Gesture Recognition Using The XWand

Daniel Wilson and Andy Wilson

CMU-RI-TR-04-31

Gesture Recognition Using The XWand

Daniel Wilson[1] and Andy Wilson[2]

[1] Assistive Intelligent Environments Group
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh. PA 15213
dan.wilson@cs.emu.edu
[2] Microsoft Research
One Microsoft Way
Redmond. WA 98052
awilson@microsoft.com

Abstract. The XWand is a wireless UI device that enables styles of natural interaction with intelligent environments. The XWand system exploits human intuition, allowing control of everyday objects through pointing and gesturing. We describe the hardware device and then examine several approaches to gesture recognition. We discuss results from experiments using a linear time warping method, a dynamic time warping (DTW) method, and a hidden Markov model-based method (HMM).

1 Introduction

A multitude of intelligent devices increasingly pervade modern environments. Often we have access to an intimidating variety of these specialized interfaces, each controlling one device. For example, today's living room coffee table is typically cluttered with multiple user interfaces in the form of IR remote controls (Figure 1a). In the near future we can look forward to these devices becoming more interconnected, more numerous and more specialized as part of an increasingly complex and powerful integrated intelligent environment. The challenge is to find ways of naturally and efficiently interacting with this environment.

We introduce the XWand, a hardware device (Figure 1b) and associated signal processing algorithms for an integrated interface capable of controlling multiple connected devices in a natural manner. The XWand hardware represents a wireless sensor package that transmits information useful for orientation and gesture recognition. Most recent work in gesture recognition uses machine vision to accomplish the same purpose. However, the XWand does not suffer from sensitivity to lighting conditions, camera movement, or larger privacy issues inherent in vision-based approaches. For example, the XWand may be used in a home where a camera network may be infeasible or unwanted. The XWand would also be more suitable outdoors as it is immune to lighting variation and atmospheric conditions (dust & debris).

Using the XWand a user can point at any device of interest and control it using simple gestures. The intelligent environment system interprets the user's manipulation of

(a) State of the art. (b) The XWand.

Fig. 1. Many specialized interfaces vs. a single multipurpose tool.

the wand to determine an appropriate action in context. The ultimate goal is to provide an interface so simple that it requires no particular instruction or special knowledge to use, and instead relies on the intelligence of the environment to determine an appropriate action.

2 Hardware Device

2.1 Design Specification

We have constructed an early hardware prototype of the XWand (Figure 2). A variety of sensors are mounted onto an 8"xl" circuit board that fits inside a clear lucite tube, making the device easy to grasp and intuitive to point. The XWand has the following features:

- Analog Devices ADXL202 2-axis MEMS accelerometer. This measures both dynamic and static acceleration. When held motionless, the sensor detects the angle of the wand relative to the constant downward acceleration of gravity. The sensor is mounted so that it reports pitch and roll.

- Honeywell HMC1023 3-axis magnetoresistive permalloy magnetometer. This sensor reports the direction of the Earth's magnetic field in 3 dimensions, and contributes to computation of the yaw angle of the device.

- Murata ENC-03 1-axis piezoelectric gyroscope. This is an angular rate sensor, and is centered on the XWand to sense motion about the vertical axis (yaw).

- Pushbutton. The button is used to indicate when a gesture is being made. On this prototype the button is placed on top of the XWand, just under the ball of the thumb.

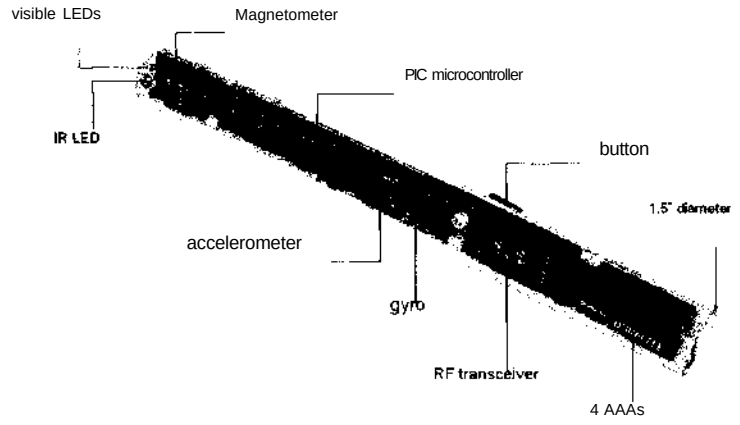

Fig. 2. XWand hardware prototype.

