A Method for Temporal Hand Gesture Recognition

Joshua R. New
Knowledge Systems Laboratory
Jacksonville State University
Jacksonville, AL 36265
(256) 782-5103

newj@ksl.jsu.edu

ABSTRACT

Ongoing efforts at our laboratory have been aimed at developing techniques that reduce the complexity of interaction between humans and computers. In particular, we have investigated the use of a gesture recognition system to support natural user interaction while providing rich information content. In this paper, we present additions to a platform-independent, gesture recognition system which previously tracked hand movement, defined orientation, and determined the number of fingers being held up in order to control an underlying application. The additions provide the functionality to determine a temporal gesture, such as movement of the hand in a circle or square. This technique has proven to replace the common "clickfest" of many applications with a more powerful, context-sensitive, and natural interface. In addition, the Simplified Fuzzy ARTMAP (SFAM) learning system could easily be trained to recognize new gestures that the user creates and map these to a specific action within the application. Sample images and results are presented.

Keywords

Temporal gesture recognition, SFAM, machine vision, computer vision, image processing, human-computer interaction.

1. INTRODUCTION

As today's computer systems grow increasingly complex, the need arises to increase the capability by which humans interact with these computer systems. Computer vision techniques have been applied to this end with significant success. Here, the computer system processes image input from a camera to gain information about the user's intentions. This technique has proven very effective at augmenting standard input devices such as the mouse and keyboard [1]. The reader is referred to [2, 3] for more information. Since this approach requires no additional encumbrance, computer vision allows user-interaction information to be gathered without requiring special hardware or being intrusive.

2. The Previous Gesture Recognition System

The user is seated naturally at his desk, simply moving his/her mouse hand into view of the camera to begin interaction with the system (see Fig. 1). The system uses an off-the-shelf web camera to capture the desk area where the user's hand is located. This image is then processed in multiple, pipelined stages (see Fig. 2). The data from this processing is then used to update a 3D skull, allowing roll, pitch, yaw, and zoom to be updated based upon properties of the hand. The user then continues interaction with the system based upon the previous manipulation (see Fig. 3). The only information that is used from the previous recognition system is the centroid, which defines the location for the center of the hand in Cartesian, pixel coordinates.

Figure 1. Typical method of interaction.

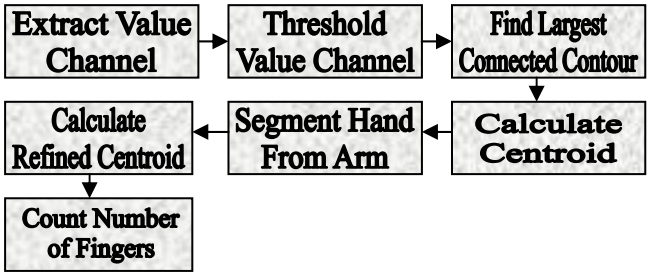

Figure 2. Design and components of system pipeline.

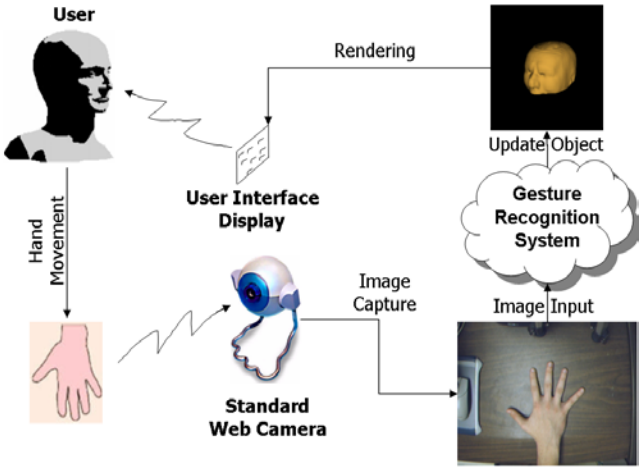

Figure 3. System Diagram

3. TEMPORAL RECOGNITION

The gesture recognition system was developed using the Microsoft Visual C++ compiler with the Image Processing Libraries (IPL) and Open Computer Vision library (OpenCV) from Intel. These libraries contain special image-processing functions optimized for the Pentium architecture to ensure efficient computational performance.

