3D Finger Posture Detection and Gesture Recognition on Touch Surfaces

Vadim Zaliva
Tristero Consulting, Email: lord@crocodile.org

***Abstract*—As the use of touch surfaces for user interfaces becomes more common, advances in the interpretations of touch input have lagged behind and are still limited to only the most basic of motions. Richer gesture-based human-computer interactions could serve to advance a wider acceptance of touch-based technology in a variety of fields. Using a 3D finger posture rather than just the 2D contact point in gesture definition opens the door to very rich, expressive, and intuitive gesture metaphors. In this paper, we present algorithms and methods for estimating the parameters of 3D finger postures on a touch surface, as well as a gesture recognition framework which uses an Artificial Neural Network to recognize 3D gestures on touchpads and touchscreens.**

I. Introduction

Gesture-based interfaces should become more prevalent as gestures are among the most primary and expressive forms of human communication. [29]. However, modern models of gesture interaction on touch surfaces remain relatively rudimentary. Companies like Apple and Microsoft are gradually introducing gesture metaphors into their products, but they are still limited to abstract gestures like the "four-finger swipe" or primitive metaphors such as "pinch to zoom." Significant progress could be made in the area of gesture recognition, allowing for the introduction of more complex gesture metaphors and thus, more complex interaction scenarios. Using a 3D finger posture rather than just the 2D contact point in gesture definition opens the door to very rich, expressive, and intuitive gesture metaphors.

A. Related Work

Virtually all modern touch interfaces consider only the point of finger contact with the touchpad, limiting themselves to measuring a pair of coordinates for each finger application.

In [24] the *pitch* and *yaw* parameters of finger pose using probabilistic inference are tracked using capacity proximity sensors. [25] attempts to distinguish thumb rolls and swipes.

An earlier work by New Renaissance Institute lays a foundation for real-time extraction of 3D posture information from tactile images [10], [17], [20], [26], [27]. Our work is an extension and continuation of this work.

B. Finger Posture

In this paper, we consider the interaction scenario of the user performing finger gestures on a flat, touch-sensitive surface. Each finger contacting the touch surface has a *position* and *posture* defined in a coordinate system illustrated in Figure 1.

Most existing touch interfaces operate only from finger *position*, which represents a point of contact between a finger and the touch surface in an X-Y plane.

However, this same point of contact could correspond to different finger *postures* in three dimensional space. A posture could be described via *Euler angles*, commonly denoted by letters: (ϕ, θ, ψ) (using *Z–X–Z convention*). The Euler angles describing finger posture are shown in Figure 1.

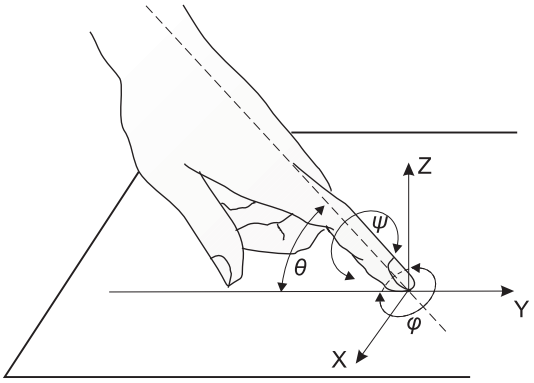

Fig. 1. Finger Posture.

The "neutral posture" is defined as a pose causing the least discomfort to the user during long term use and which is also the natural starting point for most common touchpad actions. Some ergonomic studies [6] recommend a straight wrist posture and static loading of the arm and shoulder while avoiding excess finger flexion.

II. Feature Extraction

The touch sensor reading is a frame of pixels, each representing an intensity value (pressure, brightness, or proximity).

Each frame first passes through a "frame pre-processing" step which includes: normalizing pixel values, accommodating defective sensels, and thresholding.

The next step is feature extraction: calculating a set of features (*feature vector*) for each frame. Each feature is described in Section II-B.

