MouseFree

Vision-Based Human-Computer Interaction through Real-Time Hand Tracking and Gesture Recognition

Dept. of CIS - Senior Design 2009-2010

Chris Jordan
wjc@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Hyokwon Lee
hyokwon@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Ben Taskar
taskar@cis.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

ABSTRACT

Vision-based interaction is an appealing option for replacing primitive human-computer interaction (HCI) using a mouse or touchpad. We propose a system for using a webcam to track a user’s hand and recognize gestures to initiate specific interactions.

The contributions of our work will be to implement a system for hand tracking and simple gesture recognition in real-time. We also plan to create a technique which allows HCI without the need for the user to touch or carry a peripheral device. Third we intend to build and share a dataset of diverse hand tracks so that future work on hand tracking and gesture recognition can be compared against a public benchmark.

1. INTRODUCTION

The basic goal of HCI is to improve the interface between users and computers by simplifying the interactions between them. Although the computers themselves have advanced tremendously, the common HCI still relies on simple mechanical peripherals which reduce the effectiveness and naturalness of the such interactions [10]. With computers shrinking in size, the need for hands-free HCI is becoming increasingly important so that devices such as mice, trackballs, and touchpads can be phased out. A possible alternative is the use of a vision-based interface. Fortunately, most of the new computers being sold come standard with a webcam. By using webcams, incorporating a real-time hand tracking and gesture recognition system, and developing a driver to interpret input, users will be able to emulate the functions of a mouse and interact with a computer using hand motions.

A fairly recent project that has shown great promise in making human-computer interaction more natural has been SixthSense / WUW - Wear Ur World. This is a system that uses a digital camera, mini projector, and marked fingertip tattoos to interact with a projected screen on the move wherever you are. Projection system make an ideal candidate for vision-based interactions because of the inability to use a touch screen. However, it is unfortunate that the limitations of wearing markings on the hand exist within this system. This system first appeared in publications in December of 2009 and is a promising candidate for advancing this form of technology.

Another great example of the potential of vision-based interaction is the Xbox 360 Project Natal. Project Natal is a Microsoft project which focuses on creating an interface for the Xbox 360 using an RGB camera for facial and gesture recognition and other sensors which capture full motion of gamers. The unveiling of this system demonstrated that the technology to allow interaction through body movement without the need for any controllers was possible. However, there has been little published on this project and its performance is untested by academic standards. Also the interaction is designed specifically for the constraints of the game system.

Sony also had a vision-based interaction peripheral for the Playstation 2 called the EyeToy in 1999. This peripheral captured motion using color detection, allowing gamers to interface with selective games using body motion.

Then there was the Wii. The Wii system has been manipulated by home enthusiast Johnny Lee who used the infrared camera in the Wii remote, the IR LED’s, and reflective tape to track the tips of his fingers moving through the air. Although neither of these was on the level of Project Natal, they all shared similar goals and concepts of using a vision based system for interaction in place of typical game controllers.

A vision-based interface for HCI involves the use of object detection, determining whether or not an object exists in a frame and tracking, the following an object from frame to frame. Our system performs these tasks using variations on existing methods, the Viola-Jones Object Detection method and the Lucas-Kanade Optical Flow method. Our system also performs gesture recognition, identifying a given gesture in a frame, which will be a key in allowing us to emulate the functionality of a mouse. Our system accomplishes this task by the Eigen Objects method commonly used in face recognition. Along with gesture recognition, our system performs segmentation and background subtraction, which is the separating of the foreground object from the background of the image, to help reduce the noise when recognizing gestures. To improve the performance of our object detection, we use Adaptive Boosting (AdaBoost), an iterative supervised learning method, using Yoave Freund’s and Robert Schapire’s Adaptive Boosting method [11]. Furthermore, we will be implementing a database to store hand tracks for future training data. This will also give future work involving hand sets data to start working with.
2. RELATED WORK

Several different approaches have been taken trying to track hands and recognize hand gestures. Our goal is to allow the end-user to interface with the computer without the use of a mouse or other hand-held peripheral. Rather we propose a system consisting just of a standard webcam facing the user. This means that the end-user will not need gloves, special gel, or a wearable camera to use their hand as a mouse to interact with the computer in real-time. We achieve good results tracking bare hands by using methods which were effective for face detection and tracking and preprocessing techniques of registration and background subtraction.