It is pressed by squeezing the hand or by using the thumb.

- BIM 433 MHz FM transceiver (38.4kbps). The transceiver sends information to a similarly equipped base station, which then communicates with a host PC via RS232. The base and the wand use a command/response protocol; i.e., the wand only sends a data packet when the base requests it. Continual polling by the host yields a 50Hz frame rate. The transceiver design permits simultaneous use of multiple wands that share the same bandwidth, as well as real-time two-way interaction with the host PC.

- PIC 16F873 flash-programmable microcontroller (20MHz). The microcontroller collects sensor values via digital and analog-to-digital inputs and sends data to the transceiver. It also formats outgoing data communication packets, decodes received packages, controls timing, and performs power management.

- Infra-red (IR) LED. Invisible to the naked eye, this LED can be seen by cameras equipped with an IR pass filter. This may be used to support position tracking of the XWand.

- Green and red visible LEDs. These deliver feedback to the user and may be lit in response to commands received from the host PC. A green LED indicates whether the wand is 'asleep' or 'awake'. A red LED indicates when the wand is communi-

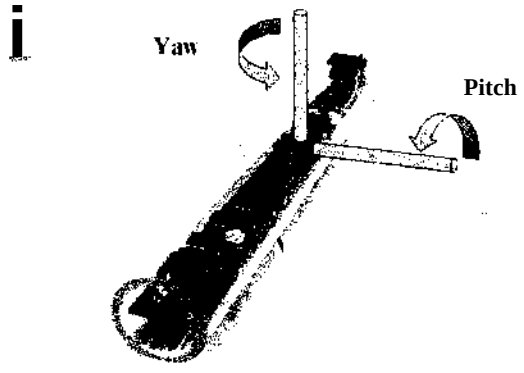

Fig. 3. Yaw and pitch directions.

eating with the base station.

- 4 AAA batteries. Quiescent power when awake is approximately 52mA, and is less than 1 mA while asleep (thousands of hours of standby operation).

There are caveats related to this combination of sensors. For example, the accelerometer only delivers true pitch and roll information when the device is motionless. For pointing tasks this problem is avoided by relying on the orientation information only when the device is motionless, as indicated by the magnetometer output. The inaccuracies are constant, so for gesture recognition purposes they simply become part of a learned pattern.

3 Gesture Interpretation

3.1 **Features**

A user makes a gesture with the wand by holding the wand motionless, squeezing the button, waving the wand in the desired pattern, and then releasing the button. The gesture is stored as a sequence of sensor values over time. During the training phase we form gesture models by combining many different examples of the same gesture collected from the XWand. During testing we perform classification (recognition) of a new gesture sequence by comparing it to each existing gesture model and then choosing the model with the highest score above some cutoff threshold.

The XWand is designed for both pointing and gesturing tasks. For gesture recognition purposes we are interested in how gestures are made with the wand (i.e., velocities), not orientation. We use velocities so that the same gesture sequence will remain constant regardless of the wand orientation. Specifically, we use changes in pitch and yaw.

Roll velocity is not used because rolling motions do not contribute distinctly to most gestures that we use. We use velocity of pitch and yaw as features for gesture recognition. The gyroscope provides angular rate of change along the vertical axis of the device (yaw), corresponding to turning the wand left or right. Up and down wrist motions are represented by changes in pitch provided by the accelerometer. See figure 4 for clarification.

3.2 **Gesture Vocabulary**

Our goal is to provide a multipurpose set of gestures useful for interacting with a multitude of interconnected devices in an intelligent environment. We detect the following gestures: *up, down, left, right, clock-wise circle, counter-clockwise circle,* and *take* gestures. These actions correspond to several commonly used actions on current remote controls. They map intuitively to common concepts such as turning up/down volume, turning to the next/previous channel, and rewind/fast forwarding a movie . The *take* gesture is useful in an intelligent environment where a user may wish to carry services from device to device. For example, a user might take a computer login and carry it to another computer, take a movie from one television to another, or music from one set of speakers to another. For each of these seven actions we train an associated gesture recognition model.

4 Gesture Recognition

Every person has a unique gesturing style. Although we have established a finite set of gestures, they can each be performed at different speeds and with different amplitudes. Consider the *clock-wise circle* gesture. One user may quickly make a small circle, and another might slowly make a large circle. Still another user might quickly make half a circle, then slowly make the other half. Sequences are comprised of derivative values, so the faster a gesture is performed the higher the amplitude. Automatic recognition of new gesture sequences must account for these variations in time and scaling. We consider three approaches: linear time warping (LTW), dynamic time warping (DTW) , and an HMM based approach.