A. Image Moments

Discrete Cartesian geometric moments are commonly used in the analysis of two-dimensional images in computer vision [4], [3].

The moments definition uses a *pixel intensity function*. Two useful kinds are $I_{raw}(x, y)$, which simply returns the

frame pixel's value and $I_{bin}(x, y)$, which uses a step threshold function and will return zero for pixel values below a specified threshold and 1 for values above it, effectively producing a binary image.

The *moment of order* $(p + q)$ for a grayscale image of size M by N with pixel intensities I_{raw} and I_{bin} can be defined respectively as:

$$\widetilde{M}_{p,q} = \sum_{x=1}^{M}\sum_{y=1}^{N} x^p y^q I_{raw}(x, y)$$
$$M_{p,q} = \sum_{x=1}^{M}\sum_{y=1}^{N} x^p y^q I_{bin}(x, y)$$

A *central moment of order* $(p + q)$ for a grayscale image of size M by N with pixel intensities I_{raw} and I_{bin} can be defined respectively as:

$$\widetilde{\mu}_{p,q} = \sum_{x=1}^{M}\sum_{y=1}^{N} (x - \bar{x})^p (y - \bar{y})^q I_{raw}(x, y)$$
$$\mu_{p,q} = \sum_{x=1}^{M}\sum_{y=1}^{N} (x - \bar{x})^p (y - \bar{y})^q I_{bin}(x, y)$$

B. Features

In this section we will define a set of useful features which could be extracted from a frame, obtained from a touch sensor.

1) Area: $M_{0,0}$ is the number of non-zero pixels in frame. This is sometimes called the *area*.

We use the term *finger imprint* to refer to a subset of frame pixels having a value exceeding the specified threshold.

2) Average Intensity: This feature represents the average intensity of non-zero pixels in the frame: $\bar{i} = \frac{\widetilde{M}_{0,0}}{M_{0,0}}$

3) Centroids: Interpreting pixel intensity as a surface density function allows us to calculate the geometric centroid of a finger imprint.

Using I_{raw} as an intensity function gives us centroids:

$$\bar{\tilde{x}} = \frac{\widetilde{M}_{10}}{M_{00}}; \bar{\tilde{y}} = \frac{\widetilde{M}_{01}}{M_{00}}$$

Using I_{bin} as an intensity function gives us centroids:

$$\bar{x} = \frac{M_{10}}{M_{00}}; \bar{y} = \frac{M_{01}}{M_{00}}$$

Centroids can be used to estimate finger *position*.

4) Normalized Eigenvalues of the Covariance Matrix: A covariance matrix of $I_{bin}(x, y)$ is:

$$\begin{bmatrix} \mu_{2,0} & \mu_{1,1} \\ \mu_{1,1} & \mu_{0,2} \end{bmatrix} \quad (1)$$

The first and second eigenvalues λ_1 and λ_2 of the matrix in equation (1) are proportional to the squared length of the axes of finger imprint [5]. From these, we can define e_1 and e_2 as two features representing scale-invariant normalizations of λ_1 and λ_2:

$$e_1 = \frac{\lambda_1}{\mu_{0,0}}; e_2 = \frac{\lambda_2}{\mu_{0,0}}$$

5) Euler's ϕ Angle: A finger imprint typically has the shape of a (typically oblong) blob. The asymmetry of this blob could be used to estimate Euler's ϕ angle.

The eigenvectors of the matrix in equation (1) correspond to the major and minor axes of the finger imprint. ϕ can be calculated as an angle of the major axis, represented by the eigenvector associated with the largest eigenvalue [4]:

$$\phi = \frac{1}{2} tan^{-1}\left(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}\right) \quad (2)$$

An alternative formula to calculate ϕ is:

$$\phi = \frac{1}{2} cot^{-1}\left(\frac{\mu_{2,0} - \mu_{0,2}}{2\mu_{1,1}}\right) \quad (3)$$

We can use one of the above equations (2), (3), depending on which of $\mu_{1,1}$ or $\mu_{2,0} - \mu_{0,2}$ is zero, to avoid an undefined value caused by division by zero [13].

