REAL TIME GESTURAL INTERFACE FOR GENERIC APPLICATIONS

C. Keskin, O. Aran, L. Akarun
Computer Engineering Dept.
Boğaziçi University
E-Mail: ckeskin@cmpe.boun.edu.tr, {aran, akarun}@boun.edu.tr

Abstract

In this study we have developed a real time gestural interface based on 3D dynamic hand gesture recognition using Hidden Markov Models (HMM). We developed a system, which captures and recognizes hand gestures of the user wearing colored gloves, where the hand coordinates are obtained via 3D reconstruction from stereo.

The gestural interface provides supplementary features such as an interactive training and gesture defining system, a gesture tutor, a self-calibration utility for the cameras and a tool for linking the interface to different applications. The overall model is designed to be a simple and robust gestural interface prototype for various PC applications.

1 Introduction

The use of hand gestures is a noteworthy alternative to cumbersome interface devices for human-computer interaction (HCI). In particular, visual recognition and interpretation of hand gestures provides the ease and naturalness desired for HCI. For the visual recognition of hand gestures HMM has been used prominently and effectively for various applications.

Starner and Pentland implemented one of the earliest dynamic gesture recognition systems, where they used HMM to recognize American Sign Language using a single camera [1]. Oka, Sato and Koike have developed another 2D vision based system, where they made use of the Kalman Filter for noise elimination and HMM for gesture recognition [2]. In an earlier study the authors also employed a neural network for a 3D hand gesture recognition system [3].

In this paper we describe a gestural interface based on real-time hand tracking and 3D gesture recognition. The proposed system uses two color cameras for 3D reconstruction, hue based double thresholding for marker detection, 2D and 3D Kalman filters for noise elimination, and HMM for gesture recognition.

Two main concerns of gesture recognition are spatio-temporal variability and segmentation ambiguity. Spatio-temporal variability means that the same gesture can differ in shape and duration even for the same gesturer. HMM is inherently capable of modeling spatio-temporal time series. On the other hand, segmentation ambiguity arises, since the start and end points of a gesture are difficult to identify. We have constructed gesture models using left-right HMM, and re-estimated the parameters of each model with the Baum-Welch algorithm [4].

Another problem arising in real time gesture recognition systems is distinguishing meaningful input sequences from unrelated hand movements, i.e. gesture spotting. Lee and Kim proposed a method to provide an adaptive threshold for classification of gestures and nongestures using HMM. They developed an ergodic threshold model, which consisted of a combination of all the states of the models in the system [5].

The flowchart of the application is given in Figure 1. Each phase of the application will be explained in detail in the following sections.

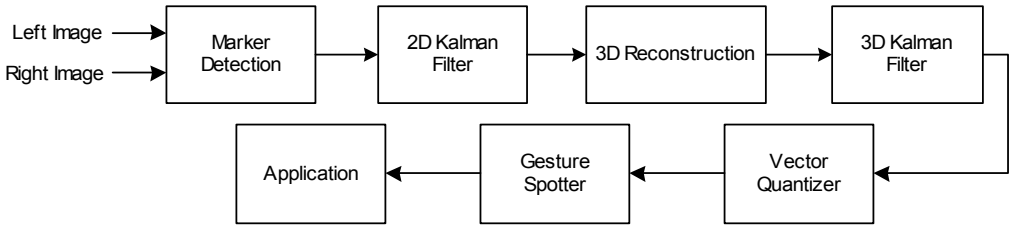

Figure 1 Flowchart of the gestural interface

2 Hand Tracking

The primary issues in tracking phase are the detection of the hand in the images, the reconstruction of the world coordinates of the user's hand and the elimination of the noise present in the system, i.e. filtering of the trajectory. We use markers to separate the hands from complex backgrounds under dynamic lighting conditions. If both hands are used at the same time the glove colors should be considerably different.

2.1 Distinguishing the Marker

The user can select any uniform color for the marker that is significantly distinguishable from the skin color. The user moves the marker after running the marker-detection utility and the difference image of two consecutive scenes with still backgrounds are obtained using a threshold. From this map the marker and its corresponding average hue component are found using a color histogram and region growing algorithm. This hue is used for marker segmentation thereafter.