4.1 Linear Time Warping Approach

This algorithm matches a given sequence of sensor values, $s = \{si, ...ST, \}$, to a stored prototype, $p = \{pi, ...,pr_P\}$. This stored prototype is collected earlier during a special training procedure. To account for the fact that the gesture made by the user during runtime may differ from the prototype in terms of speed and amplitude, the sequence matching algorithm tries matching various versions of the prototype that are stretched in time and scaled in amplitude. Sensor readings from the prototype and input sequence are then compared using squared Euclidean distance. This match score is then computed over the whole sequence for a given scale and time warp. The final match score is the maximum score out of all the scales and time warps. The sequence is classified according to the prototype with the highest match score.

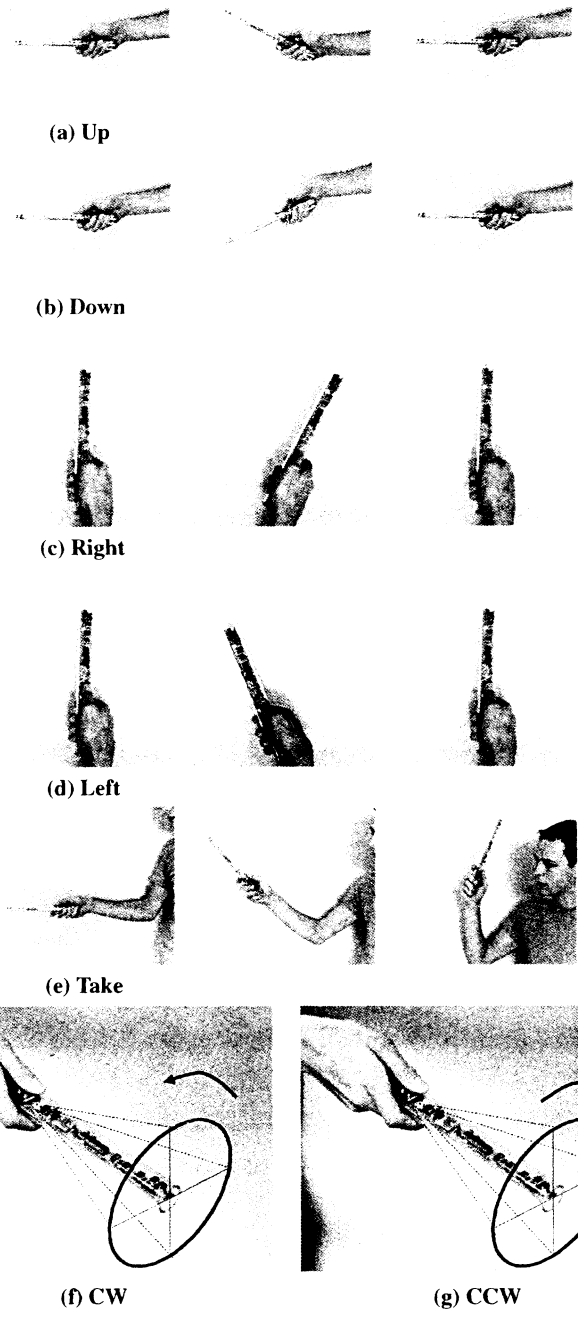

Fig. 4. Demonstration of *UP, DOWN, LEFT, RIGHT, TAKE, CLOCK-WISE*, and *COUNTER-CLOCKWISE* gesures.

During the training procedure we combine training examples to create a model for each gesture. Initially one example of each gesture is chosen as the prototype. A subsequent training example s is first linearly scaled to fit the prototype, and then we compute the mean μ between the two sequences at each time step t and store them as a vector of length T_p : $M = \{\mu_1, ..., \mu_2\}$.

The temporal match up provided by LTW is often a rough approximation and the corresponding feature vectors may only be approximately related. Scoring based strictly upon location in the time sequence places too much trust in LTW. A solution is to distribute the scoring function between neighboring time slots in a Gaussian fashion. Instead of comparing only feature vectors that share a time slot, (i.e., s_t and p_t), we calculate the score between each feature vector for $t = 1, ..., T_p$, weighted by a Gaussian centered on the current time slot t. The size of the variance on our Gaussian is inversely proportional to our trust in the warp provided by LTW. We chose our value by cross-validation.

