Robust Control of Applications by Hand-Gestures

Aakash Anuj, Tanwi Mallick, Partha Pratim Das, and Arun Kumar Majumdar
Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur 721302, India
Email: {aakash.anuj, tanwi.mallick, ppd, akmj}@cse.iitkgp.ernet.in

***Abstract*—We use the RGB-D technology of Kinect to control an application with hand-gestures. We use PowerPoint for test. The system can start/end PPT, navigate between slides, capture or release the control of the cursor, and control it through natural gestures. Such a system is useful and hygienic in the kitchen, lavatories, hospital ICUs for touch-less surgery, and the like.**

The challenge is to extract meaningful gestures from continuous hand motions. We propose a system that recognizes isolated gestures from continuous hand motions for multiple gestures in real-time. Experimental results show that the system has 96.48% precision (at 96.00% recall) and performs better than the Microsoft Gesture Recognition library for swipe gestures.

I. Introduction

In the context of *Human-Computer Interaction* (HCI), gesture is used to control a machine much in the same way we communicate with other human beings. Traditional gesture-based *Natural User Interfaces* (NUI) using RGB images or videos are now being taken over by low-cost RGB-D technology like Microsoft Kinect. Hence, we develop several gesture detection algorithms for Kinect skeletal models to control a Microsoft PowerPoint presentation as a target. The algorithms can also be used for hands-free control of touch-less surgery, immersive image navigation, emotive communication, auto-piloting, and the like. Any real-time gesture recognition system needs to address the following issues that we handle here:

Gesture Spotting is the detection of meaningful gestural patterns (computing the start and the end points) from continuous hand motions for which Lee & Kim [9] built a complex system. We use various fixed-length gestures recognized by machine learning algorithms as an alternative to gesture spotting.

Gestures need to *accommodate* *inter-variability* and *intra-variability*, as the same gesture can vary in shape as well as duration. Inter-variability refers to the differences when multiple people perform the same gesture, while intra-variability refers to the differences when a single individual performs the same gesture. We take care of variabilities through vector quantization and learning models.

Rejecting a non-gesture is usually harder than recognizing a gesture because there could potentially be infinite number of unintended gestures to eliminate. Wilcox & Bush [18] use *Garbage* or *Filler Model* to handle such gestures. Training such a model is a difficult task. For this, we characterize every gesture and use the *Threshold Model* [9] to calculate the likelihood threshold of an input pattern and build a confirmation mechanism for the provisionally matched gesture patterns.

In the paper, Section II discusses prior work. We describe the design of the NUI in Section III. Section IV discusses the gesture recognition system with choice of features and classifiers. In Section V, we explain the experiments, present the results and compare the performance of the system with the Microsoft Gesture Library [10]. We conclude in Section VI.

978-1-4673-8564-0/15/$31.00 ©2015 IEEE

II. Related Work

Gesture recognition has been applied in various application areas such as recognizing sign language [11], [16], HCI [11], [8], robot control [11], smart surveillance [11], lie detection [11], and visual environments manipulation [11]. In the last decade, several gesture interfaces [13] have been developed. For any system the first step is to collect the data for the specific task. Broadly two different technologies have been used so far to capture data:

Vision Based Approaches use one or more video cameras to record movements. Features such as shape, texture, motion, and color as extracted from the video are used to analyse a gesture [5]. These approaches are simple but many challenges like complex background, lighting variation, and skin color objects with the hand object exist here. Hand detection is a major issue here. Hence, marked gloves or coloured markers [7] are used for tracking the hand, the palm and fingers.

Sensor-Gloves Based Approaches use wearable sensors in gloves [3] for capturing hand positions and motion. They can easily provide exact coordinates of palm, fingers' location, orientation, and hand configurations. But it hinders naturalness.

RGB-D sensors like Kinect improves the above by using IR and tracking the human skeleton. Using joint-points of skeletons, the gestures are recognized by mathematical models like Hidden Markov Model (HMM) [11], [14], [19], or Finite State Machine (FSM) [11], [6], or soft computing methods like fuzzy clustering [20] and Artificial Neural Network (ANN) [17]. We use multiple HMM's to classify gestures.