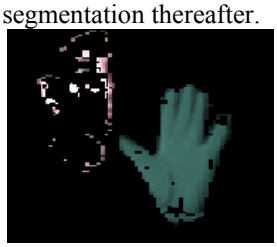

Figure 2 Difference image

2.2 Marker Segmentation

We employ connected components algorithm using double thresholding to find the marker region in images captured from the cameras. The regions with areas smaller than a threshold are regarded as noise. The pixels with grayscale values are not used in the region growing algorithm.

The elongation of the bounding box is used to determine the mode of the hand and the fingertip location is determined accordingly. The hand modes are used to simulate mouse events, and are explained in section 4 of this paper.

2.3 Filtering Marker Locations

The marker coordinates estimated are subject to noise caused by changing light conditions and inherent noise due to the discreteness of the images. Therefore online smoothing of the trajectories is a necessary step, especially for 3D reconstruction.

We assume that between consecutively captured frames the marker is linearly moving at constant velocity, subject to random perturbations in its trajectory. This assumption is convenient since sampling is done in short time intervals. Using Kalman Filter on this model has given satisfactory results. Figure 3 illustrates the measured and filtered trajectories for some common gestures.

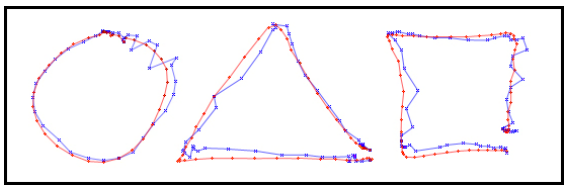

Figure 3 Effect of Kalman Filter

2.4 3D Reconstruction of Coordinates

For 3D reconstruction of the world coordinates, calibration matrices of both cameras are required. We have implemented a calibration utility within the system for a specific calibration object. The points are automatically found and matched using this utility.

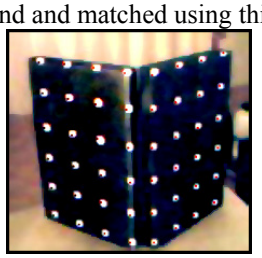

Figure 4 Calibration object

The world coordinates of the fingertip are generated by 3D reconstruction from stereo. Both the calibration and the 3D reconstruction methods are implemented using least squares approach. Since 3D reconstruction process is very sensitive to noise, even the filtered 2D coordinates yield a significant error. Therefore we filter the reconstructed coordinates using a 3D Kalman filter. Experiments revealed that such a double filtering method enhances the recognition results considerably.

3 Gesture Recognition

Once the coordinates are obtained for the images the system attempts to spot and recognize any gestures from the continuous hand motion stream. The hand motion stream does not only consist of trained gestures, since the system is designed to simulate the mouse movements as well. Therefore the start and end points of a gesture are not known beforehand and need to be found from the stream.

3.1 Gesture Spotter

To eliminate coordinate system dependence we transform the 3D coordinates of the marker in successive images into sequences of quantized velocity vectors. HMM based recognizer interprets these sequences, which are directional codewords characterizing the trajectory of the motion.

For a particular sequence to be recognized as a gesture, its likelihood should exceed that of the other models. To prevent recognition of nongestures the candidate model's likelihood should also exceed some threshold. Setting a static threshold value is impractical, since recognition likelihoods of gestures varying in size and complexity fluctuate significantly. Therefore we have

constructed the adaptive threshold model proposed by Lee and Kim [5] by fully connecting the states of all models.

To spot the endpoints of a gesture we regard each new input as a possible end point and check for candidate start points in an interval, whose limits are preset, but they can be changed by the user. For each such sequence the likelihoods of the models are calculated. If the likelihood of a model exceeds that of the threshold model for one of these sequences, the corresponding gesture is said to be recognized. (Figure 11) Otherwise classification is rejected.