Robert Y. Wang and Jovan Popovic had a very interesting approach of combining color markers in the form of a wearable multi-colored glove to make tracking the hand and pose estimation simpler. Due to the unique pattern on the colored glove the detections and pose estimations were faster and more robust [15]. Even though, the glove is just a plain cloth glove with a unique colored pattern and no devices are attached, the user is still constrained by the need to wear a glove for tracking and pose recognition to work.

Thomas A. Mysliwiec came up with a bare-handed method to use a finger as a mouse pointer [9]. The idea used a stationary top down view camera. The user was allowed to switch between using the keyboard and using one of the hands as a mouse pointer by assuming a hand position. Although this alleviated the need for a mouse, the two hands were still fully restricted to hovering over the keyboard due to the mouse click being signaled by a shift key press. This required the user to use both hands for the interaction.

To our knowledge, Mathias Kölsch, Matthew Turk, and Tobias Höllerer have created the most advanced vision-based gesture interface that has been published [7]. Their project, HandVu, allows them to use hand gestures to emulate mouse-based interaction with a head-worn camera in an augmented reality. The system uses Viola-Jones method, which was initially used for fast and accurate face detection. It also uses Flocks of Features, which provides tracking of the fast-moving and changing shape of the hand, KLT (Kanade, Lucas, and Tomasi) feature tracking, which is based on using steep brightness gradients along at least two directions to track features over time, and color cues for real-time 2D hand tracking. HandVu was originally created for a stationary computer system with a camera looking down on the hand from a top down view. They have advanced their system to a mobile computer that the users may wear on their head. The mobile system uses a stereo cam mounted in the front to capture the hand of the user and react when key gestures are performed with the hands. The concept is reliant on the user looking at their hand so the camera captures it as well. Their system is able to track and recognize certain poses in real time. However, their system is constrained by the need of a head-worn camera.

3. SYSTEM MODEL

3.1 Training Data

In order to train our system, we collected hand tracks to use as training data. We collected a diverse set of hand tracks from 11 individuals. Of these individuals, we used 10 for training and withheld data from one project member’s hands so that unbiased testing could be performed on that project member. In these hand tracks, we selected 6 simple gestures of interest each of which appears in 40 frames in each track. To improve the training of our detector we registered these samples and removed their backgrounds. These preprocessed examples were then used to generate virtual training samples, providing us with as many positive samples as we wanted to train over as seen in Figure 1.
3.2 Initial Hand Detection

Next we used the initial dataset to train an initial hand detector for a simple constrained hand pose. This detector was based on the Viola-Jones method for robust object detection [14]. It used Gentle AdaBoost [11] with Haar-like features and a cascade of classifiers to perform detection in real-time as shown later in Figure 2.

3.3 Hand Tracking

After initial detection, our system takes new frames from the webcam and tracks the hand by calculating the optical flow for a set of interest points. This technique initializes the tracker by finding corner points to follow from the center of the detection which are shown in the tracks of Figure 3. These samples followed by observing the change in their position between consecutive frames.

3.4 Gesture Recognition

Further we trained a system for recognizing gestures as seen in Figure 4. Our system performs recognition using the Eigen Object technique to match the tracked detection against a set of example hands chosen from our all poses of interest. The pose label is chosen from the nearest neighbors based on distance in a subspace of the original image pixels. This method is variant to background noise so the system performs background subtraction of the image as a preprocessing step to avoid being misled.

3.5 Cursor Control

With fingertip registration and pose recognition our system is able to move the cursor in accordance with the movement of a registered pointing fingertip, initiate a right-click, and initiates a left-click. The key poses which initiate the three actions are shown in Figure 5.

3.6 Data Storage

With fingertip registration and pose recognition our system is able to move the cursor in accordance with the movement of a registered pointing fingertip, initiate a right-click, and initiates a left-click. The key poses which initiate the three actions are shown in Figure 5.