III. Design of NUI

Following the *Human Interface Guidelines* [12] for Kinect, we have designed an NUI as a set of gestures to control the basic operations in a PowerPoint presentation. We assume the user to be right-handed and ascribe most of the controls to the right hand (except when both hands are used). The user faces the screen and Kinect when she controls the presentation:

Presentation Control gestures are *learned* and *static* [12] and are based on the positions of certain joint-points.

- *Right Hand Raised* Gesture (Table I(A)): The user raises her *right hand* above the head to *start*.
- *Both Hands Raised* Gesture (Table I(B)): The user raises *both her hands* above the head to *end*.

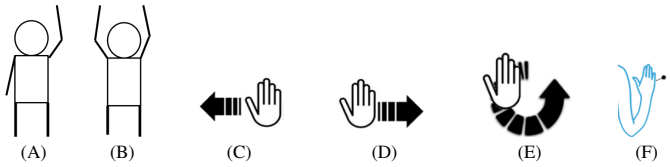


TABLE I. PRESENTATION, NAVIGATION, AND CURSOR CONTROL GESTURES. (A) *Right Hand Raised* TO START (B) *Both Hands Raised* TO END (C) *Left Swipe* TO GO TO NEXT SLIDE (D) *Right Swipe* TO GO TO PREVIOUS SLIDE (E) *Circle* GESTURE TO CAPTURE THE CONTROL OF THE CURSOR (F) *Push* GESTURE TO RELEASE THE CONTROL

Navigation Control gesture are *innate* and *dynamic* [12]. They are based on the trajectory of the palm movement.

- *Left Swipe* Gesture (Table I(C)): The user moves *right hand* from right to left to *go to next slide*.
- *Right Swipe* Gesture (Table I(D)): The user moves *right hand* from left to right to *go to previous slide*.

Cursor Control is based on the trajectory of the palm.

- *Circle* Gesture (Table I(E)): *To capture the control of the cursor*, the user raises her *right hand* and draws a circle in space by moving the open palm facing the sensor in counter-clockwise (CCW) manner.
- *Cursor Tracking* Gesture: *Once captured, the cursor keeps on moving* on the slide as the user moves her *open right palm* facing the sensor. This can be used for pointing in slides. This gesture ends with *Push*.
- *Push* Gesture (Table I(F)): *To release the cursor*, the user moves her *open right palm* towards the sensor.

While *Circle* and *Push* gestures are *learned* and *dynamic*, *Cursor Tracking* gesture is *innate* and *continuous* [12].

Using the above gestures a user can start a presentation, navigate between slides, use the cursor as a pointer while she is at a slide, and finally end the presentation.

IV. GESTURE RECOGNITION SYSTEM

We use various joint-points from the skeleton stream of Kinect to recognize and track gestures. We use elbow (EL_R & EL_L) and shoulder (SH_R & SH_L) joints for the *Hand Raised* gestures, the joints of the right hand (WR_R (wrist), EL_R, SH_R) for *Swipe*, *Circle*, *Track* and *Push* gestures, and the hip joints (HIP_R & HIP_L) to determine the orientation of the user relative to Kinect. Based on the characterization of the projected tracked points we build a separate classifier each for each gesture. The architecture is shown in Figure 1.

Feature Extraction: Since the joint-points move in a 3D space, we take their projections on a *virtual wall* (XY or YZ plane) and reduce the problem to 2D tracking. The *virtual wall* is computed for a coordinate system attached to the user and aligned with her orientation so that even if the user is inclined with respect to Kinect (imaging plane), the projection would not distort the path being tracked. *Without this correction a circle traced in 3D will become an oblique ellipse after the projection, if the user is inclined relative to Kinect.*

