A Novel System for Hand Gesture Recognition

Matthew S. Vitelli
mvitelli@stanford.edu

Dominic R. Becker
drbecker@stanford.edu

Thinsit (Laza) Upatising
lazau@stanford.edu

Abstract — *The purpose of this project is to create a real-time dynamic hand gesture recognition system from front-to-back. Users interact with the system by wearing a special glove. Motions from the user are interpreted by our application running on standard computer hardware with a commodity webcam. These motions are analyzed using computer vision and machine learning, in particular Hidden Markov Models, in order to determine which gesture is being made. Over time, the user may train the system to adapt to and learn new gestures.*

I. Prior Works

Hand gesture recognition has received a great deal of attention in recent years. Due to its many potential applications to mobile technology, gaming systems, and real-time imaging technologies, it has become an area of increased interest.

Hand gesture recognition has been explored by many researchers using a variety of methods. Visions of Minority Report-like computer interaction are becoming somewhat feasible. Mistry et al. present a wearable projector-and-camera setup that recognizes hand gestures acting on the projected images [9]. Google Glass promises similarly futuristic gesture-augmented reality interaction.

Other explorations include using the Microsoft Kinect, which has a built in stereoscopic sensor. Ren et al. recognize static hand gestures using a modified Earth Mover's Distance metric [4]. Biswas and Basu recognize upper body gestures using Kinect depth data and SVMs [5].

As early as 1994, Yang and Xu used Hidden Markov Models (HMMs) to recognize gestures drawn with a mouse on a computer [6]. In 1995, Starner and Pentland built an HMM-driven system to recognize American Sign Language [7]. Keskin et al. created a 3D gesture recognition system that also uses HMMs.

II. Purpose

Many proprietary computer vision systems that can detect the location of a hand exist in the market today. These technologies, such as Microsoft's Kinect or Leap Motion's The Leap, can be used as an input device for a gesture recognition system. However, these devices can be quite costly. Our goal is to make a gesture recognition system that can take data from any device and perform gesture recognition. Currently, there is no standard data format for gesture recognition devices, however, we hope that proprietary computer vision systems will eventually adopt one – this development will allow our system to perform gesture recognition on any input device that supports the standard data format.

In this project we create a modular system in which a custom-made input device recognizes the location of fingertips, outputs the data into a standard text file and a separate system reads the data in real time and performs gesture recognition. Our system is highly modular so that gesture recognition can be performed using any input device that recognizes fingertips and output the data in a known format.

III. Method: Vision

One major system in our project is the custom-built input device which draws together technology from different fields such as computer vision and basic circuitry.

A. *The Glove*

Users interact with our system using a custom-made glove. The glove is fitted with 4 different LED bulbs, each with a unique color. Since each brightly colored LED corresponds to a unique finger, the process of recognizing fingertips is simplified down to extracting brightly colored blobs from an input image.

B. *Computer Vision*

The user's fingertips must be correctly identified in order to accurately track their gestures. To accomplish this, the image captured by the webcam must be properly processed to identify the position of the user's fingertips, as well as categorize each finger. The vision process can be broken down into several stages as follows:

1. Threshold Pass – The image is thresholded to extract the brightest pixels. The benefits of this process are that most of the background is eliminated and most the brightest pixels are likely candidates for the LEDs of the glove.

2. Convolve Pass – The image is then convoluted using a special kernel that favors brightly colored pixels over white light. Since most of the LEDs appear oversaturated in the camera image, this pass is useful for approximating the true colors of the individual LEDs.

3. Downsample Pass – The image is then downsampled to a low resolution for later use during centroid estimation.

4. Dilation Pass – The image is dilated to increase the size of each region and provide better centroid estimates.

5. Centroid Estimation – The centroids of each blob in the image must be computed to accurately measure the position of each fingertip. To perform this task, we used a recursive flood-fill algorithm. Essentially, the algorithm scans through each pixel in the image and finds all pixels connected to the current pixel. Because the algorithm needs to be performed at every frame, we use a downsampled image to reduce the number of computations necessary. Using this approach, we can easily compute the centroids and get accurate position measurements.