4. SYSTEM IMPLEMENTATION

4.1 Database

As part of our project, we have implemented a database to store the data we have collected. This database is designed to allow us to not only store the data, but able to keep track of which tracks were used for what purposes. The information stored in our database includes the preprocessed tracks, generated virtual samples, and pose samples.

As seen in Figure 6, our database contains five tables: PreprocessedHandSet, VirtualTrainingSet, VirtualTestSet, DetectionTests, PoseRecognitionTests. The PreprocessedHandSet table contains registered hands which have been segmented from the background. The VirtualTrainingSet
Table contains the generated samples used to train our detector. The VirtualTestSet table contains generated samples, which have undergone rotation, scaling, and translation, utilized to test our detector. The DetectionTests table contains new tracks which allowed us to test our detection system on actual webcam data. The PoseRecognitionTests table contains processed images of new tracks which have found a hand detection and have segmented the background that were used to test our recognition system.

As tracks are accumulated, they will be processed and stored into the PreprocessedHandSet table. This is the only table it will reside within until the track is processed and used to train and/or test our system. Depending on the how we used the track, we will then add the data to the appropriate table. To automate this process we have written a Java Database Connectivity (JDBC) program, which is an API designed to allow access to a database, that takes information we need and enters the data into the database. The reasoning behind this is to reduce NULL values in the tables, reduce human error, and to have a well organized dataset that future research within this scope of work will be able to use when comparing their results to ours.

4.2 Training Data

In order to extract training data from recorded hand tracks, we first scaled, rotated, and cropped all images to be 72x60 images with the palm of the hand in the lower left section of the image oriented to be upright. This positioning allows the image to contain the fingers above the hand and the thumb beside it when they are fully extended. Registering the training data to be the same size and in the same position improves both our detection and recognition techniques since they are both variant to translation, rotation, and scale. Next we segment the hand from the background using a tool to threshold values on image color channels in both RGB and CIELAB color spaces.

Following this manual preprocessing step of registering and removing the background, we can automatically generate virtual training samples by introducing minor translations, rotations, and scales to a hand and then blending the hand over a random background. This method is used...
to generate the 5000 positive samples for training our initial detector. To generate the 750 examples used in pose recognition, a similar technique is performed, but no random background is introduced.

4.3 Detection System

Implementation of hand detection is performed in OpenCV according to the technique for fast and robust object detection proposed by Paul Viola and Michael Jones [14]. This technique is able to leverage high efficiency through three innovations. First, by using a post processing step to create an integral image, haar-like features can be extracted rapidly in time linear in the number of features. Second the AdaBoost machine learning technique trains classifiers which rapidly classify images using only the best features from the training feature set as seen in Figure 2. Because of this, very few features actually need to be processed at test time. Finally creation of a cascade of increasingly selective classifiers allows later stages of the cascade to have larger processing times. Since many subwindows of an image must be scanned for hand detections, it is beneficial to have early stages quickly trim obvious negatives from the test set before later stages of the cascade are reached.

Our cascade consists of 20 stages which are used to sequentially trim possible candidates. A stage is a classifier composed from a weighted ensemble of decision trees over haar features trained using AdaBoost and each following stage is trained on harder negative example than the previous. Every stage eliminates half of the negative samples. Our system uses the cascade’s detections to initialize our tracking system. The system searches the image with a scanning window of different sizes looking for a closed fist. Detections of closed fist are then sent to the tracking system until a detection which persists for many frames is found. Since there are many possible scanning window positions, this step can be expensive, even with the efficiency of the Viola-Jones classifier cascade. In order to perform at 15 frames per second, the detection system scales down incoming images. This also has the advantage of eliminating small false detections.

4.4 Tracking System

The tracking system is initialized with detections from the detection system. These detections are connected temporally across frames into a track if there is a significantly high ratio between the area of their intersection and the area of their union. When a track is found to contain detections in 75% of 8 consecutive frames, our system begins tracking it and halts the detection system. In order to track a detection, the system uses the Lucas-Kanade Optical Flow algorithm [8] [9] to compute the motion of a set of interest points between consecutive frames. Our algorithm initializes these interest points using a Shi and Tomasi corner detection algorithm [12] shown in Figure 3.