Due to anatomic limitations and ergonomic considerations, most user interactions on touch surfaces fall within a certain range of ϕ angles, somewhat centered around a value of ϕ corresponding to a neutral posture. Since equation (2) could never evaluate to $\pm\frac{\pi}{2}$ and equation (3) could never evaluate to $-\frac{\pi}{2}$, it is convenient to choose a coordinate system in which the ϕ angle corresponding to a neutral posture does not fall close to $n\pi + \frac{\pi}{2}, n \in \mathbb{Z}$ to minimize the likelihood of their occurrence. For example, a coordinate system in which ϕ value for neutral posture equals 0 is a good choice.

In real-time systems, instead of equation (3) a high-performance, closed-form, single scan algorithm [13] could be used.

6) Euler's ψ Angle: Finger posture changes which cause variation of Euler's ψ angle could be informally described as "rolling" a finger on the surface.

While the finger is in a neutral posture, the left and right edges of its imprint shape typically have roughly equal curvature. As the finger rolls away from the neutral position, the leading edge is usually "flatter" compared to the trailing edge. As ψ increases, the shape changes accordingly: the leading edge becomes flatter while the trailing edge becomes more pronouncedly curved.

These changes in curvature permit the value of Euler's ψ angle to be estimated based on the difference between edge curvatures using the following steps:

The first step is to detect the left and right imprint edges. This could be done using *zero-crossing* on per-row intensity values.

The second step is to fit a polynomial curve to the sets of points constituting the left and right edges. The row number is interpreted as abscissa and the column number as an ordinate. The shape of the edges is approximated with a second degree polynomial, as shown in Figure 2.

If for a given edge the variable r denotes row number and the variable c column number, the equation describing the edge would be:

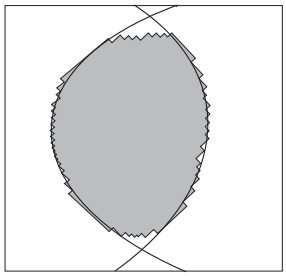

Fig. 2. Parabolas Fitting the Left and Right Edges.

$$c = a_0 + a_1 r + a_2 r^2 \tag{4}$$

The polynomial coefficients could be estimated using least squares:

$$a = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^T y \tag{5}$$

The *signed curvature* of a parabola specified by equation (4) is:

$$k = \frac{c''}{(1 + c'^2)^{\frac{3}{2}}}$$

Taking derivatives gives us:

$$k = \frac{2a_2}{(1 + (a_1 + 2a_2 r)^2)^{\frac{3}{2}}} \tag{6}$$

A parabola curvature is greatest at the vertex which is located at:

$$r_v = -\frac{a_1}{2a_2} \tag{7}$$

Thus a *signed curvature* at vertex point could be calculated by substituting r in equation (6) with r_v from equation (7):

$$k_v = 2a_2 \tag{8}$$

which is also a second derivative c'' from equation (4). As such, it will have opposite signs for parabolas fitting the left and right edges, as one of parabolas will typically concave left while the other will typically concave right.

The sum of the two k_v terms will change magnitudes and signs in a way that monotonically tracks the changing ψ angle that is defined to be zero when parabolas are similar, negative in one direction, and positive in the opposite direction:

$$\psi \propto (left_{k_v} + right_{k_v})$$

where $left_{k_v}$ and $right_{k_v}$ are curvature values at vertex point for parabolas fit to the left and right edges of a finger imprint. Substituting k_v using equation (8) gives the even simpler formula:

$$\psi \propto (left_{a_2} + right_{a_2})$$

where $left_{a_2}$ and $right_{a_2}$ are a_2 coefficients from equation (4) for parabolas fit to left and right edges of a finger imprint found using equation (5).

