Accelerometer Based Real-Time Gesture Recognition

Zoltán PREKOPCSÁK[1]

[1]Dept. of Telecomm. and Media Informatics, Budapest University of Technology and Economics, Magyar tudósok krt. 2., H-1117 Budapest, Hungary

prekopcsak@tmit.bme.hu

Abstract. *Gesture is a natural expression form for humans, but its recognition is a similarly hard problem as speech recognition. In this paper, I present a real-time hand gesture recognition system, which identifies relevant parts in the continous sensor data stream, and classifies them to the most probable gesture.*
Instead of the usual button-based segmentation, I have created an automatic segmentation method, which makes the interface more natural. The results showed that the two different classifiers reach 97.4% and 96% accuracy on a personalized gesture set, and these results can be improved for certain gesture sets with the combination of the two algorithms. Furthermore, the system has great performance and low response time, so the user experience is much better than with previous gesture recognizers.

Keywords

Gesture recognition, classification, accelerometer sensor, human-computer interaction.

1. Introduction

Hand gesture recognition is a discipline that has been around for decades in the human-computer interaction research community. Its prototypes range from special applications like sign language interpretation [13] to general gesture recognizers, and they provide various hardware and software solutions for capturing and processing human hand motion. Lately, the Nintendo Wii game console popularized the concept with its gesture-based controller. Using Wii Remote and Nunchuk, games can be controlled by natural gestures which made the console very popular, especially for sports games.

In the last few years, the fast development in sensor technology made it possible to put small accelerometer sensors into everyday devices. The Wii Remote uses these sensors to recognize hand gestures, and there are many mobile phones on the market that also have in-built accelerometers. The main advantage of using these sensors instead of vision-based methods is that the interface is not limited to a controlled environment.

One of the most important visions of computer science is ubiquitous computing, which involves ambient intelligence built into our surroundings. To reach this vision, we need intuitive interfaces using speech, touch and gestures.

2. Related Works

The first dynamic gesture recognition system was created by Hofmann et al. in the middle of 1990s. They have used dimensionality reduction and discrete hidden Markov models (HMM) to reduce complexity, but recognition took hours to complete [4]. Other researchers developed simpler task-specific recognizers which could operate in almost real-time mode, but these systems cannot be used in general cases [7][1].

Researchers from the Technical Research Centre of Finland (VTT) have published several articles in the past few years about general dynamic gesture recognition, mainly for mobile phone interaction. They have created a hardware module for capturing hand movements called SoapBox [12], and they have found that left-to-right HMMs provide very accurate models [8].

Although gesture-based games spread fastly all around the world, there has been very few comprehensive studies about everyday use of gesture interfaces from the design perspective. Korpipää et al. have created a context-aware interface prototype that included hand gesture recognition for smart phones, but the rule-based setup concept was quite complicated for first-time users. Their earlier works included more prototypes for specific applications [6].

My research lies in the intersection of the technical and design perspectives, as I have identified basic design principles, and created new segmentation and modified classification methods to satisfy them. This paper describes the research from the technical point-of-view, so for design aspects, please read my earlier paper [10].

3. System Overview

A gesture recognizer prototype was created for testing purposes. The input device was a Sony-Ericsson W910i mobile phone which has in-built three-dimensional accelerometer accessible from Java environment. The application collects the accelerometer sensor data stream and sends it to a computer nearby via Bluetooth connection. All other components are realized on the computer, but with great respect to low usage of resources, so they can be implemented to the mobile environment too.

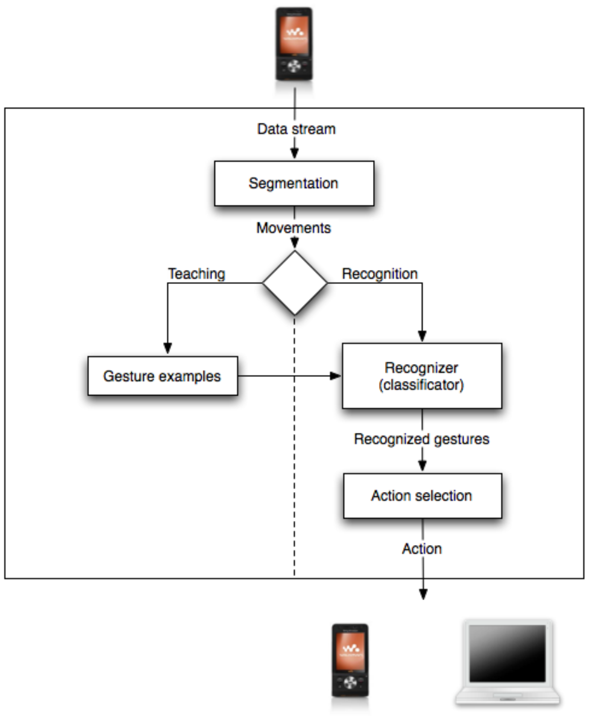

Fig. 1. System block diagram with major components.