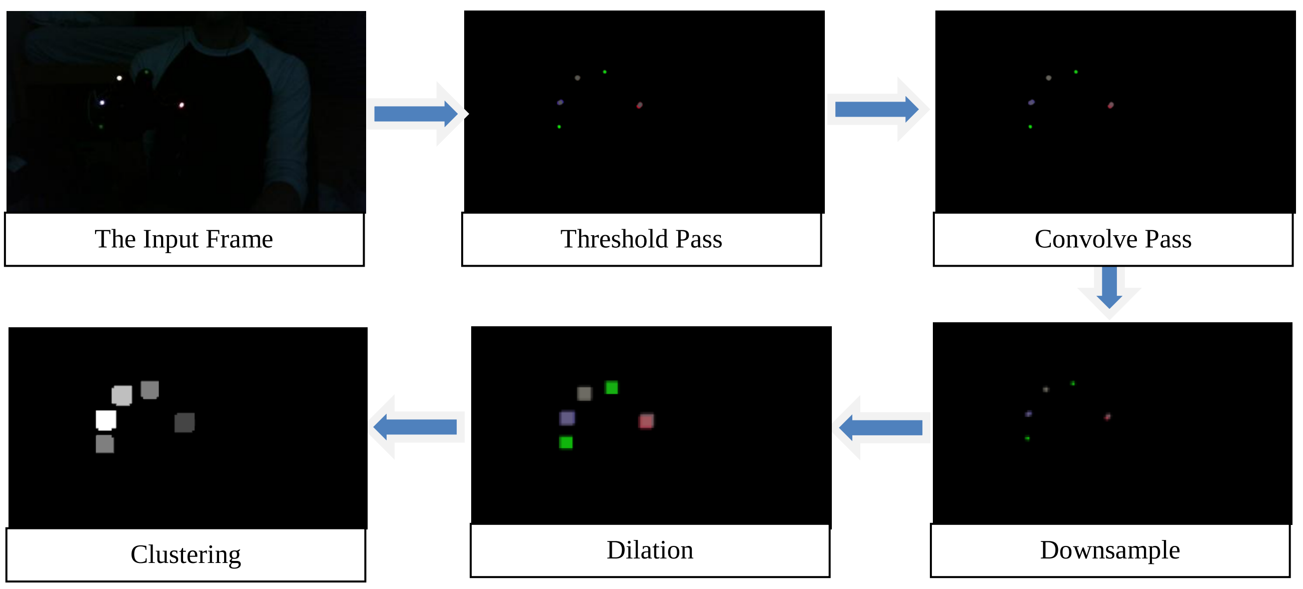
To increase the performance of our vision system, we parallelized steps 1-5 to run entirely on the GPU using programmable shaders. The system's capture pipeline utilizes DirectX to communicate with the GPU and perform data processing.

proved to be cumbersome, as we wanted our gestures to be invariant to time.

In an attempt to overcome this, we normalize each finger's velocity vector in order to compute the raw direction. However, informally, this does not seem to improve recognition of gestures that are made more quickly. The reason for this seems to be the sample rate of the data: if the gesture is made too quickly, only a few frames are captured by the camera—and these may not include important frames in the middle of the gesture, which make the gesture less recognizable.

B. Quantizing Feature Data

Before feeding the features into the Hidden Markov Model, each frame's feature data—the normalized x and y velocities for each finger—is quantized using a codebook generated by a clustering algorithm. This is primarily done to group similar features across frames together (thus reducing the size of the dataset), as well as to discretize the feature-space for later use in the Hidden Markov Models.



IV. METHOD: LEARNING ALGORITHM

Based on the literature, it seemed that Hidden Markov Models would appropriately model the four-fingered hand gestures that we hoped to recognize. Given the input data: x-y coordinates per finger over time, it made sense for our feature extraction to follow a similar pipeline to that in Yang and Xu [6]. As such, the feature data is quantized using a clustering algorithm before it is fed into the HMM.

A. Feature Selection

We experimented with a variety of different feature models and representations of the feature space. Our first approach incorporated velocity data from each fingertip; however this

In particular, we implemented the LBG algorithm, due to Linde, Buzo, and Gray, to perform the clustering. Yang and Xu employ this clustering algorithm to 99.78% accuracy with 100 samples of training data for mouse gesture recognition [6].