7) Euler's θ Angle: θ could be estimated using the following shape-based algorithm. Row and column scans are used to find the top, bottom, left, and right edges of a finger imprint. This produces vectors of x coordinates for the left and right edges: $\mathbf{X_l}$ and $\mathbf{X_r}$ respectively and similarly y coordinates for the top and bottom edges: $\mathbf{Y_t}$ and $\mathbf{Y_b}$ respectively. Taking arithmetic mean values of these vectors will give us respective coordinates for the sides of a box roughly approximating the shape of the finger imprint.

An empirical formula shown to provide a good estimate of θ is:

$$\theta \propto \frac{1}{M_{0,0}} \sqrt{(\overline{\mathbf{X_r}} - \overline{\mathbf{X_l}})^2 + (\overline{\mathbf{Y_t}} - \overline{\mathbf{Y_b}})^2}$$

Geometrically, this can be described as the length of a diagonal of a rectangle approximating the finger's imprint normalized by the value of the *area* feature. Note this incorporates several essential details: linear approximation of the edges, usage of a diagonal length, and normalization by $M_{0,0}$ rather than $width \times height$.

Our experiments show that this formula correlates well with finger application angle θ and could be used as an empirical estimator of such. It is also scale-invariant which is important due to anatomical size variations of fingers.

C. ϕ Correction

The shape-based algorithms for calculating ψ and θ described in Sections II-B6 and II-B7 are sensitive to Euler's angle ϕ of the finger's application due to the use of row and column scanning to find the left, right, top, and bottom finger edges. During these operations, rows and columns are defined in a coordinate system in which a projection of the major axis of a finger's *distal phalanx* to the X-Y plane is parallel to the Y axis. The actual finger imprint could be rotated in the X-Y plane by an arbitrary ϕ angle.

To use the algorithms discussed in Sections II-B6 and II-B7, the ϕ angle is calculated first and then used to perform ϕ *correction* before calculating ψ and θ. Equation (9) shows the correction operation – a transformation of vector F containing coordinates of a frame's pixels to F_ϕ by using a rotation matrix, effectively rotating them by angle ϕ about the origin of the coordinate system.

$$F_\phi = \begin{bmatrix} \cos\phi & -sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} F \tag{9}$$

Figure 3 demonstrates the results of ϕ correction.

The effect of ϕ correction on left and right edge detection is shown in Figure 4. The dashed lines show curves approximating uncorrected left end right edges, while the solid lines show curves calculated after ϕ correction. Without ϕ correction, incorrect ψ and θ values will be calculated using shape-based approaches described in Sections II-B6 and II-B7.

D. Signal Processing

A temporal sequence of feature vectors could be viewed as a set of pseudo-continuous signals. Some of these signals

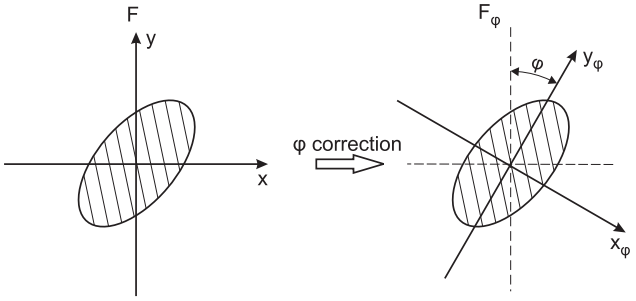

Fig. 3. ϕ correction.

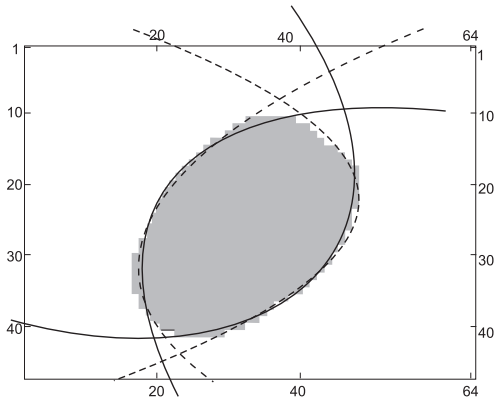

Fig. 4. Effects of ϕ correction on left and right edge detection.

could be used as control inputs to control software or hardware (see Section IV) by varying finger posture and position.