When the data stream arrives to the computer, it passes through a segmentation module which identifies the beginning and the end of gestures. Only the segmented parts are forwarded to the next component. In case of teaching, the labelled gesture examples are just saved for future use. In case of recognition, it is passed to the classifier component, which includes noise filtering and identifies the most probable gesture. Depending on the gesture and the context we can define actions, so the gesture recognizer prototype can be used as an interface for mobile phones or computer applications. In the next subsections, I will describe these components in more detail.

3.1. Segmentation

Most gesture recognizer systems use button-based segmentation, which means that the user needs to sign the beginning and the end of the gesture by pushing a button. I have decided to create an automatic segmentation method without any buttons, which results more natural interaction. A similar approach was used by Hofmann et al. [4].

The accelerometer sensor provides about 25 three-dimensional data vectors every second. The role of the segmentation is to identify the parts in this data stream where gestures occur. To be able to solve this problem, we need the definition of gestures. In my work, I have used the following largely technical definition: a gesture starts with rapid acceleration, continuously changes directions during gestures, ends in almost steady position, and lasts more than 0.8 seconds. This definition was selected after the extensive study of the data stream of different hand gestures.

To capture gestures according to the definition, we need to do some preprocessing on the three-dimensional data stream. First of all, the definition includes statements about the changing property of acceleration, so the derivative is used, which is the difference between two consecutive elements in the discrete case. Second, the size of the vector is computed, because the system should be invariant to the order of axes.

$$H_k = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2 + (z_k - z_{k-1})^2} \tag{1}$$

This value can sign changes in acceleration and direction too, but it is very hectic so a moving average should be used. I have selected the exponential moving average (EMA), because it is fast and memory efficient.

$$EMA_{H_k} = \alpha H_k + (1-\alpha) EMA_{H_{k-1}} \tag{2}$$

I have tried several values for α, and the value 0.2 resulted smooth curves without averaging too many values. After these computations, the segmentation is as simple as monitoring a threshold on EMA_{H_k}. While the value is over the threshold, a gesture is captured. User tests proved that this segmentation method can capture dynamic gestures, and it feels natural to use them without buttons.

3.2. Gesture recognition

The output of the segmentation is a variable, but finite length vector containing three-dimensional elements, so it is a matrix sized $3 \times k$. In teaching mode it is saved to the filesystem with the appropriate label attached. In recognition mode we need to decide if the matrix is very similar to one of the gestures. Two different classification methods have been implemented for this problem.

Hidden Markov Models

HMM is a statistical modeling tool that can be applied for modeling time-series with spatial and temporal variability [5]. As I mentioned earlier, it is frequently used for gesture recognition, but it has numerous applications in speech recognition too. My algorithm was based on a recent Nokia Research Center paper [11] with some modifications. I have used the freely available JAHMM library for implementation.

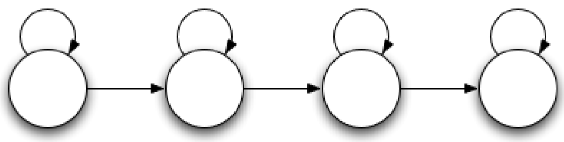

Fig. 2. Left-to-right hidden Markov model with four states

Continuous left-to-right HMMs have been created with 8 states for each gesture. Previous papers used to apply dimensionality reduction to reduce complexity, but it causes loss of information and current systems are able to handle this complexity in real-time. Pylvänäinen showed that no transformations are needed on the input data, as it is suited for the HMM technique in its raw format[11]. HMMs were trained with the iterative Baum-Welch algorithm and forward-backward algorithm was used for recognition. I do not explain these standard algorithms in this paper, but they can be found elsewhere [3].

In the recognition step, we calculate the probability of all gesture models for a given movement and select the most probable one. As the automatic segmentation also produces some noise movements, we need to include a probability threshold and below that we consider the movement being noise. If the threshold is too high then many real gestures will be considered noise and if it is too low then noise movements will be misclassified. Its value was determined according to test results.

Support Vector Machine

Support Vector Machine (SVM) is a supervised linear classification method that has the great property of maximizing margins between classes. It also has non-linear extensions in the form of kernel functions [2]. I have used the open-source LIBSVM package for implementation.