Using the codebook, each input feature per-frame is classified into a given cluster, and the observation sequence is transformed to a sequence of the clusters corresponding to the nearest centroid in the generated codebook to each frame's feature vector. Again, in order to recognize a gesture, the frame features are quantized using this LBG-generated codebook.

C. Hidden Markov Models

Hidden Markov Models are used to predict which gesture the user is currently performing. One model is generated for each gesture. The HMMs are trained by taking a collection of the codebook-discretized sequences, used as the actions of the Hidden Markov Model, corresponding to each raw training sample. The HMMs are trained using the Baum-Welch re-estimation algorithm either until convergence or to a maximum of 500 iterations (for the sake of timely model generation). This training is done offline as it cannot be completed in an acceptable amount of time for an end user to interact with directly (i.e. on the order of hours).

Once the models are built, on the other hand, recognition is performed in real-time. During recognition, the user's current input gesture is first quantized using the process described above. Next, the Viterbi algorithm computes the likelihood of the quantized observation sequence given each model. Selecting the model that maximizes the likelihood, our application is able to guess which gesture the user is performing.

V. Results and Analysis

We tested our system under a number of different parameters, including various numbers of clusters and Markov transitions. We also performed diagnostic tests with normalized and unnormalized feature data. Due to the fact that computing Hidden Markov Models is a time-consuming process, we were only able to capture a limited number of varying transition states and cluster sizes. Ultimately, we settled on 16 unique clusters with 4 Markov transition states.

We tested our results using hold-out cross validation, training on 70% of the data. The data consists of eight gestures, each with around 200 training samples. For the final presentation, we retrained the Hidden Markov Models with all of the available training data, and did not notice any significant drop in accuracy.

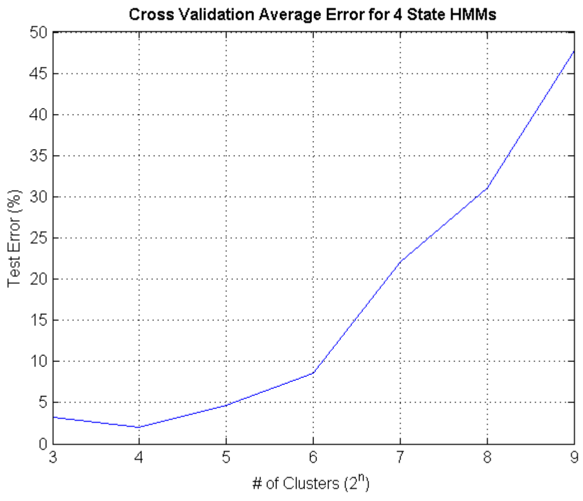


Figure 1

A. Number of Clusters

Figure 1 shows the average accuracy over eight gestures of a four state Hidden Markov Model trained over a varying number of clusters. It is apparent from the image that increasing the number of states can actually detract from the Hidden Markov Model's performance. Figure 2 shows the normalized and unnormalized 256 clusters generated by our algorithm on only four simple gestures: horizontal and vertical gestures (see appendix). The figure shows that having too many clusters will cause the algorithm to begin differentiating between motions that are extremely similar, which is undesirable. Figure 3 is 16 clusters generated by all eight gestures, we can see that lowering the number of clusters will allow the algorithm to recognize principle motion directions without causing similar gestures to be classified as different clusters.

