A $3 Gesture Recognizer – Simple Gesture Recognition for Devices Equipped with 3D Acceleration Sensors

Sven Kratz
Deutsche Telekom Laboratories
TU Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
sven.kratz@telekom.de

Michael Rohs
Deutsche Telekom Laboratories
TU Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
michael.rohs@telekom.de

ABSTRACT
We present the $3 Gesture Recognizer, a simple but robust gesture recognition system for input devices featuring 3D acceleration sensors. The algorithm is designed to be implemented quickly in prototyping environments, is intended to be device-independent and does not require any special toolkits or frameworks. It relies solely on simple trigonometric and geometric calculations. A user evaluation of our system resulted in a correct gesture recognition rate of 80%, when using a set of 10 unique gestures for classification. Our method requires significantly less training data than other gesture recognizers and is thus suited to be deployed and to deliver results rapidly.

Author Keywords
Gesture recognition, recognition rates, classifier, user interfaces, rapid prototyping, 3D gestures

ACM Classification Keywords
H5.2 [Information interfaces and presentation]: User interfaces – *Input devices and strategies.* I5.2. [Pattern Recognition]: Design methodology – *Classifier design and evaluation.* I5.5 [Pattern Recognition]: Implementation – Interactive Systems.

General Terms
Algorithms, Measurement, Performance, Experimentation

INTRODUCTION AND RELATED WORK
An increasing number of mobile devices are equipped with 3D accelerometers, which calls for suitable methods for 3D gesture recognition on these platforms. Gesture input for mobile devices can be a way to overcome the limitations of miniature input facilities and small displays, since the range of movement is not restricted by the size of the device. An example would be to perform gestures on a keypad-locked mobile phone to immediately start intended applications.

The Nintendo Wii [2] controller is a prominent and commercially successful example for a new generation of game consoles that use acceleration sensors for input to allow for more natural interaction.

Our work is based on previous work by Wobbrock et al. [8], who developed a simple "$1 Recognizer" using basic geometry and trigonometry. The "$1 Recognizer" is targeted at user interface prototyping for 2D touchscreen-based gesture recognition and therefore focuses on ease of implementation on novel hardware platforms. (The authors provide a pseudocode implementation of the complete recognizer in the paper.) We extend and modify Wobbrock et al.'s algorithm to work with 3D acceleration data. Instead of capturing exact pixel positions on a touch screen, acceleration data is of much lower quality because it is prone to noise, and additionally, drift error accumulates as the path of a gesture entry is integrated. We extend Wobbrock's original algorithm with a scoring heuristic to lower the rate of false positives. Using actual user input, we present an evaluation of the performance of Wobbrock's modified algorithm, and show that this method is well suited to implement 3D gesture recognition in rapid prototyping environments.

Past contributions [5, 7] in the area adopt established, but highly complex techniques (Hidden Markov Models, Neural Networks) for gesture recognition. Rabiner [6] is the standard introduction to implementing HMM-based classifiers. The gesture recognizers of Schlömer et al. [5] and Kratz et al. [7] feature good recognition rates, but only use a small gesture vocabulary. Moreover, these recognizers require relatively large gesture training sets. Producing repetitive movements can be a nuisance for the user. In contrast, our approach produces good results with about five training examples per gesture. Other past contributions focus on finding appropriate gestures for certain application domains (e.g., VCR control) [4] and mostly use "flat" 2D gestures.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IUI'10, February 7-10, 2010, Hong Kong, China.
Copyright 2010 ACM 978-1-60558-515-4/10/02...$10.00.

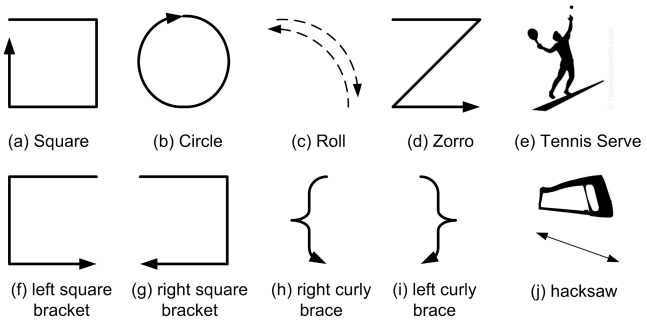

Figure 1. The reference gesture vocabulary containing the gesture classes used for the preliminary evaluation. (b) describes a clockwise circular motion, (c) a wrist rolling motion (e) stands for a gesture resembling the serve of a tennis player and (j) represents a repeated rapid forward-backwards motion.