Some signals could benefit from applying filters as described below.

When a human finger touches the sensor surface, it deforms. Some signals, such as Euler's angles cannot be reliably calculated during this initial deformation. This could be addressed by using a *dampening filter*. This filter ignores frames for time t_d following initial finger contact with the sensor surface. To avoid filter activation due to noisy sensor readings, it is activated only if a finger touch is detected after an absence for a given period of time t_n.

Figure 5 illustrates the dampening filter operation. $M_{0,0}$ is used to detect whether a finger is touching the surface. In the depicted example, the finger is removed from the surface at t_0 and re-applied at t_1. Since the duration of finger absence $(t_1 - t_0) \geq t_n$ the dampening filter is activated, suppressing output of unreliable calculations of ϕ, ψ, and θ signals for t_d, until t_2. The dashed line shows suppressed signals values.

A signal's random noise could be attenuated by using a *low-pass filter*. A *causal filter* approach is used to estimate the value of a signal at a given point in time using LOWESS [2] applied to w_s prior values. These values are called *smoothing window*. Such a filter is used for smoothing Euler's angles.

III. Gesture Recognition

A temporal sequence of feature vectors could be used to recognize a set of predefined gestures performed by changing finger posture and position. The gesture recognition module

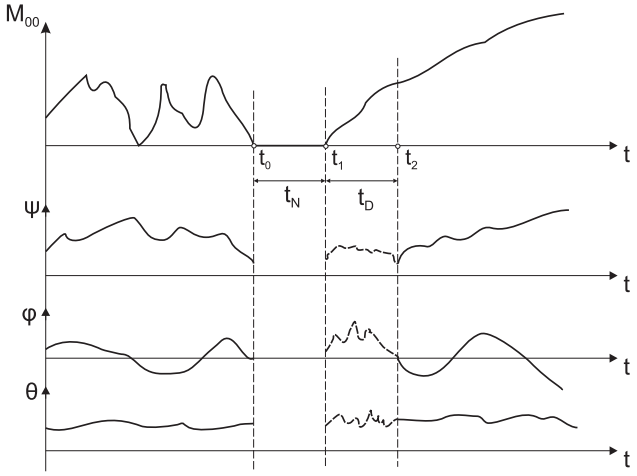

Fig. 5. Effects of application of a dampening filter.

processes a stream of feature vectors and attempts to recognize a gesture presence and boundaries.

A user can perform a variety of gestures, the most basic involving variation of only a single parameter of finger posture or position. The initial set of basic gestures could be:

- Sway User changes x coordinate of finger position (swiping the finger left to right or right to left).
- Surge User changes y coordinate of finger position (swiping the finger towards or away from the body).
- Heave User changes $\bar{i}$ (varying the pressure, applied by the finger to the touchpad).
- Yaw User changes ϕ, varying corresponding angle.
- Roll User changes ψ, varying corresponding angle.
- Pitch User changes θ signal, varying corresponding angle.

A gesture recognition problem could be formulated as a *sequence labelling* [21] problem. Representing each gesture as having two directional (left and right) labels produces the following initial set of gesture labels:

$$\Sigma_0 = \{yaw_l, yaw_r, roll_l, roll_r, pitch_l, pitch_r\} \tag{10}$$

To represent a situation where no gesture is present, an additional *null label*, denoted by symbol $\square$, is introduced producing the final set of labels: $\Sigma = \{\Sigma_0, \square\}$.