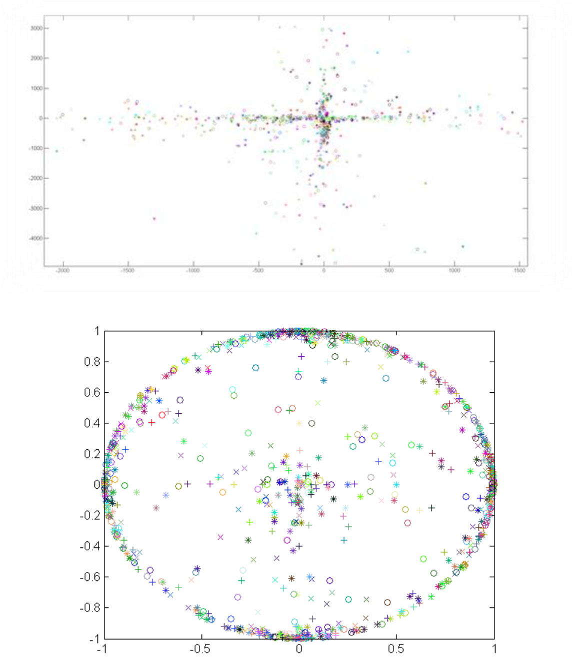


Figure 2

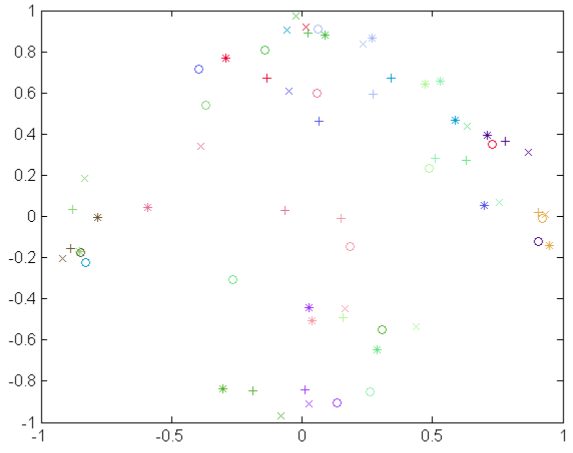
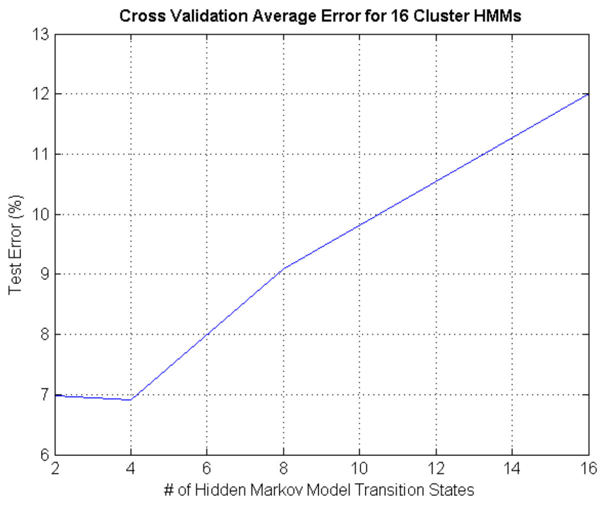


Figure 3

Figure 4



B. *Number of Hidden Markov Model States*

We can see from the Figure 4 that the optimal number of states in the Hidden Markov Model is 4. We thought that increasing the number of states in the Hidden Markov Model would allow the model to capture more states that represent the user's gesture. However, empirical data shows otherwise. We postulate that this may be due to the limited number of training samples that we obtained – a closer analysis of the emission matrices for Hidden Markov Models with more than 8 states shows that many of the emission probabilities were too low.

VI. Future Work

A. *Live Recognition*

Having to click a start-stop button to recognize an individual gesture is inconvenient. In particular, using gesture recognition as an input method would be infeasible if the user needed to indicate the beginning and end of each gesture. Instead, it would be ideal for the system to automatically determine when a gesture has been made. One way to do this would be to identify gestures by applying some threshold to the likelihoods generated by the Viterbi algorithm. While the basic idea would be to run the Viterbi computations at some per-frame interval, issues may arise such as what data to include (last 20 frames, last 2 seconds, etc.).

B. *More Flexible Input Data*

Our current training and recognition system accounts for exactly four fingers. If a finger is hidden during data capture (or another is added), the data captured becomes very erratic. It would be ideal to simply remove such data before feeding it into the model. However, with such different data sets, there would have to be more data, perhaps encapsulated in different Markov Models, with/without those corresponding features. A system that handled fewer or more fingers could be much more flexible in terms of practical usability.