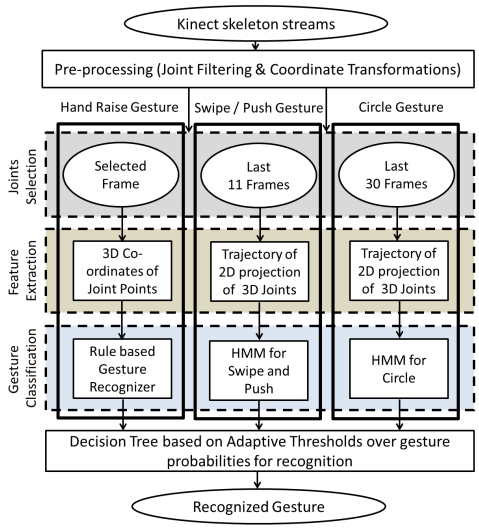


Fig. 1. Architecture of Gesture Recognition System. Every vertical box runs in a separate thread and checks for gestures of a class. Every dotted box depicts a common stage of processing across threads

For the *static* gesture (*Hand Raised*), we use the joint-points directly as features while for the *dynamic* (or tracking) gestures (*Swipe* & *Push*, *Track*, and *Circle*) we use the orientation between consecutive joint-points as features. A simple rule-based classifier works well for static gestures but we need a learning-based model (HMM) for tracking gestures.

Hidden Markov Models (HMM) as Gesture Classifiers: To track the features we consider n consecutive frames to train and evaluate different HMMs. For stability we use vector quantization of orientation values as code-words. We configure [4] the HMM's in *Forward Banded Topology* and use *Forward and Backward* algorithm for *Evaluation*, *Viterbi algorithm* for *Decoding*, and *Baum-Welch* algorithm for *Training*.

Adaptive Thresholds on Gesture Probabilities: We create a discrete HMM for each dynamic gesture[1]. We feed the stream of code-words simultaneously to all HMM's. Each HMM, in turn, returns a likelihood for match. The likelihoods are converted to probabilities by normalization, followed by adaptive threshold [9] to classify the gestures to the correct class.

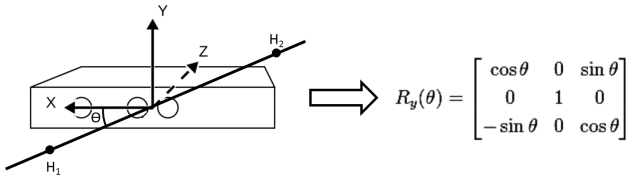
A. Data Acquisition and Preprocessing

We have used 5 subjects to build a test data-set for gestures. Every subject performs each gesture 16 times – correctly 8 times and incorrectly 8 times – to build a data-set of 400 total gesture actions for 5 gestures. We use this to test our system. The training gestures are recorded separately.

We first filter the joint-points by *Holt Double Exponential Smoothing* [2] to produce smooth tracking. It is controlled by five parameters, namely, *smoothing*, *correction*, *prediction*, *jitter radius* and *maximum deviation radius*. Next we translate

[1]Compared to a single HMM for all gestures, multiple HMM's better preserve the discriminating features of every gesture and have better accuracy. For every HMM we use 5 parameters – 2 of them structural viz. number of states and number of distinct observation symbols per state (set as input); and 3 of them canonical viz. state-transition probability distribution, observation symbol probability distribution, initial state distribution (these are estimated).

the origin from Kinect to the mid-point of line L (Figure 2) joining the hip joints (HIP_R & HIP_L) of the user and treat L the new X–axis. Finally, we compute the orientation of the user relative to Kinect, the angle between L and the X–axis of Kinect, and rotate the skeleton about the Y–axis to align. This ensure a stationary origin and well-behaved projections.



$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

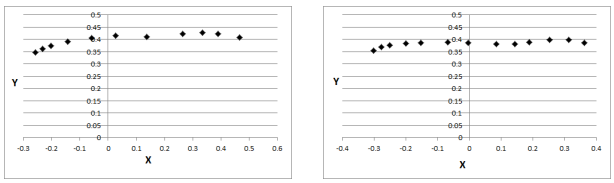
Fig. 2. Axial rotation for orientation independence. The line $L = H_1H_2$ joining H_1 (HIP_L) and H_2 (HIP_R) defines the axis of alignment.