The system developed to date comes with recognition capabilities of circle, square, up/down, and left/right, but can be easily trained to recognize a diverse set of gestures. This is done by each of the successive stages for temporal gesture recognition:

1) Collect 12 centroids
2) Start at Top-left
3) Compute centroid differences
4) Normalize using perimeter
5) Format for SFAM
6) Write training file
7) Classify with SFAM

3.1 Collect 12 Centroids

The gesture recognition system previously developed is capable of tracking hand movement, also known as the centroid or Center of Mass (CoM). In order to perform temporal recognition of gestures, it was found that the average gesture takes approximately 3 seconds. The current gesture recognition system works at 22 frames per second, but also uses an interface for manipulating a 3D object, which works at approximately 4 frames per second. Thus, the centroid was tracked for 12 frames (3 seconds at 4 frames per second). Since continuous and immediate/interactive feedback was desired, these 12 frames are captured continuously and classified. However, in a practical system application, this would have to be changed by utilizing a "start gesture" gesture or be smart enough to recognize when someone is not gesturing.

These twelve centroids provide a "connect-the-dots" picture of what the hand has done for the past three seconds. This information is stored in a 24-value array as $(x_1, y_1, \ldots, x_n, y_n)$ and then passed to the next stage. In addition, the system is designed to be expandable so that if a higher frame-rate is capable, the user need only change one variable from 12 to the desired number of frames to be used for recognition and all relevant system changes are made. The only limitation is that additional training sets must be provided for the new setup, either by recording gestures or writing a sub-sampling or up-sampling routine.

3.2 Start at Top-Left

Due to the decision that the system will attempt to classify gestures every 12 frames, there was a complication that arose with robust classification. An example demonstrates this best. Let's say that a user is performing the gesture we all know to be

"square". The complication arises in that if the user isn't in-sync with the recognition phase, they could be starting at any corner of the square or any point along the square. Needless to say, the gesture for a square starting at the top left is quite different (opposite in fact) from starting at the bottom right. Due to this concern, the decision was made to start at the top-leftmost point and then continue from there. This would allow rotational sensitivity (clockwise square vs. anti-clockwise square) while being independent of the starting point.

The top-left coordinate is found by finding the minimum of (x_j,y_j) and the index "j" is stored. All contents of the centroid array are then cyclically shifted so that the values previously stored in the j^{th} index are now in the first.

3.3 Compute Centroid Differences

Since human gesturing is, by nature, imprecise when it comes to repeatability, it is important that a gesture recognition system be robust to gestural fluctuations. This is done by taking the difference between each set of coordinates.

The last value in the array will store the difference between the first and last x and y values. For example, let's say our array of coordinate pairs looks like (x_1=1, y_1=2, x_2=1, y_2=4, x_3=4, y_3=4, x_4=4, y_4=2, x_4=1, y_4=2) stated more simply as (1,2, 1,4, 4,4, 4,2, 1,2). The difference would be (0,2, 3,0, 0,2, -3,0, 0,0) where the first 0,2 comes from (x_2-x_1,y_2-y_1) = (1-1,2-2). The last entry in the array becomes the first minus the last. Here, +x corresponds to move right, -x to move left, +y to move down, -y to move up.

3.4 Normalize Using Perimeter

As stated previously, it's important to demonstrate robust recognition of "similar" gestures. While the difference introduced previously quantifies the direction of movement, it does not specify the magnitude of that movement. In order to do this, a perimeter for each direction is computed where $P_x = \Sigma |x_n|$ and $P_y = \Sigma |y_n|$. All the values that have been differenced are then divided by the appropriate perimeter. That is, $x_j = x_j/P_x$ and $y_j = y_j/P_y$. This gives the inputs magnitude by defining a certain percentage of the x or y movement to be in the current direction. For example, a given input of x_j now codifies the knowledge, "the user moved his hand to the right by n% of the entire x movement." As can be seen, this makes the inputs codify scale-independent gestures. Therefore, a gesture we know to be "square" will be "square" whether the user makes a large, medium, or small square. Again, this is necessary due to the non-uniformity of human gesturing.