Our system also provides robustness using RANdom SAmple Consensus (RANSAC) [6]. RANSAC repeatedly estimates a projective transformation matrix which projects the interest points to their new observed position. While calculating these transform matrices, RANSAC determines whether points are outliers based on how well they follow the consensus of the interest points. These outliers are then thrown out and our tracking window is shifted to be centered on the mean of the remaining points. Whenever the number of remaining interest points drops below a fixed threshold, our algorithm reinitializes the interest point set.

4.5 Recognition System

The recognition system uses the Eigen Objects method to match poses. First our system is trained with 750 example
hand images, 250 from each pose which it recognizes. These images are placed into a feature matrix with each pixel’s grayscale value as a feature. Since the images are all sized to be 72x60, this amounts to 4320 pixel features. So the resulting feature matrix is 4320x750. Principle Component Analysis (PCA) [13] is then performed on this matrix by converting it into a 4320x4320 matrix holding the covariance between each variable. Then the largest eigenvalues of the matrix are found. Our system uses the top 8 eigenvalues based on experimental testing which found this value to give the best results. Then the eigenvectors corresponding to these eigenvalues are computed and used to project our training examples into an 8 dimensional space. The resulting eigenvectors arranged such that each feature is placed at the corresponding pixel can be seen in Figure 7.

During a recognition task, our system projects a tracked detection into the eigenspace of the top eigenvalues. The distance between the projected tracked detection and each projected training example is then computed and the nearest neighbors to the training sample are found as illustrated Figure 3. The pose chosen for the detection is the pose which receives the most votes from its neighbors. Projecting examples into the eigenspace is beneficial to matching because it drastically reduces the number of dimensions over which the images are compared which makes distance a much more informative measure. It also removes noise resulting from the background behind the hand. In order to further deal with this noise, the background is removed from all example images and our system performs a preprocessing step of background subtraction on our tracked detection. In addition, since this method is variant to changes in scale and position, our system takes a scanning window approach over minor scales and translations of our tracking window and chooses from those windows the result with the closest neighbors.

4.6 Demo Application

To demonstrate the functionality of our system, we created an image viewing application. The application is broken up into four frames as shown in Figure 8. The demo populates a scrollable box full of thumbnail images that the user may select to view in the larger frame. Once the user has selected an image to view, the larger image view box displays the selected image. We have also provided a camera feed on the upper right to allow the users to see what the camera is receiving and the resulting detection, tracking, and recognition from our system. Finally the last portion of our demo is instructions on how to initialize hand tracking and what poses for navigating and interacting with the application.

5. RESULTS

For our experiments, we trained over hand samples in the closed fist pose from 10 of our 11 individuals, reserving samples from one individual as holdout for fitting parameters. For our detection cascade, we experimented with the number of stages in the cascade and found 20 to be an effective number of stages. Since each stage was trained with the goal of at least 0.997 recall and at most a 0.5 false alarm rate, we expected our final cascade to perform at $0.997^{20} = 0.942$ recall and $0.5^{20} = 9.54 \times 10^{-11}$ false alarm rate on the training samples. This expected performance is compared with the actual results in Table 1. The recall is where expected, but the false alarm rate is about 2.5 times what is anticipated. This result is likely caused by a classifier overfitting to the negative samples it is trained over and not being as effective on random samples. In webcam driven experiments, this detector is able to perform consistently at 15 frames per
To test our detector on a less synthetic data, we record video of our holdout individual’s hand in the closed fist pose. We have two videos, one in which the individual moves his hand around in an upright position and a second in which in addition to moving his hand, the individual varies his hand’s pose, rotates it, and alters its distance from the camera. The results of this experiment are shown in Table 2. The precision remains high for both videos, suggesting the detector would benefit very little from additional stages and the overfitting to negative samples is a minor issue. More interestingly, the recall drops drastically for the difficult test frames. This outcome suggests the detector suffers significantly from hand variation in both size and orientation as well as changes in pose. Some approaches to remedy this might be to vary rotation and scale in our virtual training samples, but this also raises the risk of the detector not learning effectively. It could result in higher false alarm rates and a slower detector. Another approach would be to run the detector over the image under a couple rotations. Although this would be very effective at increasing our recall on rotated hands, it would drastically reduce our performance to below 10 frames per second.

