Real-time, Static and Dynamic Hand Gesture Recognition for Human-Computer Interaction

S.M. Hassan Ahmed[a], Todd C. Alexander[b],
and Georgios C. Anagnostopoulos[b]

[a]Electrical Engineering, University of Miami, Miami, FL;
[b]Electrical & Computer Engineering, Florida Insitute of Technology , Melbourne, FL;

ABSTRACT

Real-time, static and dynamic hand gesture recognition affords users the ability to interact with computers in more natural and intuitive ways. The hand can be used to communicate much more information by itself compared to computer mice, joysticks, etc. allowing a greater number of possibilities for computer interaction. The purpose of this work was to design, develop and study a practical framework for real-time gesture recognition that can be used in a variety of human-computer interaction applications with the aim of developing a prototype system for controlling Microsoft PowerPoint$^{\text{TM}}$presentations. The approach taken is to locate hands starting with the finger tips after investigating potential regions of interest that are maintained through motion detection and subsequent tracking. The fingertips are very distinguishable from many backgrounds allowing a very robust tracking system. Using techniques such as Features from Accelerated Segment Test (FAST) corner detection the tips can be found efficiently. Circles generated using Bresenham's algorithm were employed for finding corners as well as the center of the palm. Using features generated from these results along with a semi-supervised Adaptive Resonance Theory (ART) neural network, static gestures were able to be classified with an overall accuracy of 75%. Dynamic gestures on the other hand were able to be detected using the trajectory formed by the center of the hand over a finite amount of time. Simple decision heuristics were then utilized to detect the movement of the hand. This research produced a working prototype of a robust hand tracking and gesture recognition system that can be used in numerous applications.

Keywords: ART neural networks, computer vision, FAST corner detection, gesture recognition, hand tracking, human-computer interaction, human-computer interface, motion detection, real-time gesture classification

1. INTRODUCTION

The way humans interact with computers is constantly evolving, with the general purpose being to increase the efficiency and effectiveness by which interactive tasks are completed. Real-time, static and dynamic hand gesture recognition affords users the ability to interact with computers in more natural and intuitive ways. The research presented in this paper is geared towards bringing together static and dynamic gesture recognition approaches, in order to improve the user experience, when interacting with computers through gestures. The main focus of the research was on bare-hands[*] utilizing a simple web camera with a frame rate of approximately 30 frames per second.

Mid-century brought with it a new outlook on science and technology, and human-computer interaction was no exception to this. As early as 1963 the first device to use gestures existed, that is the Sketchpad light-pen system. 1964 saw the first trainable gesture recognizer from Teitelman.[1] Since this period many successful work has been done in this field from sign language recognition[2–4] to object interaction.[5, 6]

Gestures are a form of non-verbal communication used by people on a daily basis. Similarly using these particular postures or motion of the hand can be used to communicate with machines. Hand postures or *static*

Further author information: (Send correspondence to G.C.A.)
G.C.A.: E-mail: georgio@fit.edu, Telephone: 1 321 674 7125
S.M.H.A.: E-mail: h.ahmed@umiami.edu
T.C.A.: E-mail: talexand@fit.edu

[*]Gloves or detection aiding devices were not used

gestures are gestures that do not depend on motion, as is the case with Malassiotis'[7] and Strintzis' work. *Dynamic gestures* on the other hand require motion and are based on the trajectory that is formed during the motion in question.[5, 6]

Hand gesture recognition plays an important role in the following areas:

- touchless interaction with objects[5, 6]
- sign-language recognition[2–4]
- controlling computers[8, 9]
- behavior understanding[10, 11]

Bare-hand gesture recognition presents many challenges to researchers. Over the years various approaches to overcome these challenges have been published; however, several drawbacks arise due to inherent assumptions in many of these approaches.

Graph matching: an approach presented by Miners et al.[12] has been successful in understanding the meaning of three to five ambiguous hand gestures. Their goal was to obtain a greater amount of understanding from gestures rather than simply recognizing them. This was not a bare-hand system, however it provided useful insight into ways to get dynamic gestures.

Monte Carlo tracking: to address the issue of many possible finger articulations, Wu et al.[13] used the Monte Carlo tracking approach in order to maintain translation and rotation invariance which are common issues in hand tracking.

Infrared camera: an excellent way of dealing with dynamic backgrounds and low contrast environments is to work in a different part of the electromagnetic (EM) spectrum as was done by Oka et al.[14] Their use of an infrared camera made it possible to easily segment a human hand based on human body temperatures. This approach is very reliable for tracking; however, expensive hardware is needed.

The motivation for the work in this paper is to utilize very low-end hardware to perform robust tracking of the hand, and classification of static and dynamic gestures. In addition, making the system easy to use on many platforms was taken into consideration, hence personal computers were used to perform the data processing. In order to minimize the amount of the computer's resources used by this system for computations, very efficient algorithms were selected for performing tasks such as finger tip detection, palm detection and gesture classification. The overall goal of this research is to be able to recognize static and dynamic gestures performed by a human hand in real time. This involves working in the areas of computer vision, with special emphasis on image processing and machine learning.

Initial test results for classifying static gestures show a classification rate of 75%. This may be less than perfect but is only from preliminary testing. It is fair to expect improvements to the features extracted along with adjustments to the classifiers parameters to improve the classification rate. The approach taken for hand detection yielded excellent results, with the hand being detected 99% [†] of the time.

In Section 2 an overview of the research performed is presented. The subsequent sections—5, 6, and, 7 —include an in-depth investigation of the various components. Finally, Section 8 looks at the results from experimental data followed by any conclusions that can be drawn in Section 9.