B. Feature Extraction and Gesture Classification

Gestures for presentation control (Section III) fall into four classes – *Hand Raised*, *Swipe* & *Push*, *Circle*, and *Track*.

Hand Raised Gestures (Table I(A) & (B)) are *static*. The trajectory is immaterial as they are well-defined by a set of rules. Using the 2D joint-points on the *virtual wall*, the rules for *Right Hand Raised* are: (1) WR_R is above EL_R, (2) EL_R is above SH_R, (3) WR_R is to the left of EL_R, (4) WR_R is to the right of SH_L, and (5) distance(WR_R, EL_R) + distance(EL_R, SH_R) = distance(WR_R, SH_R). We use rule-based approach to classify these gesture. *Both Hands Raised* (Table I(B)) gesture is implemented similarly.

Swipe and Push Gestures (Table I (C) & (D), (F)) are all defined by near-straight-line trajectories of joint-points. We first represent the trajectory of the right hand (WR_R), then encode it as a feature vector (or codeword), and finally use an HMM to classify. Since the trajectory for all three gestures is a straight line, specific gestures are distinguished by distinct representation and encoding processes. To represent the trajectory, we use the 2D joint-points on the XY plane for *Swipe* gestures (Figure 3). *Push* gesture uses the YZ plane.



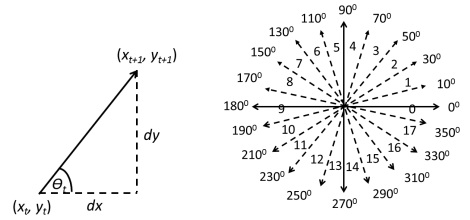
(a) *Left Swipe* on XY plane (b) *Right Swipe* on XY plane

Fig. 3. Projected trajectories of *Swipe* gestures on *virtual wall*

To encode the projected trajectory, we use the change of orientation $\theta_t = tan^{-1}[(y_{t+1} - y_t)/(x_{t+1} - x_t)], t = 1, 2, 3, \cdots, n-1$ between the consecutive points, where n is the number of frames that the gesture spans and θ_t is taken from the +ve direction of the X–axis. Thus for a *Left Swipe*, θ_t is nearly 180^o; while it is nearly 0^o for a *Right Swipe*.

Angle θ_t is quantized[2] by *Naïve Vector Quantization* (Figure 4). Hence for a *Left Swipe* with the sequence of

[2]*This quantization scheme from [4] is rotated CCW by* 10^o *to ensure stability around the* X*-axis. Otherwise,* Left Swipe *or* Right Swipe *would fluctuate between two different codewords on very small changes of the angle.*



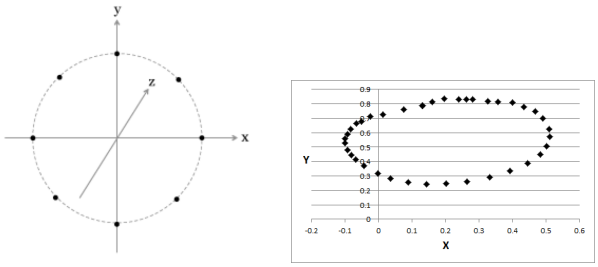
(a) (b)

Fig. 4. Naïve Vector Quantization. (a) Orientation θ_t between (x_t, y_t) and (x_{t+1}, y_{t+1}). (b) 18 angular bins (of 20^o each) are marked by the central angles $[0^o, 20^o, 40^o, \cdots, 340^o]$ and corresponding codes $[0, 1, 2, \cdots, 17]$. The range of a bin is taken as $\pm 10^o$ around the central angle.

angles as $[180^o, 170^o, 182^o, \cdots, 170^o, 180^o, 185^o]$, the feature vector (codewords) is $[9, 9, 9, \cdots, 9, 9, 9]$; while for a *Right Swipe* $[0^o, 356^o, 1^o, \cdots, 0^o, 357^o, 5^o]$ the feature vector would be $[0, 0, 0, \cdots, 0, 0, 0]$. So *Right* is distinguishable from *Left*.