This template matching approach allows the user to train the prototype template rather than setting it up by hand. Also, the model gesture may be arbitrarily complex. For example, this technique can model a left to right motion of two complete periods. A drawback of this approach is that runtime variations of the gesture may involve more than constant scaling of amplitude and linear time warps.

4.2 Dynamic Time Warping Approach

This approach is similar to linear time warping except that DTW can account for non-linear time-alignment differences between test and reference patterns. These may occur when a user performs a gesture with variable speed. As in LTW our goal is to match a given sequence of sensor values to a stored prototype. The stored prototype is collected earlier during a special training procedure. Again, training sequences are combined into prototype sequences for each type of gesture. Training sequences are aligned temporally using DTW before combination. To account for amplitude differences, the sequence matching algorithm tries matching several versions of the prototype with differently scaled amplitudes. Sensor readings from the prototype and input sequence are then compared using Mahalanobis distance. The sequence is classified as the gesture with the highest score.

DTW provides an efficient way to temporally align two sequences of different lengths. Suppose we are given an input sequence $s = \{s_1, ..., s_{T_S}\}$ and prototype sequence $p = \{p_1, ..., p_{T_P}\}$. The problem can be thought of in terms of a grid with horizontal axis associated with s, and vertical axis associated with p. The sequence s is a k-dimensional feature vector with values collected from k sensors (in our case $k = 2$). Each element of the grid contains a Euclidean distance measure D_{ij} representing the distance between the s_i and p_j. The best time warp will be the optimal path (minimized accumulated distance) from point $(0, 0)$ to (T_S, T_P) on a finite grid. The optimal solution can be found efficiently by dynamic programming in $O(T_S\ T_P)$ time. For a more detailed description of DTWs see [?]

Training. We use DTW to align a training sequence *s* to the prototype length *Tp.* We compute the mean */JL* between the two sequences at each time step *t* and store them as a vector of length T : *M* = {/ii,..., *fir}-*

Scoring. Given a test pattern *s* we compare it to a prototype sequence *p* representing each gesture. We use DTW to align the input sequence *s* to the prototype sequence p. For each time step *t* we gather a cumulative score. The score is computed by summing the Mahalanobis distance between feature vectors from each time step. Given a test sequence *s* = {si,..., sj_P} and reference sequence*p* = {pi, ...,pr_P} the Mahalanobis distance between *Si and pj* is :

$$D_{s_i,p_j} = (s_i - p_j)' \Sigma^{-1} (s_i - p_j) . \quad (1)$$

Where Σ^{-1} is the inverted covariance matrix. This distance metric removes several limitations of the Euclidean metric. Specifically, the Euclidean distance does not provide any statistical measurement of how well the unknown sequence matches the training set. In addition, the Euclidean distance only measures a relative distance from the mean point in the sequence. It does not take into account the distribution of the values in the sequence. In our case the added computation is manageable. For each prototype sequence our input sequence is aligned and then scored. The input sequence is classified as the model with the highest score.

4.3 Hidden Markov Model Approach

We also tried using an HMM recognizer to train and later classify gestures. Although originally developed for problems in speech recognition [?], [?], HMMs have become a popular approach to gesture recognition [?] as well. HMMs provide a probabilistic framework that can account for dynamically time-varying gesture sequences. See [?] for a detailed description of how HMMs work. Our implementation is based on the public domain HTK toolkit version 3.0 [?]. We modified the code to handle a continuous data stream. This toolkit has been used in several other gesture recognition applications [?].

The approach works by using examples collected during a training phase to create a Markov model describing the temporal structure of the gesture. The Markov model is composed of several states that represent different temporal sections of the gesture sequence. For example, the *right* gesture can be broken into two states : (A) moving to the right and (B) returning to the original position. Our Markov models each have six states. This number was chosen using cross-validation. Each Markov model is given several training sequences and associates sections of these sequences with the different states (e.g., A to B).

Before the model is trained, however, the initial topology must be consistent with the gesture vocabulary. None of our gestures exhibit periodicity, therefore each state transitions either to itself or to the next state. During runtime a new sequence is compared to each Markov model and scores are collected. The model with the highest probability is chosen.