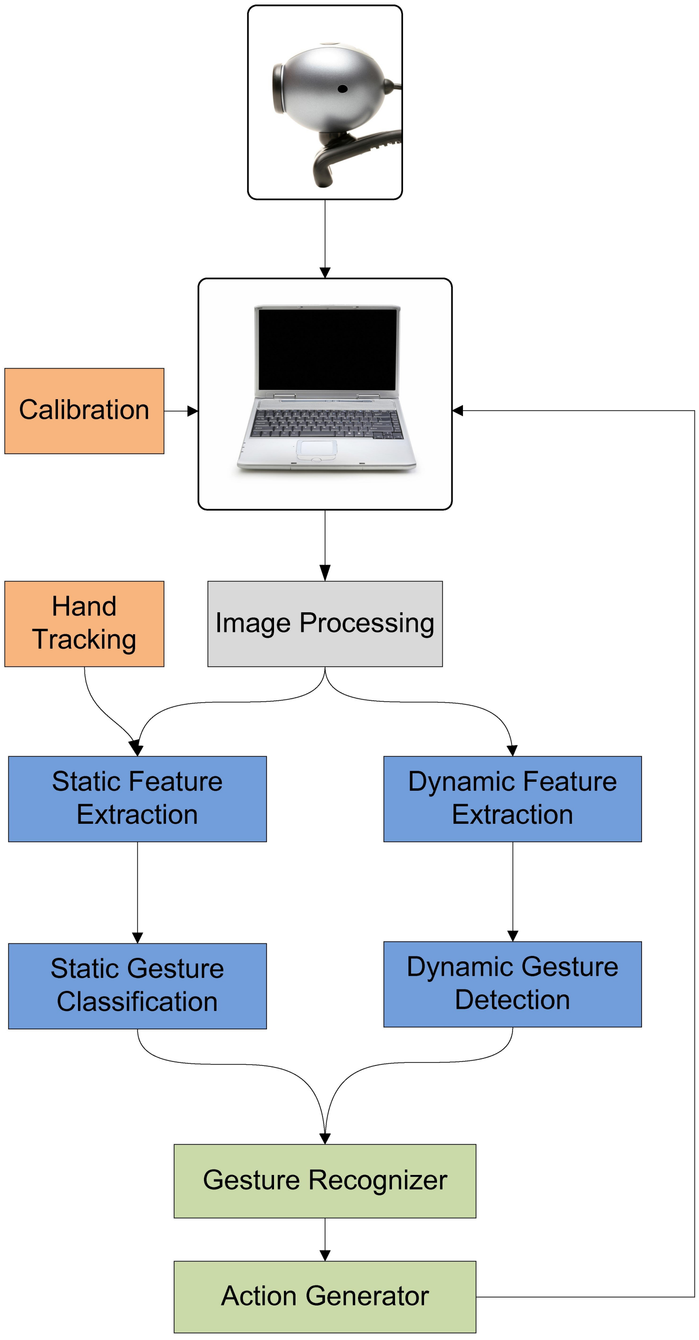
Calibration
Hand Tracking
Image Processing
Static Feature Extraction
Dynamic Feature Extraction
Static Gesture Classification
Dynamic Gesture Detection
Gesture Recognizer
Action Generator

Figure 1. Overview of the proposed system's various components. The process begins with the computer system capturing a frame from the web camera, which is then processed to detect candidate regions of interest. After a hand is found in a region of interest it is tracked by the *Hand Tracker*—all subsequent processing is done on this region. While being tracked static and dynamic features are collected, with the static features being passed to an ARTMAP Neural Network classifier and the dynamic features analyzed to obtain high level features. The static and dynamic information is then passed to the *Gesture Recognizer*, which communicates the type of gesture to the *Action Generator*. The action generator passes the necessary commands to the operating system to create the end result.

2. SYSTEM OVERVIEW

Fig.1 illustrates the major components present in the recognition system. The two fundamental hardware components are the web camera and the computer system that it is connected to. They are both connected using a media—Universal Serial Bus (USB), Firewire (IEEE1394), Coxaxial, etc.—then it is up to the computer system to perform frame grabbing.

Frame grabbing in this case was done through an application programming interface (API) in order to abstract frame capture as much as possible while allowing the software to run on any computer using a particular operating system. The original implementation was based on the Microsoft Windows$^{\text{TM}}$operating system written in the C# programming language.[15] This area of focus warranted the use of a popular open source image processing library, AForge.NET.[16] This library and its underlying technology, Microsoft DirectShow$^{\text{TM}}$,[17] are discussed in greater detail in Section 7.

Initially, a reference or *key frame* is set after a determined amount of time from startup. This frame is set a few seconds after the web camera is activated in order to allow ample time for hardware initialization. At the beginning of runtime this frame also serves as the previous frame, which is then used to find the difference with subsequent frames, aiding in the process of motion detection. This is all taken care of in the processing component.

After processing is complete candidate *regions of interest* (ROI) are investigated to detect finger tips. If characteristics of a human hand are found in such a region, this region is considered to be containing a hand, and is processed as a sub-image utilizing the key frame. Once the hand has been found features for static gestures are extracted from individual frames. Concurrently, by tracking the hand's motion, the dynamic features of the gesture is established.

This system can be used to recognize static and dynamic gestures, resulting in a plethora of interaction possibilities; however, the research presented here focuses on applying real-time hand gesture recognition to presentation control as a proof of concept.

The description given for the system in this section hinted at a few restrictions that users would have to abide by in order for tracking, detection and recognition to all work robustly. They are as follows:

- users and their surrounding environment should be approximately static (non-moving) prior to performing a gesture
- the color and intensity characteristics of the hand's region should be distinguishable from the background

These restrictions are reasonable and do not hinder users so much as to make the chosen application impractical. Firstly, keeping the environment static prior to performing a gesture affords the system the ability to know when to update the key frame. Secondly, if the hand is camouflaged via its surrounding background, then it will be a challenge for a human to distinguish, much less a machine.

3. CALIBRATION

In order for tracking and feature extraction to be done robustly and efficiently, the system needs to know certain information about what to look for in the object(s) that needs to be tracked. It is possible to determine this information by taking an unsupervised learning approach; however, this can be computationally expensive in a real-time environment. Due to the fact that the tracked object in this case is the hand it can quickly be calibrated prior to using the system. Biologically, a human hand motion must obey biomechanical and neuromuscular constrains.[18]

In Section 5.1 finger tip detection using FAST corner detection is discussed in detail. However, before fingers can be accurately detected, suitable parameters have to be obtained. As it will explained further ahead, finger detection entails choosing an appropriate circle radius for locating potential finger tips. For that purpose, during calibration a suitable radius is obtained via a supervised process, whereby users place their hands inside of a preset location in the scene. Then, knowing the number of fingers that should be present, the system iteratively tries a number of radii, until the best results are obtained.