3.5 Format for SFAM

The learning system used is known as Simplified Fuzzy ARTMAP (SFAM) developed by Grossberg. An in-depth treatment of the SFAM system is beyond the scope of this paper, but the interested reader is referred to [4]. SFAM is a fast, online, incremental/interactive learning system that was chosen since its performance capabilities elegantly satisfied the requirements of the problem. This learning system was implemented in C++ using the object-oriented paradigm. For ease-of-use, initialization, training, classifying, and other such necessary actions were created as methods of the SimplifiedFuzzyARTMAP class.

One feature that allows the learning system to quickly and efficiently categorize data is that the input features must be between [0,1], where the input features correspond to the array described in Section 3.4 with additional complement coding.

In order to assure that the original array is between [0,1], the extreme cases must be analyzed. The most extreme case is where the hand moves some amount between frame two consecutive frames, but remains still the rest of the entire time. While this is not possible in practice, it is the theoretical limit of the system. In the example described, the features would be (-1,-1) if the user moved their hand up and left or (1,1) down and right [with all other features being (0,0) for no movement]. The most efficient way to reduce [-1,1] to [0,1] while preserving order is to divide by two and then add 0.5. This is the manner in which all inputs are condensed to a region that is required by the learning system while preserving the necessary relations.

One small concern with this approach is that, while it is guaranteed to normalize the inputs, for most gestures that move continuously, each movement is a relatively small percentage of the total. By dividing by two and adding 0.5, most inputs will be very close to 0.5. It is the case that information is being lost, but with 15 values after the decimal point, the amount of lost information proved to be a non-issue.

In addition, the inputs must be complement-coded. This means that the "opposite" of the values must be tacked onto the end of the array, where "opposite" means $1-x_j,1-y_j$. Once the values have been normalized between [0,1] and complement-coded, they are ready to be stored and/or used by SFAM.

3.6 Write Training File

It is a necessity of learning systems that a representative training set be provided by which the learning system can be trained to recognize gestures. The current system allows the user to "record" their hand movement and dump this data to a text file, which can be reformatted to be learned by the recognition system. For example, to train the system to recognize the "square"

gesture, you need only turn on the writing feature (a boolean variable) and move your hand continuously in a square motion. The user must then rename the inputs.txt file created to "ARTMAP_Train.txt". This file must then be edited by adding "#samples #features" to the top line. The #samples used in some versions of the system is 239 (239 sets of 12-frame recordings, the author's arm was very sore afterward) and all use 48 as the #features (12 x-y coordinates pairs = 24 coordinates = 48 complement-coded inputs). In addition, an integer code at the end of each line to signify the gesture; for example, 1 for circle, 2 for square, etc. Multiple files can be created and synthesized for the training of the learning system.

3.7 Classify with SFAM

When the system first starts, an SFAM network is initialized and trained according to the ARTMAP_Train.txt file described earlier. Once the inputs are pre-processed in the manner described in Sections 3.1-3.5, the array's data is copied into a matrix and then classified by the SFAM system using the classify method. The classification (1-circle, 2-square, 3-left/right, 4-up/down) is then printed to the screen. While this is a sufficient process to determine the performance of the learning system, much more useful and powerful techniques are possible. These are discussed in more depth in the future work section.

4. FUTURE WORK

The current system classifies correctly a majority of the time. The failure that remains is mostly due to the forced recognition of every 12 frames. While the current system is nicely tailored to the computationally efficient manner of robust classification, there are many additions that could be made.

4.1 Temporal Gestures

One primary shortcoming of the temporal gesture recognition system developed is that every 12 frames are recorded and recognized, whether the user is performing a gesture or not. This means that while the user is gesturing, recognition is very good. However, when the user is doing something besides gesturing, or nothing at all, the system will falsely classify the gesture as something it knows.

Another complication is that the static frame-count used can lead to the incorrect classification of a gesture if a user switches between gestures in the middle of the 12-frame recording phase. While this definitely leads to errors, the chances of a correct classification are still high, depending on how early in the recognition phase they begin the desired gesture. Also, while many gestures (all used in this system) consist of loops, the user typically repeats the gesture continuously for several recognition phases, in which case, the system will categorize correctly a majority of the time.