4 The Gestural Interface Application

Common PC applications usually have a simple command interface that can be controlled via mouse selections or keyboard shortcuts. More advanced applications make use of the motion of the mouse as well, as in games or painting programs. The gestural interface is designed to work with all such applications.

4.1 Controlling Applications

The interface should be configured for each application by choosing the events to be triggered for detected hand gestures using a configuration file and a simple scripting language. The code of the target application is not needed, since the gestural interface sends the mouse and keyboard events through the operating system and not directly to the program.

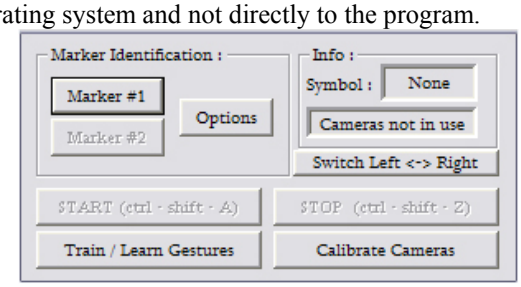

Figure 5 Application interface

The gestural interface simulates the mouse and keyboard events via hand motion and gesture recognition. In the simple one handed mode, where only one hand of the user is tracked, the hand motion is mapped to mouse movement and the elongation of the bounding box of the hand is used to decide whether the left click is pressed or not. In the two handed mode, where both hands of the user are tracked simultaneously, the mouse follows both of the hands in sequence, causing the mouse pointer to appear at two different locations. In this mode more advanced means of controlling the application is possible by using one of the hands for tracking and the other one for giving commands, which is not implemented yet.

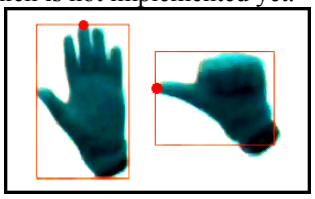

Figure 6 Hand modes

To start controlling an application, the markers must be registered with the interface. This can be done from the main application window by pressing the buttons labeled 'Marker #1' and 'Marker #2' and waving corresponding hands to the cameras. The second button is activated only in the 'two hands mode, which can be selected from the options menu along with other basic options and advanced parameters.

The registration of the markers allows the user to start tracking of the hands. To verify the calibration of the cameras the indicator on the info section of the main window can be used, which shows the motion direction symbols detected by the system. If the symbols are not correct, the calibration tool should be used to re-estimate the calibration matrices. The tool uses a known calibration object to estimate the camera matrices. The calibration parameters can be changed from the corresponding dialog.

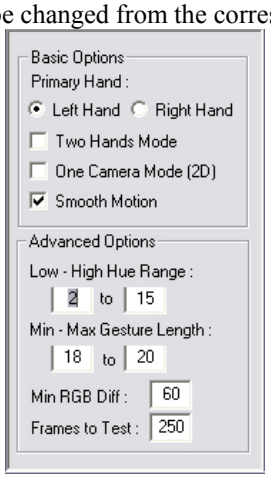

Figure 7 Options menu

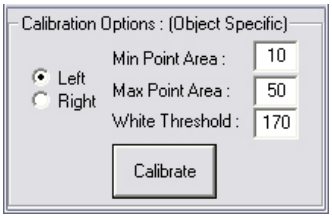

Figure 8 Calibration dialog

The keyboard shortcuts and mouse events are triggered when corresponding gestures are recognized in the tracking mode. The gestures that trigger these events need to be trained first. The training dialog has options to train a new or existing gesture, test if a gesture is recognized properly. It also has a tutor designed to interactively teach the users predefined gestures with provided movie files.

The user can choose the number of HMM states for a gesture, or let the system optimize the number of states used in training. Changing the number of states and adding a new sample re-trains the model with the new parameter.

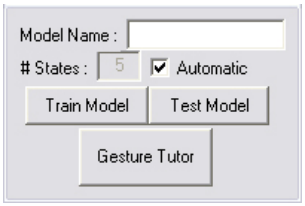

Figure 9 Training dialog

4.2 Application Testing

The application requires at least one camera that is capable of capturing uncompressed RGB images. For the 3D mode to work, two such cameras are necessary. One natural requirement arising from this is that the PC should be able to handle simultaneous interaction with two cameras. Current computers have enough bandwidth to support two cameras.