[†]Valid provided that the limitations are respected

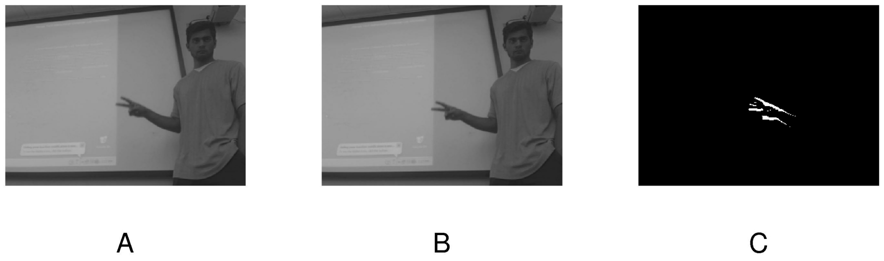

Figure 2. Example of a previous, current, and binarized difference of the previous and current frames in A, B and C, respectively. Regions of high intensity in C imply objects of fastest motion. The image in C makes is possible to discriminate a moving hand, while the rest of the background is virtually motionless.

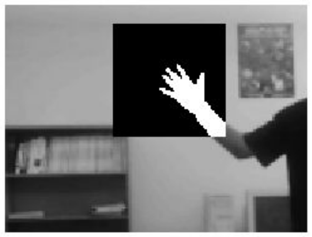

Figure 3. A ROI containing a hand. this initial ROI was found from motion detection. Then the image was differenced with the reference frame in that region and binarized, giving a segmented hand image. The size of the ROI is preset during calibration

4. IMAGE PROCESSING

4.1 Motion Detection for Potential Region of Interest

Motion Detection involves finding the areas of motion in a given scene. An easy way of detecting motion in a scene is through differencing consecutive images from a stream of images from a particular scene.

In this research motion detection was used for locating a potential ROI. It is generally assumed that there would be little motion in the background and that the associate body would not be moving. This is a reasonable assumption, when giving a command to a computer.[5,6]

The first module for our system involves motion detection for the location of the hand. First, it is assumed that the input stream consists of grayscale images. The proposed method calculates the pixel-wise, absolute difference in intensity between successive frames. This will result in a grayscale image, where the regions of motion will be lighter, while the static regions will be darker, since the difference between pixel values (see Algorithm 1 would be very close to zero; see Fig.2 for an example.

This grayscale image is then converted to a binarized image by examining which pixels are above and below a certain threshold. The binarized image is much easier to work with and process. It is also desirable to perform morphological erosion[19] on the binary image, since it removes noise and slight motion of the body.

After performing these steps, there may still be regions, where slight undesirable body motion shows up in the image as white pixels. A quick technique is required that would differentiate between the hand motion and other motion such as the one of an elbow or forearm. Since the hand is an appendix, it is assumed that its motion will have the greatest linear velocity and, hence, the densest amount of white pixels in the differenced image. A quick search for the densest region in the differenced image is used to pinpoint the potential hand region. The ROI is then established with center $\mathbf{c}$ as the center of the densest region with a width R_w and height R_h. The values of the R_w and R_h depend on the distance of the hand from the the camera. The values for R_w and R_h are obtained via system calibration (see Section 3). Any subsequent processing done by the system would only be performed in this region as opposed to the entire frame. See Fig.3 for an example of an initial region of interest.

```
Input:
Current Frame: $F$
Reference Frame: $R$
Threshold: $\theta$
Output:
Binarized frame difference $D$
1 $D = |F - R|$ ;
2 foreach pixel $p$ of $D$ do
3     if $D(p) > \theta$ then
4         $D(p) \longleftarrow 1$ ;
5     end
6     else
7         $D(p) \longleftarrow 0$
8     end
9 end
```

Algorithm 1: Simple procedure for motion detection. This grayscale image is then converted to a binarized image by examining which pixels are above and below a certain threshold. The binarized image can be processed more efficiently.

4.2 Region of Interest Tracking

ROI tracking is the next step to tracking the hand and extracting features for gesture classification, which will be performed later. Initially, motion detection (described in Section 4.1) was performed to get a potential regions of interest. In general, these candidate regions are not guaranteed to contain the hand.

After motion has been detected and some basic criteria are met, the current frame from the stream is differenced with the reference frame—see Fig.4. The reference frame is taken during system initialization and is then updated based on activity rules. The difference image is grayscale by default; however, it is then converted to a binary image as in Fig.4 using another threshold θ_{ref}, such that we now have a completely segmented hand within the initial ROI.

Next, fingers are detected within the initial ROI, that was initialized by the motion detection module discussed in Section 4.1. The presence of fingers will definitely guarantee that the hand is present in the ROI. Detection of fingertips is performed using an adaption of the *FAST algorithm*[20] and is discussed in Section 5.1.

When fingertips are found, the mean of the x coordinates of the fingertips, μ_x, and the mean μ_y of the y coordinate of the fingertips are calculated. The point (μ_x, μ_y) will be the new center for the region of Interest. If no fingertips were detected, the process reverts again to detecting motion and identifying a new initial ROI as described in Section 4.1. Also, the number of fingertips N_f is stored as a feature to be used to classify a potential gesture, which will be discussed in more detail in Section 5.

5. FEATURE EXTRACTION

In order to determine the gesture being performed by the user in real-time, quick extraction of separable features is desired. These features need to fulfill certain qualities like being rotationally-, scale-, translation-, and, even, reflection-invariant. This allows for robust gesture recognition as well as for both left- and right-handed system use.

