and Pattern Recogni- tion, pages 379-385. IEEE Press, 1992.

[8] Y. Cui, D.L. Swets, and J. J. Weng. Learning-based hand sign recognition using SHOSH- LIF-M. In Proc. Fifth International Conf. on Computer Vision, pages 631-636. IEEE Press, 1995.

[9] A. F. Bobick and A. D. Wilson. Using configuration states for the representation and recog- nition of gesture. In Proc. Fifth International Conf. on Computer Vision, pages 631-636. IEEE Press, 1995.

[10]] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. IEEE ASSP Magazine, pp. 4-16, January 1986.