For tracking and recognition, our system performs at 14 frames per second as shown in Table 4. Although this is not quiet our goal of 15 frames per second, it is our opinion it is close enough and the cost of increasing it further is not worth the speed up. Our tracking system performs well under very controlled circumstances, but some unfortunate errors can often lead to catastrophic failure. Since we are tracking interest points using optical flow, objects occluding our tracked hand will often completely dislodge our track points from where they belong. This behavior is difficult to recover from even though we have a mechanism for reinitializing tracking points because reinitialization often occurs at the wrong location. We are trying to implement a method for falling back on the detector to reinitialize the hands position, but it is still in development. Although this is our worst case of failure, it does not occur too often because it is rare for something to pass between the user’s hand and the camera.

Similar failures can occur when a hand moves too fast or the camera lags in producing up-to-date images. The tracking points can be left behind after a sudden shift in the hand’s location. These errors are often partially recovered from when the interest points are reinitialized. Still these pose a larger threat to our application than occlusions because at least one such event is likely to occur in a five minute session of use. Also after partially recovering from these events, our tracker often drifts because the corners on the edges of the hand are not as stable as the corners in the center and points are rapidly lost resulting in frequent reinitializations. Our project had hoped to avoid such drifting by using a MILBoost tracker [1] which is based on updating
a boosted classifier online. Unfortunately our experiments with the papers distributed code were only able to achieve tracking at 5 frames per second, a wide difference from the 25 frames per second reported in the paper. Although our setup only uses Dual-Core laptop while theirs involves a Quad-Core desktop, we are unsure of how to account for the 5x slowdown.

For recognition using Eigen Objects, our experiments involved tests using our preprocessed tracks, with hand position registered and background removed. In these experiments our system misses only 11 out of 750 test cases see Table 3. Beyond that in these results, all of our missed recognitions were cases of mistaking a pointing finger for a pointing finger with the thumb extended. These results show a lot of promise for ideal cases. Unfortunately we are still adjusting what step size would be good to perform a local scanning window around a track to achieve good recognition without sacrificing runtime. Also since our background subtraction is imperfect, background noise can be an issue, especially when the hand is over a similarly hued background area such as the user’s face. Additionally, when we allow the recognition system to vary the scale of the detection, drift can occur during which the window size becomes really large or really small. We are still the cause of this and hope to have a solid recognition system working soon.

### 6. FUTURE WORK

There are still many improvements that can be made to our system. The current detection system is variant to rotation and scale changes and requires a set pose to allow for the initially detect the hand. Training with more variation in the training set could improve results, but if too much variation is introduced it will become difficult to keep good precision. Precision can always be increased at the cost of recall by adding more stages, but each successive stage takes twice as much time to find harder negative samples and the gains diminish as the number of stages increase. One very interesting approach to improving the detector would be to experiment with a tree based arrangement of stages rather than just the sequential model. With this different poses and different orientations could be expressed with different branches of the cascade. This way the advantages of having a separate detectors for different poses could be obtained without the runtime costs of running multiple detectors.

The tracking system is sensitive to camera lag and rapid motion and vulnerable to occlusion. Although our system includes some mechanisms for recovering from tracking errors, there is still a good amount of tweaking necessary to get our tracking to recover robustly. Create a more complex state machine so that our system falls back to reinitialize points when that is effective and rerunning our hand detector in other cases would be very advantageous.