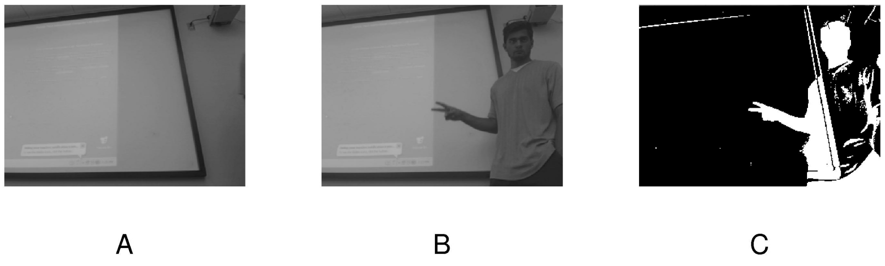

Figure 4. Reference/key frame, A, frame containing presenter's hand, B, and the thresholded difference of A and B. The human subject's body cna be distinguished in the figure, as it represents a new object in the scene.

Some of features that are used in hand gesture recognition are the number of fingers present, distribution of fingers with respect to the center, angle between the arm direction and the palm center and the centroid of the fingers, and other possible features derived from curvature information or skeletonization information.[21]

The features used in this work include: the number of fingers present; the angle between the vertical direction and the arm's orientation; and, the palm center and the centroid of the fingers present. These features are not very computationally expensive as compared to other features and they satisfy all the qualities of invariance mentioned above.

The following subsections describe in more detail how these features are extracted. Pseudo-code is also provided.

5.1 Fingertip Detection

An essential step in our system is correctly determining how many fingers are being shown by the user and their relative locations. We use a fast corner detection method on our binary hand image to determine where the fingers are. This method, also known as the Features from Accelerated Segment Test (FAST)[20] can give us real-time performance.

The concept behind this algorithm is traversing all the white pixels and at each of those pixels, examining the binary value of the points that lie on a circle of a certain radius r_{finger} around it, see Fig.5. The circle is traversed and the maximum amount of black (non-hand) pixel points on that circle N_b is calculated. Due to the specific geometry of the fingers, the value of N_b should be greater than a certain minimum threshold N_{min}; see Fig.5. The circles are generated using the Bressenham circle-drawing algorithm,[22] which is a very efficient way of rasterizing circles. Those pixels on the hand that satisfy the ratio are potential fingertip points.

Since it is possible that several pixels on the same fingertip may match the criterion, it is checked whether a fingertip point is too close to another fingertip point. The L_1 norm is computed between the pixel in question versus pixels that are already considered part of the tip. If the L_1 norm is too small then this means the pixel in question is really redundant and needs not to be added to the list of fingertip points. See Algorithm 2 for an outline of the FAST algorithm.

The benefits of this method is that it is also rotational-, translatoinal-, and reflection- invariant given the search radius. This implies that the hand can be in any orientation in front of the camera and the algorithm will still be able to detect fingertips.

5.2 Palm Center Location

Locating the palm center is another essential step in the feature extraction process. A similar idea for identifying the palm center is used as was used for identifying the fingertips. Essentially, the ROI is processed to fit the largest circle within the hand. From rough calculations, the radius of the palm r_{palm} was found to be about 1.66 times as much as the radius r_{finger} used for discovering the fingertips. All the hand pixels are iterated through and a circle of radius twice as much as the radius used for detecting fingers is fitted. If the circle fits within the the white region, then the point is added to the list of possible center points. Since the width of the

```
Input:
Binarized Image: $B$
finger radius: $r_{finger}$
Fingertip threshold ratio: $N_{min}$
Fingertip closeness parameter: $dmax$
Output: List of pixels denoting detected finger tips $A$
1  foreach $pixel\ p|B(p) = 1$ do
       /* getCircle will return the x-coordinate and y-coordinate of all the points on a
       circle using the Bressenham algorithm with center $p$ and radius $r_{finger}$ to a data
       structure $circle$ that encapsulates that information                              */
2      $circle \longleftarrow$ getCircle($r_{finger}$, $p$);
       /* returns the max amount of continous black pixels on the circle                  */
3      $N_b \longleftarrow$ findMaxContiguous($circle$) ;
4      if $N_b > N_{min}$ then                                  /* potential finger tip point */
5          $isFingerTip \longleftarrow TRUE$ ;
6          foreach $q \subset A$ do
7              $dist \longleftarrow |q - p|_1$;
8              if $dist < d_{max}$ then  /* checks if too close a point that is already a fingertip
               */
9                  $isFingerTip \longleftarrow FALSE$ ;
10             end
11         end
12     end
13     if $isFingertip = TRUE$ then
14         $A \longleftarrow \{A; p\}$ ;
15     end
16 end
```

Algorithm 2: FAST algorithm for detecting fingertips: The concept behind this algorithm is traversing all the white pixels and at each of those pixels, examining the binary value of the points that lie on a circle of a certain radius r_{finger} around it. The circle is traversed and the maximum amount of black (non-hand) pixel points on that circle N_b is calculated. Due to the specific geometry of the fingers, the value of N_b should be greater than a certain minimum threshold N_{min}

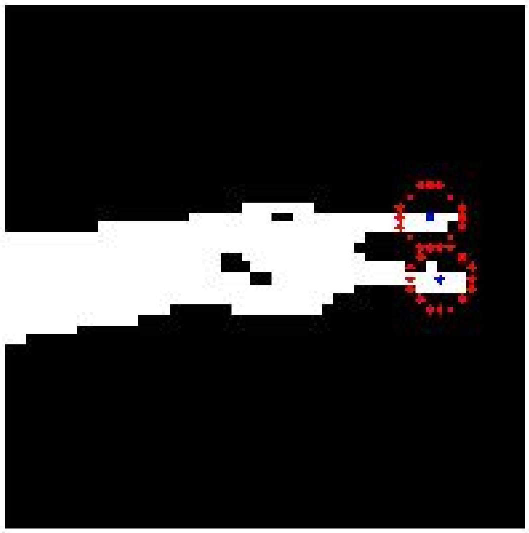

Figure 5. Fingertip detection using FAST

forearm increases near the elbow, there is a possibility that circles of radius r_{palm} may fit in those locations. An additional check can be performed, where the L_1 distance from a fingertip is calculated. If a pixel in question is too far away from the fingertip pixels, it cannot be the palm center.