4.2 Spatial Gestures

While the current system is sufficiently good at recognizing gestures from hand movement, it would be more powerful to combine it with spatial recognition as well. That is, instead of recognizing if the user is moving their hand in a hook formation, determine if they're making an American Sign Language "j". Current approaches vary from recognition based on edge-detection to interior pixel values [5, 6].

4.3 Learning System

While SFAM proved both easy and efficient as a gesture recognition system, other learning systems should also be considered. For example, hidden Markov Models have been used successfully in the realm of gesture recognition [7-10].

Considerations involved in the design of a learning system would include invariance to hand pose, selection of the training set to support robust performance across multiple users and/or lighting conditions, and the type of training system utilized (i.e. off-line vs. on-line learning, supervised, etc.) A given learning system could also support the calibration of the system by adapting the system parameters as a function of current conditions.

4.4 Interface Extensions

The system developed simply allows the analysis of classification in order to determine the behavior of the learning system. It would be much more powerful to use this information to drive an interactive system that could adapt to the user's desires. One can easily imagine an application where players in a real-time strategy game simply select troops and drag their hand in a box formation to tell the troops to line up with weak units inside and strong units facing outward [11].

Previously, similar recognition systems have been used successfully to aid user's with reduced mobility or impaired motor skills, such as those who suffer from cerebral palsy. While the current system has not been evaluated for those uses in that context, relatively easy extensions could be made to address such applications.

Other contexts and applications of the system are being considered including interaction with graphical user interfaces utilizing a "graspable interface" [12] similar to that used in Spielberg's "Minority Report" film [13]. Other examples include interaction with multiple objects in a virtual environment [14] and its use in augmented reality applications in order to enable transparent interaction.

5. REFERENCES

[1] Kjeldsen, F., "Visual Interpretation of Hand Gestures as a Practical Interface Modality", Ch. 3, 29-36, Ch. 4, 6-7, Columbia University, 1997.

[2] New, J.R., "A Method for Hand Gesture Recognition". In Proceedings of the ACM Mid-Southeast Chapter Fall Conference, Gatlinburg, TN. November, 2002.

[3] New, J.R., Hasanbelliu, E. and Aguilar, M. "Facilitating User Interaction with Complex Systems via Hand Gesture Recognition." In Proceedings of the 2003 Southeastern ACM Conference, Savannah, GA. March, 2003.

[4] Grossberg S., Neural Networks and Natural Intelligence, Cambridge, MA: MIT Press, 1988.

[5] Ernst, H., Schafer, K., and Bruns, W., "Creating Virtual Worlds with a Graspable User Interface". *15th Twente Workshop on Language Technology*, University of Twente, NL, 1999.

[6] Kjeldsen, F., "Visual Interpretation of Hand Gestures as a Practical Interface Modality", Ch. 3, 29-36, Ch. 4, 6-7, Columbia University, 1997.

[7] Chen, F.S., Chih, M.F., Huang, C.L., "Hand gesture recognition using a real-time tracking method and hidden Markov models". In Proceedings of the Image and Vision Computing 21, Taiwan, March, 2003.

[8] Morimoto, C., Yacoob, Y., Davis, L. "Recognition of Hand Gesture Using Hidden Markov Models." 1996.

[9] Meyer, D. "Human Gait Classification Based on Hidden Markov Models". 1998.

[10] Lee, C. and Xu, Y. "Online, Interactive Learning of Gestures for Human/Robot Interfaces". 1996.

[11] Buckland, M. and LaMothe A., AI Techniques for Game Programming, Cincinatti, Ohio, 2002.

[12] Ernst, H., Schafer, K., and Bruns, W., "Creating Virtual Worlds with a Graspable User Interface". *15th Twente Workshop on Language Technology*, University of Twente, NL, 1999.

[13] Clarke, D., "MIT grad directs Spielberg in the science of moviemaking", *MIT Tech Talk*. July 17, 2002.

[14] Furness, T., Exploring Virtual Worlds. ACMMEMBERNET, Vol. 34. No. 7, July 1991.