To use the SVM, we need to transform the $3 \times k$ matrix to a fixed length vector. In this case, the matrices were transformed to a 10-dimensional vector with basic statistical elements like minimum, maximum, mean, length, etc. These statistical values describe the direction and dynamics of the gesture. To make all dimensions equal, z-normalization have been used:

$$x_i' = \frac{x_i - \bar{x}}{\sigma} \tag{3}$$

where $\bar{x}$ is the mean and σ is the variance.

I have used the linear kernel with all gesture examples for teaching. During recognition, the incoming movement should be transformed and normalized the same way like the gesture examples, and then the SVM decides which gesture it belongs to.

3.3. Action selection

In the system prototype, there is a possibility to predefine actions for each gestures. These actions can be runnable scripts or applications on the computer, or even the mobile phone can be controlled via Bluetooth serial connection. AT modem commands can simulate key-presses on the mobile phone, so we can navigate and reach regularly used menu items by doing gestures. It works like a perfect mobile gesture interface in the Bluetooth range of the computer.

The above described algorithms are all designed for low resource usage, so they can be implemented to the mobile phone. In this case the gesture interface works without computers.

4. Results

To test the accuracy of the recognition, a dataset of gestures have been created. I have defined 10 different gestures, half of them are hand movements using mainly the wrist, and half of them are wider arm movements. The gestures were repeated 10 times by four different users, which provided 400 examples overall.

Part of these examples were used to train the models and the other part was used for testing. In every test, cross-validation was used to produce reliable results. We have also recorded 50 noise examples that were captured by the segmentator.

4.1. Accuracy of the classifiers

The accuracy greatly depends on the number of gesture examples used for teaching. The more examples we have, the more accurate the model will be. Both algorithms have been tested with different number of teaching examples, which is denoted by N.

Hidden Markov Models

HMMs are able to filter movements that are not similar to predefined gestures. However, some noise still gets misclassified to a gesture. These are called false positives, while the real gestures that get filtered out, are called false negatives. As it can be seen on Table 1, noise filtering is around 95% accurate, and recognition rate also rises above 95% in most cases with 4 teaching examples already. In case of 8 teaching examples, the HMM algorithm results 97.4% accuracy on average.

User	Accuracy			Noise
	N=2	N=4	N=8	filtering
U1	93.69%	97.69%	98.44%	94.1%
U2	91.25%	97.63%	98.44%	94.54%
U3	95.86%	99.53%	100.0%	96.46%
U4	82.64%	90.94%	92.67%	96.8%

Tab. 1. Accuracy of the HMM model for different users. N is the number of teaching examples.

Support Vector Machine

There is no noise filtering included in the SVM, so false positives and false negatives never occur. The SVM performs very well for low values of N, but it fails to improve as fastly as the HMM. In case of 8 teaching examples, the SVM algorithm results 96% accuracy on average.

User	Accuracy		
	N=2	N=4	N=8
U1	97.44%	98.29%	99.0%
U2	98.94%	99.24%	99.0%
U3	84.58%	90.59%	95.0%
U4	84.33%	87.9%	90.67%

Tab. 2. Accuracy of the SVM model for different users. N is the number of teaching examples.

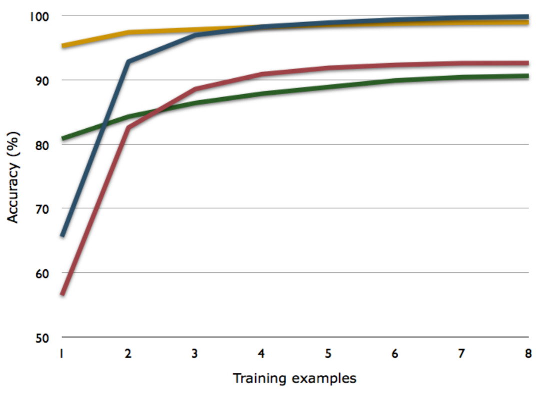

Fig. 3. Accuracy curves for different users and methods. The more teaching examples we have, the more accurate the model will be.

4.2. Error analysis

It is interesting to investigate the typical errors of the classifiers. The SVM and HMM classifiers have some misclassification errors that might be corrected with the combination of the two algorithms. Table 3 shows misclassifications where the error rate is above 0.5%. It can be seen, that there is an overlap between the errors, but in some cases only one classifier makes the mistake, so the recognition rate can be improved by the combination of the two algorithms.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	-									
G2		-						○		
G3			-							
G4				-		○				
G5					-		×	×		×
G6				⊗		-				×
G7					×		-			
G8		⊗			⊗		×	-		×
G9									-	
G10					×	×		×		-

Tab. 3. Confusion matrix for gestures. The ○ marks typical errors by HMM and the × marks misclassification by SVM.