5.3 Arm Orientation Extraction

In order to extract the second feature, which is the angle between the arm orientation and the vector from the center of the palm and the centroid of the fingertip (see Fig.6), the arm direction must be calculated first. This is a rather trivial task, but the assumption must be made that there is no wrist movement and the hand is located away from the body, i.e. the wrist is in line with the rest of the arm as shown in Fig.3. The angle is calculated by traversing the boundary of the region of interest and finding the mean of the white pixels that are on the boundary. The vector $\mathbf{v}_1$ constructed from the center of the hand and that boundary point is the vector describing the hand direction. The vector $\mathbf{v}_2$ is then constructed using the center of the palm and the centroid of the fingertip points. The angle between these 2 vectors is used as another feature for the static gesture classification stage.

The advantage of such a feature is that it satisfies all the invariance qualities mentioned in Section 5. On the other hand, a disadvantage to this feature is that it becomes less informative for classification, when wrist movement is not precluded.

5.4 Hand Motion

The basis of dynamic gesture recognition is understanding the movement that a particular object experienced over a certain interval of time. One of the static features extracted is the center of the palm—section 5.2—and hence this acts as the reference point for keeping track of the hand's trajectory.

One of the limitations of the system is that before performing a gesture the user must remain still. This acts as an indicator to the system to update the reference frame, any subsequent movement will be seen as a potential start of a gesture, if a hand is detected its center is kept track of in a vector of points—x and y coordinates.

Identifying the end of the gesture is another issue because there needs to be a reliable indicator that tells the system that a gesture has been completed. In this project completing a gesture in a fixed period of time

```
Input:
palm radius: $r_{palm} = 1.66 * r_{finger}$
maximum distance the center of palm can be away from the fingers: $D_{max}$
binarized segmented hand image: $I$
fingertip points: $A$
Output:
Palm Center pixel: $\mu$
Data:
List of pixels that are close to palm center: P
Number of pixels that constitute circle of radius $r_{palm}$: $N_{circle}$
foreach pixel $p|I(p) = 1$ do
    /* getCircle will return the x-coordinate and y-coordinate of all the points on a
    circle using the Bressenham algorithm with center $p$ and radius $r_{palm}$ to a data
    structure circle that encapsulates that information                                  */
    $circle \longleftarrow$ getCircle($r_{palm}, p$);
    foreach pixel $q \subset circle$ do
        if $I(q) = 1$ then
            increment $whiteCounter$ ;
        end
    end
    if
    $whiteCounter = N_{circle}$
    then                                    /* if all surrounding points are white */
        $dist \longleftarrow \max |p - A|_1$ ;
        if $dist < D_{max}$ then        /* if point is within reasonable distance to fingers */
            $P = \{P; p\}$ ;
        end
    end
end
$\mu = mean(P)$ ;
```

Algorithm 3: Detecting the center of the palm: All the hand pixels are iterated through and a circle of radius twice as much as the radius used for detecting fingers is fitted. If the circle fits within the the white region, then the point is added to the list of possible center points. An additional check is performed, where the L_1 distance from a fingertip is calculated. If a pixel in question is too far away from the fingertip pixels, it cannot be the palm center.

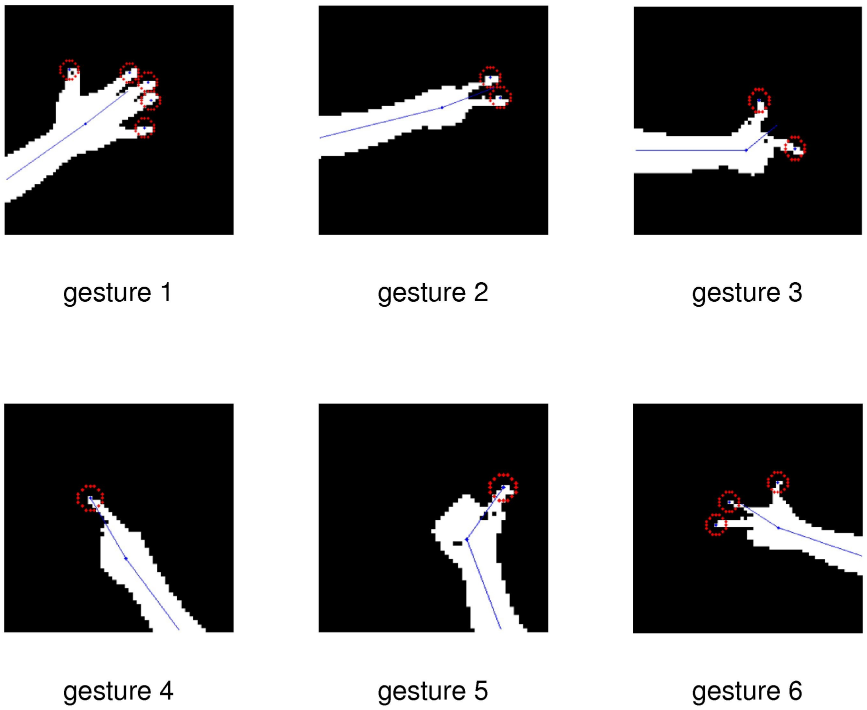

Figure 6. Ideal feature extraction on the initial six gestures: This figure shows a visual depiction of several features that are being extracted. It shows the finger tips, the palm center, the orientation of the arm, and the vector between the center and the centroid of the fingertips.

Gesture	No. of fingers	Angle (degrees)
1	5	2-10
2	2	5-15
3	2	25-35
4	1	3-10
5	1	55-60
6	3	5-15

Table 1. Ideal features for each of the six gestures. Please refer to Fig.6 for a depiction of the individual gestures

was deemed reasonable and hence 1.5 seconds was chosen. After this period of time the trajectory information stored in the history vector is examined to determine what type and or direction the hand moved.