The average length for a *Swipe* / *Push* is found to be n = 11 frames. Hence, the last 11 frames are used for its detection.

Circle Gesture (Table I(E)) is complex and is characterized by: (1) Closed trajectory of WR_R, (2) Arbitrary start-point, (3) Approximate end-point that is nearby the start-point, (4) Long duration (˜1 sec, 3 times *Swipe*) of about 30 frames at 30 fps, (5) Speed agnostic, (6) Highly dependent on the build and orientation of the user, (7) Flexibility in its circularity.



(a) *An ideal one with 8 states* (b) *A real one with several states*

Fig. 5. Projections of a *Circle* gesture

The projection of circle is often noisy (Figure 5). So we again use *Naïve Vector Quantization*[3] (Figure 4) to codify them with minimal noise. Using 30 consecutive frames, we

Step 1: Quantize the points using the scheme in Figure 4.

Step 2: Train the HMM (8 states) with multiple gestures.

Step 3: Generate more training sets from cyclic permutations (for arbitrary start and end points) of the gesture in Step 2.

Step 4: Continually extract the last 30 frames and, using the HMM, estimate its likelihood of being a circle. Declare a circle once the likelihood exceeds an adaptive threshold.

Cursor Tracking Gesture: The mouse cursor is grabbed (*lock*) with the *Circle* gesture and released (*unlock*) by the *Push* gesture. Once it has been captured, the cursor can be moved

[3]*Since a circle has 8 symmetric octants, we first attempt to quantize the 30 points of a potential circle into 8 clusters by K-means Clustering. That fails as the density of points (depending on speed) in every cluster is same and the seed is indeterminable (start / end of a circle is arbitrary).*

by moving the right hand (WR_R). This gesture is *continuous* in nature and is characterized by the lack of any specific movement pattern. The trajectory is converted to orientation and streamed as mouse control events to move the cursor.

V. Experiments and Results

We use the C# Accord.Net machine learning framework [1], [15] for HMM implementation and perform several experiments to tune the configuration of the HMM to arrive at the best suite of training data, and to determine the optimal number of states (s) and gesture lengths (n). For example, for *Swipe* $s = 5$ & $n = 11$ and for *Circle* $s = 8$ & $n = 30$.

After training, tests are carried out with 400 gesture samples from 5 gestures performed by 5 subjects. 200 of these are correctly performed and 200 are incorrectly (unintended or improper gestures) performed. Table II shows that for a mix of all gestures for tests, our system has 96.48% precision and 96.00% recall. In Table III, we compare this with Microsoft Gesture Library (MGL) [10] for *Left* and *Right Swipe* gestures using 400 swipe gesture instances. The precision and recall values are found to be 82.98% & 97.00% and 95.43% precision & 94.00% respectively for MGL and our system.

TABLE II. Performance of our Gesture Recognition System

Test Case	*Classified as Correct*	*Classified as Incorrect*
Correct Gesture	192	8
Incorrect Gesture	7	193

TABLE III. Comparative Performance for SWIPEs

Test Case	*Classified as Correct*	*Classified as Incorrect*
Using Microsoft Gesture Library		
Correct Gesture	194	6
Incorrect Gesture	37	163
Using Our System		
Correct Gesture	188	12
Incorrect Gesture	9	191

Our system shows a better precision (95.43%) than MGL (82.98%). Hence, MGL has less capability of handling the incorrect gestures and often classifies them to some gesture class, thus having a low precision. We use the adaptive threshold to avoid this issue. MGL, however, has a higher recall (97.00% vis-a-vis 94.00%), implying that a correct gesture has more chances of getting classified to the correct class. We have traded off on recall to get a very high precision, which is ideal for gesture recognition because executing a wrong command is usually worse than missing a command. Thus our system is less constrained for the user and more natural to use.