In our previous work [6] we tested the gesture recognition system for eight 3D gestures (Figure 10) with 60 training sequences for each model. The gestures were linked to specific commands of a third party painting application. With two misclassifications out of 160 trials, the system yielded a recognition performance of 98.75%.

Figure 11 shows the recognition process of a particular gesture. The graph displays the likelihoods of the models over time for the performed gesture. We see from the graph that at the tenth sample the likelihood of the dedicated model surpasses the one for the threshold model and recognition occurs.

5 Conclusion

In this study we have developed an application that works as a gestural interface for generic applications. The application is capable of tracking two hands and 3D motion in real time. First we have applied several image processing algorithms to detect markers in bitmap images acquired from simple web cameras. Then we have applied 3D reconstruction from stereo images. Next we have trained HMMs to recognize different gestures using the input sequences. In addition we have implemented an adaptive threshold model for gesture spotting. Combining all this work, we have developed a simple and robust framework where we can use gestures as inputs for interfaces for various PC applications.

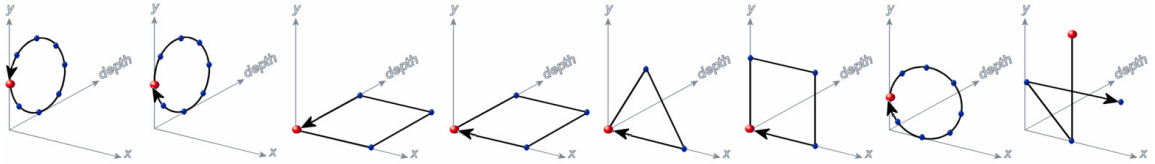

Figure 10 Trained 3D gestures

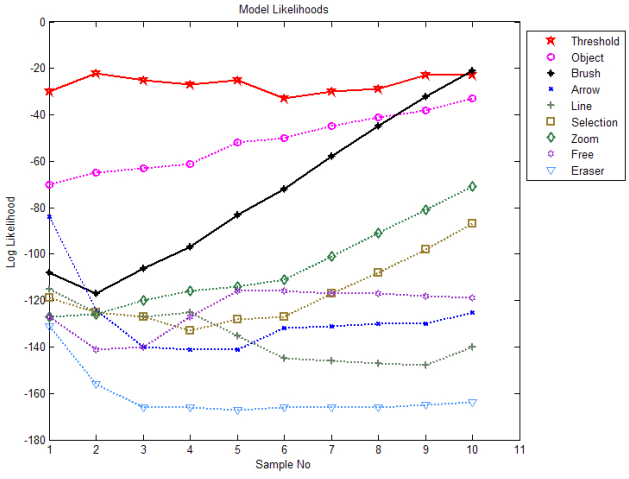

Figure 11 Likelihoods of the models

REFERENCES

[1] T. Starner and A. Pentland, "Real Time American Sign Language Recognition from Video Using Hidden Markov Models," Technical Report TR-375, MIT's Media Lab., 1995

[2] K. Oka, Y.Sato and H. Koike, "Real-Time Fingertip Tracking and Gesture Recognition", *IEEE Computer Graphics and Applications*, November-December, 2002, pp. 64-71.

[3] Y. Sato, M. Saito, and H. Koike, "Real-time input of 3D pose and gestures of a user's hand and its applications for HCI," *Proc. 2001 IEEE Virtual Reality Conference*, pp. 79-86, 2001.

[4] L.R Rabiner, "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc IEEE*, vol.77, pp.257-285, 1989

[5] H. Lee and J.H Kim, "An HMM-Based Threshold Model Approach for Gesture Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 961-973, 1999.

[6] C. Keskin, A. Erkan, L. Akarun, "Real Time Hand Tracking and 3D Gesture Recognition for Interactive Interfaces Using Hmm", ICANN/ICONIPP 2003