6. CLASSIFICATION

After static features are extracted the next module of the system consists of correctly classifying the presented pattern to the appropriate gesture. This classification process must be very robust, adaptive and must allow for real-time performance. For these and a couple more reasons the *semi-supervised Fuzzy ARTMAP* (ssFAM)[23] classifier was implemented for the static gesture classification stage.

ssFAM is a neural network architecture, which is built upon Grossberg's *Adaptive Resonance Theory* (ART).[24] It has the ability to learn associations between a set of input patterns and their respective class/label, which then can be utilized for classification purposes. The ssFAM network clusters training patterns into categories, whose geometrical representations are hyper-rectangles embedded in the feature space. Each time a test pattern is presented a similarity measure is computed to determine, which of the discovered categories it belongs too. Via this mechanism, it also has the ability to discover, if completely new information is being presented to the network. This is the case, when a given test pattern does not fit the characteristics of any of the categories that the system has learned so far. This mechanism is useful in our application so that irrelevant or ambiguous gestures are appropriately ignored.

An additional, important quality of ssFAM (and of ART-based classifiers) is the fact that the network is able to learn incrementally and, therefore, is ideal for online learning tasks. In our framework of gesture recognition this is a highly desirable feature, as it will allow our system to add new gestures or to accumulate additional knowledge about already learned gestures in an online fashion.

Finally, the fact that ssFAM is capable of organizing its acquired knowledge by forming categories in a semi-supervised manner,[25] will allow the system to generalize well when attempting to learn gestures.

7. SOFTWARE DESIGN AND ORIENTATION

The majority of the research performed involved implementing and testing algorithms, detectors and classifiers in software. To facilitate future development all software applications were developed for Microsoft's Windows^{TM}operating system running the .NET framework. The language of choice for development was C#[15] for its growing popularity and readability.

7.1 Design

The starting point of the code is based on the AForge.NET package—it provides support for a variety of Machine Learning and Image Processing methods. However, it is an evolving project; thus, only a few aspects of the areas mentioned are covered. AForge.NET was used because it is open source, hence the results of this research can potentially be contributed towards the project. For the research done, the main area of interest was in the AForge.Video namespace, which provided methods and event handlers for capturing frames from any Windows®certified camera.

The Unified Modeling Language (UML) class diagram shown in Fig.7 shows how the AForge package is used along with the two main classes developed—*HandTracker* and *Hand*.

Objects of type *HandTracker* are composed of *Hand* objects. For the presentation control application only one hand is required; however, the class is written so that this property can be set for future development involving multi-hand tracking. A *HandTracker* is able to find candidate regions of interest, investigate these regions then (if data conforming to a hand is found) extract all the necessary features, which are stored in a *Hand* object. *Hand* objects also have methods for obtaining high-level dynamic features, which are in turn used to determine the hand's motion. In addition, *Hand*s are composed of one or more *Classifier* objects, which are used for classifying static gestures.

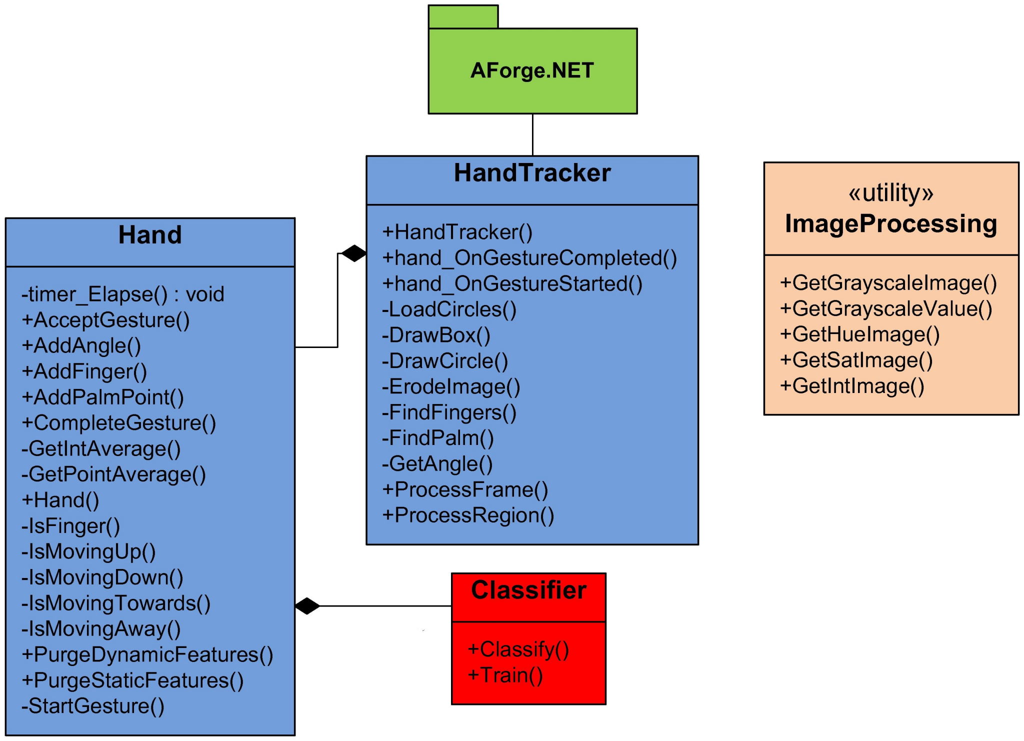

Figure 7. Design of hand tracking software. The AForge.NET package was used as the starting point for capturing frames from any Windows Certified web camera. The *HandTracker*, *Hand*, and *Classifier* classes were then developed to facilitate tracking, feature extraction and classification.