HMMs have false positives and false negatives too. The number of false positives and negatives can be adjusted by setting the probability threshold in the HMMs. When raising it higher, there will be more false negatives and less false positives, and vice versa when lowering the threshold.

4.3. Runtime analysis

Runtime was tested on a new generation MacBook computer with a dual core 2 GHz processor and 1 GB memory. The software was developed and tested in the Eclipse 3.3 environment, Java 1.5 platform and Mac OS X 10.4 operating system. Tests were made for the teaching and the recognition phase separately. Measurements were repeated 10 times, the smallest and biggest values threw out, and averaged on the remaining 8 values.

Teaching time turned out to be linearly dependent from the number of teaching examples. HMMs were trained in $46ms$ for one example and $324ms$ for nine examples. These values were $11ms$ and $39ms$ for SVM. Recognition time was independent from the number of teaching examples and averaged at $3.7ms$ for HMM and $0.4ms$ for SVM.

On mobile platforms, these values are expected to rise with an order of magnitude, which results a few seconds of startup time, but the recognition time will be far below 100ms, which is the threshold for real-time interfaces [9].

5. Conclusion

In this paper, I have presented and evaluated a hand gesture interface prototype which could be used in industrial applications. The gesture set is completely personalizable, so everyone can use the gestures that he/she feels natural for a certain function. Instead of using button-based segmentation I have implemented an automatic segmentation method with which the interface is easier to use.

I have created two classification algorithms that provided 97.4% and 96% accuracy respectively. The HMM classifier is able to filter more than 95% of noise movements. The system has great performance and low response time, and the user experience is much better than with previous gesture recognizers.

Acknowledgements

Most of the research described in the paper was sponsored by Kitchen Budapest and I thank all the researchers and coordinators for their continuous support. I also thank my supervisors, Csaba Gáspár-Papanek and Péter Halácsy, for their helpful discussions about this research.

References

[1] BENBASAT, A.Y., PARADISO, J.A. An inertial measurement framework for gesture recognition and applications. *Gesture and Sign Language in Human-Computer Interaction, International Gesture Workshop*, 2001.

[2] BURGES, C.J.C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 1998, vol. 2, no. 2, p. 121 - 167.

[3] DUGAD, R., DESAI, U.B. *A tutorial on hidden Markov models*. Indian Institute of Technology, 1996.

[4] HOFMANN, F.G., HEYER, P., HOMMEL, G. Velocity profile based recognition of dynamic gestures with discrete hidden Markov models. *Gesture and Sign Language in Human-Computer Interaction*, 1997, p. 81 - 85.

[5] KALLIO, S., KELA, J., MÄNTYJÄRVI, J. Online gesture recognition system for mobile interaction. *IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[6] KORPIPÄÄ, P. et al. Customizing user interaction in smart phones. *IEEE Pervasive Computing*, 2006, vol. 5, no. 3, p. 82 - 90.

[7] LOVELL, S.D. *A System for Real-Time Gesture Recognition and Classification of Coordinated Motion*. Massachusetts Institute of Technology, 2005.

[8] MÄNTYLÄ, V.M., MÄNTYJÄRVI, J., SEPPÄNEN, T., TUULARI, E. Hand gesture recognition of a mobile device user. *IEEE International Conference on Multimedia and Expo*, 2000.

[9] NIELSEN, J. et al. *Usability Engineering*. Morgan Kaufmann, 1994.

[10] PREKOPCSÁK, Z. Design and development of an everyday hand gesture interface. *MobileHCI 2008 (pending)*, Amsterdam (The Netherlands), 2008.

[11] PYLVÄNÄINEN, T. Accelerometer based gesture recognition using continuous HMMs. *Lecture Notes in Computer Science (IbPRIA 2005 Proceedings)*, 2005, p. 639 - 645.

[12] TUULARI, E., YLISAUKKO-OJA, A. SoapBox: A platform for ubiquitous computing research and applications. *Lecture Notes in Computer Science*, 2002, vol. 2414.

[13] VOGLER, C., METAXAS, D. A framework for recognizing the simultaneous aspects of American sign language. *Computer Vision and Image Understanding*, 2001, vol. 81, no. 3, p. 358 - 384.

About Authors...

Zoltán PREKOPCSÁK was born in Budapest, Hungary. He is an information technology student at Budapest University of Technology and Economics specializing in data mining, machine learning and human-computer interaction. He is a researcher at the Kitchen Budapest medialab, and he participates in university research projects. Beyond being a student and a researcher, he works for a web startup company.