VI. Conclusion

We present a gestured based control system using Kinect. This involves the kinematic analysis, modelling and quantization of Kinect's skeletal data to recognize complex gestures. Instead of complex Gesture Spotting Network [9] to identify gestures out of the continuous hand motion; we propose a simpler, yet effective, way to work with the data for gesture recognition. Depending on the gesture we use either the last frame or the last 11 or 30 frames to recognize it. We engage a rule based method or machine learning for recognition. We use HMM's to handle the time series data and classify sequences, where a gesture can differ in shape and duration.

Our results show 96.48% precision at 94.00% recall with very low (only 8) false recognition. Hence our system adapts very well to avoid accidental activation. We compare our system with MGL [10] to show that we have better precision.

Acknowledgement

We acknowledge TCS for financial support.

References

[1] Accord. The Accord.NET Framework. URL as accessed on 16-June-2014: https://code.google.com/p/accord/, 2014.

[2] M. Azimi. Kinects skeletal joint smoothing white paper. URL as accessed on 16-June-2014: http://msdn.microsoft.com/en-us/library/jj131429.aspx, 2014.

[3] L. Dipietro, A. M. Sabatini, and P. Dario. A survey of glove-based systems and their applications. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38:461 – 482, 2008.

[4] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis. A hidden markov model-based isolated and meaningful hand gesture recognition. *International Journal of Electrical and Electronics Engineering*, 3:156–163, 2009.

[5] M. M. Hasan and P. K. Mishra. Hand-gesture modeling and recognition using geometric features - a review. *Canadian Journal on Image Processing and Computer Vision*, 3:21 – 26, 2012.

[6] P. Hong, M. Turk, and T. S. Huang. Constructing finite state machines for fast gesture recognition. In *IEEE Proceedings, 15th International Conference on Pattern Recognition (ICPR 2000)*, pages 691 – 694, 2000.

[7] S. K. Kang, M. Y. Nam, and P. K. Rhee. Color based hand and finger detection technology for user interaction. In *Convergence and Hybrid Information Technology, 2008. ICHIT '08. International Conference on*, pages 229 – 236, 2008.

[8] C. Keskin, A. Erkan, and L. Akarun. Real time hand tracking and 3D gesture recognition for interactive interfaces using HMM. In *In Proceedings of International Conference on Artificial Neural Networks*, pages 265 – 270, 2003.

[9] H.-K. Lee and J. H. Kim. An HMM-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21:961 – 973, 1999.

[10] Microsoft. KinectInteraction. URL as accessed on 09-July-2014: http://msdn.microsoft.com/en-us/library/dn188671.aspx, 2014.

[11] S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37:311 – 324, 2007.

[12] M. D. Network. Kinect for windows human interface guidelines v1.8.0. URL as accessed on 16-June-2014: http://msdn.microsoft.com/enus/library/jj663791.aspx, 2014.

[13] K. O'Hara, G. Gonzalez, A. Sellen, G. Penney, A. Varnavas, H. Mentis, A. Criminisi, R. Corish, M. Rouncefield, N. Dastur, and T. Carrell. Touchless interaction in surgery. *Communications of the ACM*, 57:70–77, 2014.

[14] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257 – 286, 1989.

[15] C. Souza. Hidden markov models in C#. URL as accessed on 16-June-2014: http://crsouza.blogspot.in/2010/03/hidden-markov-models-in-c.html, 2014.

[16] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Computer Vision, 1995. Proceedings., International Symposium on*, pages 265 – 270, 1995.

[17] E. Stergiopoulou and N. Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *Engineering Applications of Artificial Intelligence*, 22:11411158, 2009.

[18] L. Wilcox and M. Bush. Training and search algorithms for an interactive word spotting system. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 97–100, 1992.

[19] A. D. Wilson and A. F. Bobick. Parametric hidden markov models for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21:884 – 900, 1999.

[20] J. Wuchs, H. Stern, and Y. Edun. Parameter search for an image processing fuzzy c-means hand gesture recognition system. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on (Volume:3)*, pages 341–344, 2003.