5 Experiments and Results

5.1 Process

Six male subjects were selected from around the research lab to participate in training. None of the subjects had ever used the XWand. Each subject was shown how to perform each gesture, and was then allowed to practice each gesture once. Afterwards, ten examples of each of the seven gestures were collected. For every subject the entire process was complete in well under *fi\e* minutes. This amounted to seventy examples from each user, and 420 examples total. The users performed gestures at a variety of speeds, facing different directions, and with distinct overall styles. Two users were left-handed, and the rest right.

Each algorithm used the same test and training sets. We tested on one instance of seven gestures from six users, for a total of 42 instances. For each user we discarded the first two instances of each gesture, due to learning effects. The remaining seven instances of the seven gestures from six users were used for training, leaving a total of 294 examples. See Table 1 for a summary of the results.

Table 1. Comparison of recognizer performance

Algorithm	Accuracy	Correct / Total
Linear Time Warping	40.42%	17 / 42
Dynamic Time Warping	71.64%	30 / 42
Hidden Markov Model	90.43%	38/42

6 Discussion

The performance of LTW is probably due to the inability of LTW to compensate for dynamic time warps. DTW performance was better, but not nearly as good as HMMs. The order in which LTW and DTW receive training examples can directly affect performance. If the original prototype sequence for a gesture has noise or is cut short, then that model will be flawed from the beginning.

The use of the button is a major factor affecting performance. The starting and endpoints of gesture sequences were roughly known, but there is room for improvement. A gesture is collected when the user squeezes and releases a button, which leaves a variation in button press time. For example, one user may hold down the button longer than another before starting a gesture or hold it down longer after the gesture is completed. Sequences may be skewed, which would have a serious effect on LTW, but not on HMMs. This could be solved in future work, as it has been shown that these linear trends can be efficiently removed [?], [?]. Often users completed part of a gesture before pressing the button, or let go of the button before a gesture was completed. Again, this

poses a problem for LTW, but less for HMM. For an HMM both problems are solved by using extra states, which can then represent button pressing behavior.

Performance in the HMMs was heavily affected by the *take* gesture. This gesture accounted for all four misclassifications. Initially the *take* gesture was formulated as a pull towards the left shoulder. Left-handed users naturally performed the gesture towards the right shoulder. The difference in feature vectors was too great to be encompassed in a single model. Ideally, either the gesture should be the same for both right and left-handers, or two models should be trained. Barring this miscalculation, the HMM approach performed perfectly.

7 Conclusion

We have introduced the XWand, a UI capable of pointing and gesturing in an intelligent environment. The XWand relies on the natural tendency to point and gesture at objects that we wish to control. We describe three approaches to gesture recognition with the XWand. We report positive results indicating that the XWand can be a useful interface for controlling the many interconnected devices populating environments of the future.

References

1. Darrell, J. and Pentland. A. Space-time gestures. CVPR, p.335-340, 1993.
2. Rabiner, L. and Juang, B. An introduction to hidden Markov models. IEEE ASSP Magazine, p.4-16,Jan.1986.
3. Young, S. HTK: Hidden Markov Model Toolkit VI.5. Cambridge Univ. Eng. Dept. Speech Group and Entropic Research Lab. Inc.. Washington DC, Dec. 1993.
4. Iba, S. Vende Weghe, J.M. Paredis, C. and Khosla, P. "An Architecture for Gesture Based Control of Mobile Robots," Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'99), October, 1999.
5. Myers, R. and Whitson, J. "Hidden Markov Model for automatic speech recognition," ftp://svrftp.eng.cam.ac.uk/pub/comp.speech/recognition/. University of California at Irvine, 1994.
6. Starner, T. Visual Recognition of American Sign Language Using Hidden Markov Models. Master's thesis, MIT Media Laboratory, Feb. 1995.
7. Huang, X. Ariki. Y. and Jack. M. Hidden Markov Models for Speech Recognition. Edinburgh Univ. Press, Edinburgh, 1990.
8. Keogh, E. & Pazzani, M. Derivative Dynamic Time Warping. In First SIAM International Conference on Data Mining (SDM'2001), Chicago, USA. 2001.
9. Keogh, E., & Pazzani, M. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. Proceedings of the Fourth International Conference of Knowledge Discovery and Data Mining, pp 239-241, AAAI Press. 1998.
10. Agrawal, R., Lin, K.I., Sawhney, H.S., & Shim, K. Fast similarity search in the presence of noise, scaling, and translation in times-series databases. In VLDB, September. 1995.