The major contribution of this work is the creation a simple gesture recognizer that is designed to recognize "true" 3D Gestures, i.e. gestures which are not limited to shapes that can be drawn in a 2D plane. The advantage of true 3D gesture recognition is that more natural movements, such a tennis serve or boxing punches can be input by the user.

Like the "$1 Recognizer," our approach is quick and cheap to implement, does not require library support, needs only minimal parameter adjustment and minimal training, and provides a good recognition rate. It is therefore very valuable for user interface prototyping and rapid application development. It can also easily be integrated into mobile interfaces that take advantage of other modalities, like speech, or touch-based interaction with RFID/NFC.

THE $3 GESTURE RECOGNIZER

Extending Wobbrock's [8] work, we present a gesture recognizer that can recognize gestures from 3D acceleration data as input. To test our algorithm we used acceleration samples obtained from a Nintendo Wii Controller (WiiMote). The WiiMote features an ADXL 330 Accelerometer [1] and the acceleration data can be sent, as in our case, via a Bluetooth connection to a PC. Our algorithm is by no means limited to the WiiMote. It can be used in any acceleration-enabled device, for instance modern smart-phones.

Gesture Trace

In contrast to [7], we do not modify or pre-process the raw acceleration data in any way (filtering, smoothing, etc.). To determine the current change in acceleration, we subtract the current acceleration value reported by the WiiMote from the previous one. We thus obtain an *acceleration delta*. By summation of the acceleration deltas, we obtain a *gesture trace* T which can be plotted in 3D space (Figure 1 gestures (e),(j)), or projected into a 2D plane (gestures (a)-(d), (f)-(i)) to obtain a graphical representation of the gesture [3].

Gesture Class Library

The *gesture class library* L contains a predefined number of gesture traces for each *gesture class* G. We also refer to these traces as *training gestures*.

Gesture Recognition Problem

The basic task of our algorithm is to find the best matching gesture class G from the gesture class library L, for a given input gesture I. (Example representatives of gesture classes is given in Figure 1.) To find a matching gesture class, we compare the trace t_i of I to the traces of all training gestures $t_{G_k} \in L$ and generate a score table that lists the comparison score of t_i and each t_{G_k}. A heuristic then is applied to the score table to determine if a gesture has been recognized.

Resampling

For optimal classification, the original gesture trace T needs to be resampled to have a number N of points equal to that of the template gestures. This is because the gesture input duration and movement speeds can vary between users, even for the same intended gesture. Resampling ensures that the points are re-distributed to be at equal distances from each other.

In our case $N = 150$, which is slightly above the average amount of acceleration deltas received while users enter a gesture with the WiiMote. Setting N to a lower value decreases the gesture recognition precision, while choosing a higher N just increases the computation time for gesture recognition, without a significant gain in accuracy.

Resampling is performed using piecewise linear interpolation, in which a resampled gesture trace T_N consisting of N equidistant points t_N is created. The locations of the T_N are built up by successive addition of the points t_k of the original gesture trace T to generate N equidistant segments connecting the new points t_N of T_N.

Rotation to "Indicative Angle" and Rescaling

To correct for rotational errors during gesture entry, the resampled gesture trace T_N is rotated once along the gesture's *indicative angle*. Like Wobbrock, we define the indicative angle as the angle between the gesture's first point p_0 and its centroid $c = (\bar{x}, \bar{y}, \bar{z})$. The angle is determined by taking the arcus cosine of the normalized scalar product of p_0 and c:

$$\theta = acos(\frac{p_0 \bullet c}{\|p_0\| \, \|c\|})$$

The rotation along the indicative angle is then performed using the unit vector of the vector orthogonal to p_0 and c. The orthogonal vector is obtained using the cross product of P_0 and c:

$$v_{axis} = \frac{p_0 \times c}{\|p_0 \times c\|}$$

The trace T_N is the rotated using v_{axis} and ϑ to obtain the rotated trace T_{N_ϑ}.

After rotation, T_{N_θ} is scaled to fit in a normalized cube of 100^3 units, to compensate for scaling differences between gestures. The algorithm has now finished preprocessing the original user input and has obtained a gesture T_M, which is ready for matching with candidate gestures from the gesture class library.