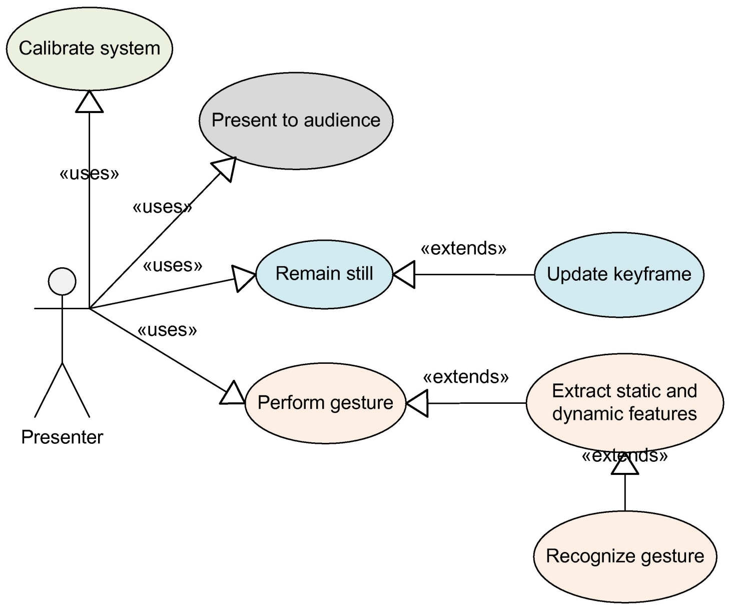

Figure 8. UML use case diagram outlining the various operations allowed by the developed system. Users are allowed to calibrate, present to an audience, remain still or performed one of the trained gestures.

7.2 Use Cases

As illustrated in Fig.8, there are four actions that users can be performing at any given point in time:

1. calibrating the system
2. presenting to an audience (random hand and body gestures)
3. remaining still (no moving objects)
4. performing one of the trained gestures

The four main actions also hint at the various states the system is allowed to be in. For example, during calibration users are expected to place their hand infront of the camera in order for necessary information to be collected. Likewise random motion will be ignored, but once the user remains still the system will update the reference frame and stand by waiting for a gesture to be performed.

7.3 Open Source Contribution

One of the objectives of this research was to provide a robust library for hand tracking and, static as well as dynamic hand gesture recognition. There is room for improvement in the current version and by making the library open source many more researchers in the field will be able to contribute to the project.

It is possible for our work to be added under the machine learning namespace in AForge, but it can also just as easily be released as an independent project. Projects such as AForge.NET provide great insight into different various aspects of image processing and by making this system open many other people with new ideas will be able to contribute. Once this decision is made the developed library will be added to sourceforge.net or google.com/code.

8. EXPERIMENTAL RESULTS

Initial experimental results using the ssFAM neural network system will be presented here. The experiment was conducted in front of a projector system. A Logitech^{TM}web camera was used in this experiment with resolution of 1 MegaPixel. The camera was placed about 6 feet away from the projector at a tilted angle of about 30 degrees. The hand in this experiment was approximately 5 feet away from the camera. The application was started and the presenter performed different hand gestures in front of the camera. The feature extraction was performed real-time and recorded in a file format for later use. Each gesture was recorded separately. Afterwards, online training and testing was performed from the accumulated real-time data collection. The results are presented in the form of a confusion matrix in Fig.9.

The confusion matrix depicts the rates of correct classification of gestures and the rates of misclassification. Overall it can be observed that for each performed gesture, the majority of the time it was classified correctly. Gesture 5 had the highest rate of correct recognition of 88% and this is in accordance with the observable fact that its different from all the other gestures, since only a thumb is presented. Gesture 6 and gesture 1 have the highest rate of confusion with each other. Since gesture 1 involves all the fingers being present and gesture 6 involves 3 fingers being present, the cause of the error lies in the fact that in the case of gesture 1, all fingers may not be detected all the time, especially if the fingers are spaced close to each other.

9. CONCLUSIONS

In this project we explored the area of hand-gesture recognition for applications in human-computer interaction interfaces. A system was developed for classifying both dynamic and static gestures and based on the combination employed to Microsoft Power Point ^{TM}presentations. The problems that were dealt with to a sufficient degree of success were segmentation and tracking of the hand, classification of trajectories of the hand motion and classification of the static hand gestures.

Our system consists of several modules such as motion detection, segmentation, feature extraction, and classification. Motion detection is done through differencing consecutive frames. Segmentation is achieved from

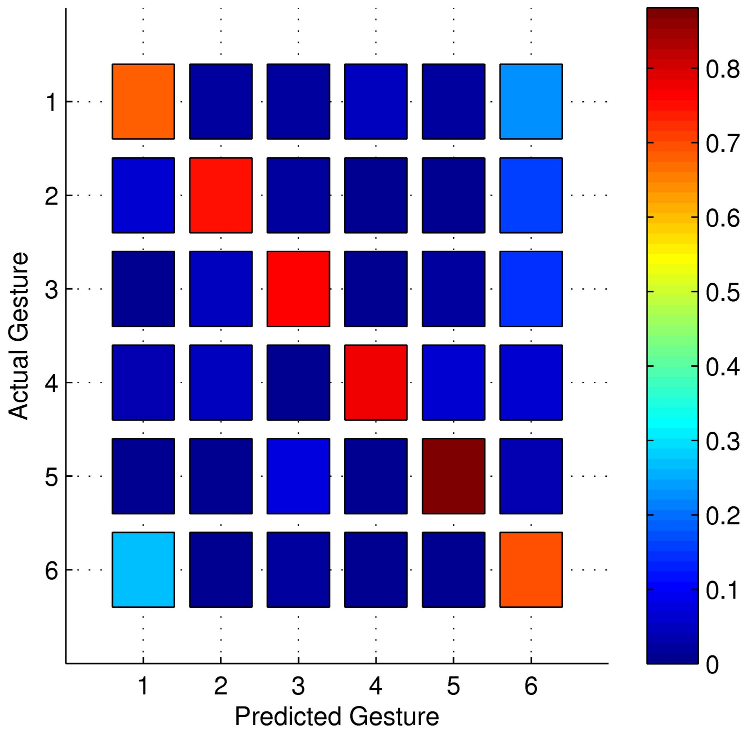

Figure 9. Confusion matrix for the 6 initial gestures: The colors at each (x,y) position show the percentage for which the gesture represented by the y coordinate was classified as the gesture represented by the x coordinate.