C. *Improved Feature Selection*

Certain gestures are harder to recognize than others. With only finger velocities as features, gestures like circles are difficult to recognize. In many of the gestures that were successfully recognized, the finger positions relative to one another were constant. For other gestures though, say a snap of the fingers, additional features like relative position may be more valuable. Another feature manipulation to explore is normalization: better normalization may lead to improved recognition regardless of temporal length of the gesture.

VII. Conclusion

We successfully prototyped a front to end gesture recognition system using Hidden Markov Models and a custom built input device. The system is highly accurate for the majority of the gestures in our database. While we successfully prototyped a flexible system for hand gestures, this project just scratches the surface of what is possible. Given more time, we would like to increase the complexity of our gestures, as well as the number of gestures used in our system. Additionally, we would like to parallelize more of our codebase to accelerate the process of training the clusters and Hidden Markov Models.

VIII. Acknowledgement

We gratefully acknowledge Professor Andrew Ng for valuable feedback on our project and the excellent lecture notes on Hidden Markov Models.

IX. References

[1] Pavlovic,V.: Dynamic Bayesian Networks for Information Fusion with Applications to Human–Computer Interfaces, Dept. of ECE, University of Illinois at Urbana-Champaign, Ph.D. Dissertation, (1999)

[2] Stenger, B.: Model-Based Hand Tracking Using a HieraDynamic Time Warping

[3] Blob Recognitionrchical Bayesian Filter (2006).

[4] Ren, Zhou, Junsong Yuan, and Zhengyou Zhang. "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera." Proceedings of the 19th ACM international conference on Multimedia. ACM, 2011.

[5] Biswas, K. K., and Saurav Kumar Basu. "Gesture Recognition using Microsoft Kinect®." Automation, Robotics and Applications (ICARA), 2011 5th International Conference on. IEEE, 2011.

[6] Yang, Jie, and Yangsheng Xu. Hidden markov model for gesture recognition. No. CMU-RI-TR-94-10. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1994.

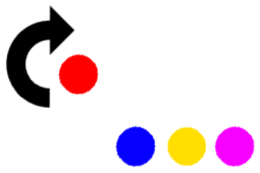
[7] Starner, Thad, and Alex Pentland. "Real-time american sign language recognition from video using hidden markov models." Computer Vision, 1995. Proceedings., International Symposium on. IEEE, 1995.

[8] Keskin, C., A. Erkan, and L. Akarun. "Real time hand tracking and 3d gesture recognition for interactive interfaces using hmm." ICANN/ICONIPP 2003 (2003): 26-29.

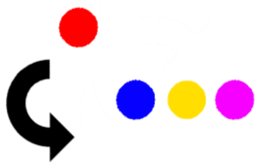
[9] Mistry, Pranav, Pattie Maes, and Liyan Chang. "WUW-wear Ur world: a wearable gestural interface." Proceedings of the 27th international conference extended abstracts on Human factors in computing systems. ACM, 2009.

X. Appendix

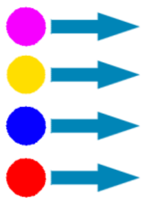
Here are the eight recognized gestures:



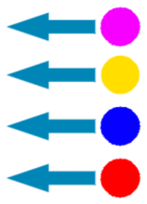
Thumbs Up



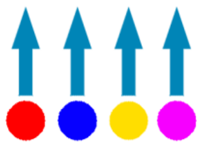
Thumbs Down



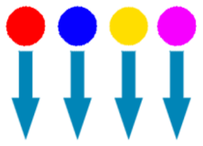
Swipe Right



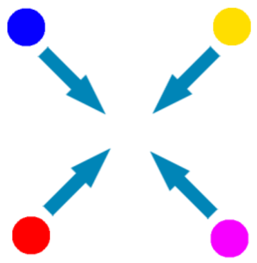
Swipe Left



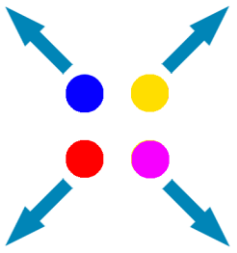
Swipe Up



Swipe Down



Pinch In



Pinch Out