Search for Minimum Distance at Best Angle

Like Wobbrock, we use the average MSE (Mean Square Error) to calculate the path distance d between T_M and candidate gesture from the gesture class library. We convert the path distance to a $[0, 1]$ scale using a version of Wobbrock's scoring equation adapted to three dimensions, where d signifies the path distance and l the side length of the cube that T_M was scaled to in the rescaling step.

$$Score = 1 - \frac{d}{0.5\sqrt{3l^2}}$$

Following Wobbrock's discussion of rotation invariance of path distances, we have adapted a Golden Section Search (GSS) using the Golden Ratio $\varphi = 0.5(-1+\sqrt{5})$ to approximate the local minimum path distance within an angular range of $[-180° \ldots 180°]$, for rotation around the three axis of the coordinate system, signified by the angles α, β and γ. We define a minimum cutoff angle for GSS of 2°, in order to guarantee that the approximate minimum is found after exactly 11 iterations of GSS. We compared this approach to a brute-force implementation of the angle search and found that the result of GSS lies within 5° of the optimal rotation angle in the majority of cases.

The GSS-based minimum distance approximation is repeated for each trace of every gesture class in the library. The result is a table sorted by matching scores with the corresponding gesture class ID.

Scoring Heuristic

Wobbrock's original algorithm did not feature a heuristic to reduce the occurance of false positives, which is a common problem for simple gesture recognition algorithms operating on large gesture vocabularies [8]. The matches obtained from gestures entered as 3D acceleration data are not as precise as strokes entered on a touch screen. To compensate for the weaker matches, we have developed our own scoring heuristic, which processes the score table described in the previous section. Using this heuristic, we achieved a considerable reduction of false recognitions compared to Wobbrock's original strategy of selecting the gesture candidate with the highest matching score to determine the recognized gesture.

After sorting the score table by maximum score, our heuristic determines the recognized gesture with the following rules:

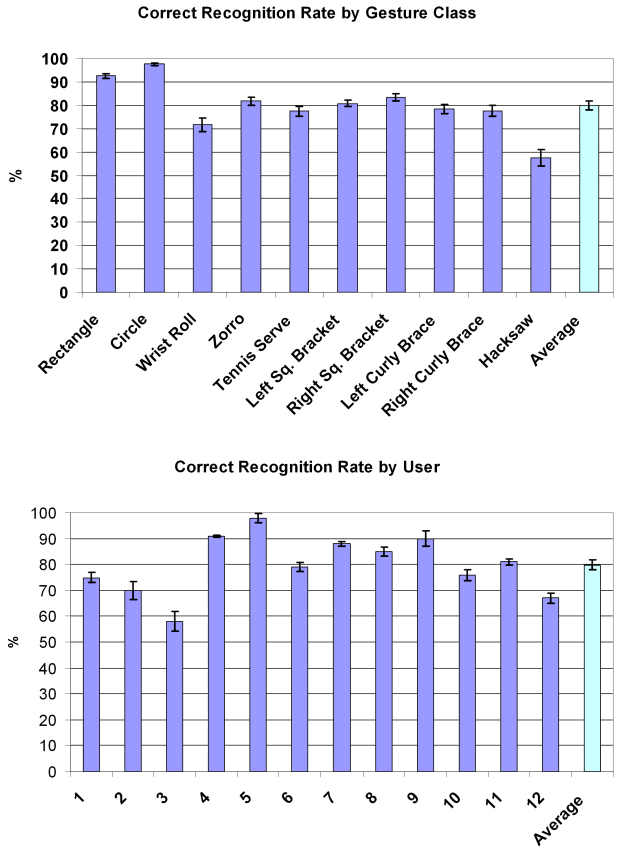

Figure 2. Average correct recognition rates with standard error, sorted by gesture class (top) and by user (bottom).

- ε is defined as the threshold score.
- Iff the highest-scoring candidate in the score table has a score $> 1.1\varepsilon$, return this candidate's gesture ID.
- Iff, within the top three candidates in the score table, two candidates exist of the same gesture class and have a score $> 0.95\varepsilon$, respectively, return the gesture ID of these two candidates.
- Else, return "Gesture not recognized!".

EVALUATION OF THE $3 GESTURE RECOGNIZER

To get an initial estimate of the gesture recognition performance of our method, we evaluated the 3$ gesture recognition algorithm with twelve participants, who were compensated for their effort.

Our reference gesture vocabulary contained all of the gestures utilized by [7] as well as a subset of Wobbrock's unistroke gestures [8], as displayed in Figure 1, totaling 10 unique gesture classes. We chose this particular set of gestures to make our study comparable to the previous work. Each user was asked to enter each gesture class in

the reference set 15 times using a WiiMote. The gesture data was recorded and stored on a PC.