Performed Gesture	Confused Gesture	Rate of Confusion (%)
6	1	27.17
1	6	22.92
2	6	15.24
3	6	14.79
5	3	7.12
4	6	6.52
2	1	6.24
4	5	6.09

Table 2. This table shows a the gestures with the highest rate of confusion and the corresponding gestures they were confused with

differencing from a reference frame. Feature extraction is then performed for acquiring the number of fingers present, the arm orientation, and the center of the palm. The fingers are detected through the FAST algorithm. The center of the palm is determined by attempting to fit the largest circle. The orientation of the arm is determined by finding white pixels points on the boundary of the ROI.

Overall we were able to develop a working prototype using the techniques mentioned in this report. A classification rate of 75 % was achieved for identifying the gestures shown in Fig.6. This field has many opportunities for future research and improvement. Questions like tracking in unstable/changing backgrounds, quick movements of the hand, better scene understanding for recognition of the hand, appropriate use of color information, are just a few topics needed to be dealt with. One direction of future research is the investigation of static gesture features that are more robust to mild wrist movement.

ACKNOWLEDGMENTS

This material is based upon work/research supported in part by the National Science Foundation under Grant No. 0647120 and Grant No. 0647018. Also, Todd Alexander wishes to acknowledge partial support by the National Science Foundation under Grant No. 0647120. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Myers, B. A., "A brief history of human-computer interaction technology," *interactions* **5**(2), 44–54 (1998).

[2] Vogler, C. and Metaxas, D., "A framework for recognizing the simultaneous aspects of american sign language," *Comput. Vis. Image Underst.* **81**(3), 358–384 (2001).

[3] Fang, G., Gao, W., and Ma, J., "Signer-independent sign language recognition based on sofm/hmm," *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on* **N/A**, 90–95 (2001).

[4] Jiangqin, W., wen, G., yibo, S., wei, L., and bo, P., "A simple sign language recognition system based on data glove," *Signal Processing Proceedings, 1998. ICSP '98. 1998 Fourth International Conference on* **2**, 1257–1260 vol.2 (1998).

[5] von Hardenberg, C. and Bérard, F., "Bare-hand human-computer interaction," in [*PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*], 1–8, ACM, New York, NY, USA (2001).

[6] Laptev, I. and Lindeberg, T., "Tracking of multi-state hand models using particle filtering and a hierarchy of multi-scale image features," in [*Proc. Scale-Space'01*], *Lecture Notes in Computer Science* **2106**, 63+ (2001).

[7] Malassiotis, S. and Strintzis, M. G., "Real-time hand posture recognition using range data," *Image Vision Comput.* **26**(7), 1027–1037 (2008).

[8] "Control your computer or car by waving your hands." `http://www.switched.com/2007/10/08/control-your-computer-or-car-by-waving-your-hands/` (2008).

[9] Rainer, J. M., "Fast hand gesture recognition for real-time teleconferencing applications," (2001).

[10] McAllister, G., McKenna, S. J., and Ricketts, I. W., "Hand tracking for behaviour understanding," *Image Vision Comput.* **20**(12), 827–840 (2002).

[11] Pentland, A., "Looking at people: Sensing for ubiquitous and wearable computing," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(1), 107–119 (2000).

[12] Miners, B., Basir, O., and Kamel, M., "Understanding hand gestures using approximate graph matching," *Systems, Man and Cybernetics, Part A, IEEE Transactions on* **35**, 239–248 (March 2005).

[13] Wu, Y., Lin, J., and Huang, T., "Analyzing and capturing articulated hand motion in image sequences," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **27**, 1910–1922 (Dec. 2005).

[14] Oka, K., Sato, Y., and Koike, H., "Real-time fingertip tracking and gesture recognition," *Computer Graphics and Applications, IEEE* **22**, 64–71 (Nov/Dec 2002).

[15] Hejlsberg, A., Wiltamuth, S., and Golde, P., [*C# Language Specification*], Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003).

[16] "Aforge.net." http://code.google.com/p/aforge/ (2008).

[17] "Microsoft directshow." http://msdn.microsoft.com/en-us/library/ms783323(VS.85).aspx (2008).

[18] Jerde, T., Soechting, J., and Flanders, M., "Biological constraints simplify the recognition of hand shapes," *Biomedical Engineering, IEEE Transactions on* **50**, 265–269 (Feb 2003).

[19] Gonzalez, R. C. and Woods, R. E., [*Digital Image Processing*], Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001).

[20] Alkaabi, S. and Deravi, F., "Candidate pruning for fast corner detection," *Electronics Letters* **40**, 18–19 (Jan. 2004).

[21] Ionescu, B., Coquin, D., Lambert, P., and Buzuloiu, V., "Dynamic hand gesture recognition using the skeleton of the hand," *EURASIP J. Appl. Signal Process.* **2005**(1), 2101–2109 (2005).

[22] Angel, E. and Morrison, D., "Speeding up bresenham's algorithm," *Computer Graphics and Applications, IEEE* **11**, 16–17 (Nov 1991).

[23] Anagnostopoulos, G., Georgiopoulos, M., Verzi, S., and Heileman, G., "Reducing generalization error and category proliferation in ellipsoid artmap via tunable misclassification error tolerance: Boosted ellipsoid artmap," *Proc. IEEE-INNS-ENNS Int'l Joint Conf. Neural Networks (IJCNN'02)* **3**, 2650–2655 (2002).

[24] Grossberg, S., "Adaptive pattern recognition and universal encoding ii: Feedback, expectations, olfaction, and illusions," *Biological Cybernetics* **23**, 187–202 (1976).

[25] Anagnostopoulos, G., Bharadwaj, M., Georgiopoulos, M., Verzi, S., and Heileman, G., "Exemplar-based pattern recognition via semi-supervised learning," *Neural Networks, 2003. Proceedings of the International Joint Conference on* **4**, 2782–2787 vol.4 (July 2003).