The recognition system is sensitive to background noise and lighting changes and has a fixed set of poses that it is able to recognize. Change in translation, scale, and rotation also hinder the accuracy of our recognition. There is still a significant need to tweak our scanning window method for finding nearest neighbor. Our recognition system would also benefit from a state machine for ensuring persistence of gestures. This way misclassifications in isolated frames would not have an effect on our system. Although we collected tracks for 6 unique poses, we only ended up using 3 of these poses. Extending our recognition system to work on a broader range of poses is another plan for our system’s future development.

Collecting many more tracks with more diversity of hands will always be beneficial. Now that we have a good initial hand detection and tracking system, we could deploy a system for recording tracks of volunteers and automatically classifying them. Thus we could generate many additional examples which would improve both our detector and our recognition system. Such a system might also allow users to automatically label their hand poses in different video segments or to register and segment their hands from the background, eliminating our need for processing.

Improved methods of hand segmentation and background subtraction would lead to better recognition. Also more complex features can be used for recognition. Some examples are more complex interest point features such as Speeded Up Robust Features [2] or better shape context features such as Histogram of Oriented Gradient descriptors [5]. These features would allow for improved recognition and perhaps could also be used for improved tracking. They can be used in Eigen Object techniques in place of pixel intensity values or they can be used to train classifiers in place of Haar features. Although these more complex features are more expensive to compute and thus not as many features can be used, they are often far more informative than the simple features we use.

The final step of future work would be development of real

### Table 1: Training Results

<table>
<thead>
<tr>
<th>Training Metrics</th>
<th>Expected</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.942</td>
<td>0.944</td>
</tr>
<tr>
<td>False Alarms</td>
<td>$9.54 \times 10^{-7}$</td>
<td>$2.53 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

### Table 2: Detection Results

<table>
<thead>
<tr>
<th>Detection</th>
<th>Easy Tests</th>
<th>Difficult Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.94</td>
<td>0.52</td>
</tr>
<tr>
<td>Precision</td>
<td>0.97</td>
<td>0.93</td>
</tr>
</tbody>
</table>

### Table 3: Pose Recognition Results

<table>
<thead>
<tr>
<th>Pose Recognition</th>
<th>Perfect Window and Background Subtraction</th>
<th>Actual Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.985</td>
<td>???</td>
</tr>
</tbody>
</table>

### Table 4: Performance

<table>
<thead>
<tr>
<th></th>
<th>Frames Per Second (FPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection</td>
<td>15</td>
</tr>
<tr>
<td>Tracking &amp; Recognition</td>
<td>14</td>
</tr>
</tbody>
</table>
applications which benefit from this technology. We present an image viewing application as an example of where this technology could lead to a more natural user interface. The same could be said for navigating something like Google Maps or browsing folders on a screen. But the applications reach far beyond that. They are particularly compelling in situations where touchscreens are not applicable or less than ideal. For example, with projection systems there is no screen to touch. Here vision-based technology would provide an ideal replacement for touchscreen technology. Similarly in public terminals, constant use results in the spread of dirt and germs. Vision-based systems would remove the need to touch such setups, and would result in improved interaction.

7. CONCLUSION
The goals of this project included:

- Database of labeled training samples.
- Robust hand tracking.
- Gesture recognition for a couple simple gestures.
- Cursor movement and button commands based on hand tracking and gesture recognition.
- Robustness for in cursor control for good usability.

We have met the majority of our goals and provided a means for unconstrained human-computer interaction. The ultimate motivation was to eliminate the need to touch peripherals. We eliminated the necessity to touch a mouse by allowing hand gestures to emulate mouse function. In doing so, we have succeeded in eliminating many of the general peripherals that common users touch when interacting with a computer. The portion that is left is the keyboard which could be removed as well given more pose recognitions. Even though our system is not as robust as we would like, we see it as a large step in the right direction. What sets us apart for previous works is the freedom from wearing gloves or markers to interact with our system and alleviating the need for top down view and static backgrounds.

During the course of building this system, we found that collecting a diverse set of handsets was one of the most time consuming and difficult tasks. In an attempt to alleviate the need for future works in this scope to have to do the same, we have packaged all our samples, tracks, and other relevant data in hopes that future research done will have a handset to work with from the start.

8. REFERENCES