The actual gesture recognition was performed offline using the stored gestures entered by the users. From each gesture class, the first five entered gestures of a particular gesture class were chosen as the training set for that class. The remaining gestures were input into our gesture recognition algorithm. Knowing the gesture class of the tested gesture beforehand, we recorded the number of times the gestures were correctly recognized, incorrectly recognized or not recognized at all.

Our evaluation resulted in average (correct) recognition rate of 80%. Between test subjects, the recognition rate varied between 58% and 98%, with a standard deviation of 11.4. As can be seen in Figure 2, the recognition rate was fairly constant across all users and gestures, with gesture class (b) having the highest average recognition rate and gesture class (j) being the most error-prone gesture. We speculate that the low recognition rate of gesture class (j) is due to the ambiguity of that gesture, as users varied the "sawing motion", which they were expected to perform, considerably. Notably, users commented that gesture classes (h) and (i) were the most uncomfortable gestures to perform. Furthermore, our results indicate that our scoring heuristic functioned acceptably, as only about 8% of all detected gestures were false positives.

Our gesture recognition algorithm yielded a lower correct recognition rate than those obtained with the system featured in [7]. In spite of this, we deem our correct recognition rate to be fully acceptable given that we used substantially simpler methods, and, which is more important, twice as many gesture classes with a significantly smaller gesture training set per class to achieve this recognition rate.

It is likely that the nearly 20% lower recognition rate of our method compared to [8] is influenced by the following factors. Gestures in 3D space are more difficult for a human to re-produce perfectly than in 2D, even for simple 2D shapes. More important, the equipment which we used to capture the gesture information was far from perfect, and may have contributed to the reduced recognition rate.

Limits of the $3 Gesture Recognizer

As it is a simple algorithm, the $3 Gesture Recognizer has several limitations. For one, in contrast to more refined methods such as those based on HMMs, it cannot be used to detect gestures in a continuous motion stream. Only gestures which are explicitly started and stopped by the user can be recognized. A further limitation is the size of the gesture vocabulary. Not only does the number of false positive recognitions rise together with the size of the gesture vocabulary, but also the computational overhead ($O(N \cdot M)$, where N is the amount of motion samples, and M is the number of training gestures in the gesture class library). This is due to the intensive use of trigonometric functions, increases as well, which limits the maximum practicable size of the gesture vocabulary to about 10-15 gestures. These limitations, however do not represent an impediment for the use of our recognizer in its target domain — rapid prototyping of gesture-based interfaces.

DISCUSSION AND FUTURE WORK

We presented a simple, easy-to-implement gesture recognizer for input devices equipped with 3D acceleration sensors. The idea behind our gesture recognition algorithm is to provide a quick and cheap way to implement gesture recognition for true 3D gestures (such as the reference gesture (e)). Our method does not require any advanced software frameworks or toolkits. The gesture set is not fixed but can be specified by as needed, even at runtime. An example application area for our gesture recognizer is user interface prototyping.

In an initial evaluation of our algorithm, we obtained gesture recognition rates which are comparable to those of more advanced approaches. The advantage of our system is that it is specifically targeted for use in prototype environments, in which gesture-based interfaces (or multimodal interfaces using gestures as one of multiple components), are needed that provide quick results with little coding and minimal training data.

REFERENCES

1. Analog Devices ADXL330, http://tr.im/GTHc.
2. Nintendo inc. http://wii.nintendo.com.
3. S. Kallio, J. Kela, J. Mäntyjärvi, and J. Plomp. Visualization of hand gestures for pervasive computing environments. In *Proc. AVI '06*, pages 480–483, New York, NY, USA, 2006. ACM.
4. J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and S.D. Marca. Accelerometer-based gesture control for a design environment. *Personal Ubiquitous Computing*, 10(5):285–299, 2006.
5. L. Kratz, M. Smith, and F.J. Lee. Wiizards: 3d gesture recognition for game play input. In *Proc. Future Play '07*, pages 209–212, New York, NY, USA, 2007. ACM.
6. LR Rabiner. A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
7. T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In *Proc. TEI '08*, pages 11–14, New York, NY, USA, 2008. ACM.
8. J.O. Wobbrock, A.D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proc. UIST '07*, pages 159–168, New York, NY, USA, 2007. ACM.