Each frame (at time t) could be represented by a feature vector, for example:

$$s_t = (M_{0,0}, \bar{i}, \bar{x}, \bar{y}, \bar{\bar{x}}, \bar{\bar{y}}, e_1, e_2, \phi, \theta, \psi) \tag{11}$$

A sliding window approach to real-time sequence labelling is used, where the classification of a sample at time t is made based on w_d current and previous samples $(s_t, s_{t-1}, ..., s_{t-(w_d-1)})$. The value w_d is called *gesture recognition window size*. This window size is selected experimentally, based on several factors such as sampling rate and average gesture duration.

The input of the classifier at time t is the concatenation of w_d most recent feature vectors:

$$x_t = (s_t, s_{t-1}, ..., s_{t-(w_d-1)}) \quad (12)$$

The output of the classifier is a label from the set Σ.

A. Artificial Neural Network Classifier

To assign labels to each feature vector, we used an ANN classifier.

The classifier will have $|x_t|$ inputs and $|\Sigma_0|$ outputs. The input of the classifier is vector x_t (see equation (12)). The output could be interpreted as a vector of probabilities for each label from set Σ_0 (see equation (10)).

Based on this vector of label probabilities, a single label is selected by applying *accept* and *reject* thresholds: the label is chosen if its probability is above the acceptance threshold, and all other label probabilities are below the rejection threshold. This classification approach is sometimes called "one-of-n with confidence thresholds" [28]. If no label passes the threshold test, the *null label* ($\square$) is assigned.

A simple *feedforward* ANN with two hidden layers using the *tanh* activation function was used. The ANN output layer uses the *logistic* activation function, so as to produce outputs in $[0, 1]$ interval, convenient for probabilistic interpretation.

Under certain conditions, some features cannot be calculated. In this case, we store a special *NULL* symbol, indicating a missing value. Since an ANN could not handle *NULL* values, they have to be handled outside of the ANN. There are two cases of missing values: 1) If within a given window a feature is *NULL* for all frames, do not send these windows to the ANN classifier, assume that no gesture is present, and assign a *null label*, and 2) if within a given window for a feature some values are *NULL*, try to interpolate those missing values by replacing them with the mean value for the respective feature across the window.

B. Principal Component Analysis

Increasing ANN inputs above a certain number can cause a degradation of the performance of the ANN classifier with a noticeable impact on training time and required CPU resources, because the number of ANN cells and required amount of training data grows exponentially along with dimensionality of the input space [1].

To address this problem the number of features used can be limited. However, it is difficult to predict a priori the usefulness of different features in classification decisions. One can employ a *dimensionality reduction* technique such as a Principal Component Analysis (PCA). A PCA operation is applied to an extended feature vector including more moments in addition to those features defined in s_t:

$$s_{pca} = s_t \cup (M_{0,1}, M_{1,0}, \widetilde{M}_{0,0}, \widetilde{M}_{0,1}, \widetilde{M}_{1,0}, \widetilde{\mu}_{1,1}, \widetilde{\mu}_{2,0}, \widetilde{\mu}_{0,2}, \widetilde{\mu}_{2,1}, \widetilde{\mu}_{1,2}, \widetilde{\mu}_{2,2})$$

Each feature in the feature vector is *mean centered* and scaled to have unit variance. The PCA operation, applied to s_{pca}, produces a list of *principal components* corresponding to dimensions in a new space. Low-variance components could then be dropped.

Assuming that the original data has N intrinsic degrees of freedom, represented by M features with $M > N$, and some of the original features are linear combinations of others, the PCA will allow us to decrease the number of dimensions by orthogonally projecting original data points to a new, lower-dimension space while minimizing the error caused by dimensionality decrease. One limitation of this technique is that it could not detect non-linear correlations within data.

The PCA parameters are calculated offline, based on a sample dataset of feature vectors calculated from a representative sequence of pre-recorded frames. The parameters consist of a vector of scaling factors p_s, a vector of offsets p_o, and transformation matrix P_t.

During ANN training and ANN-based gesture recognition, these three parameters are used to convert the feature vector s_{pca} into a vector of principal components: $c_t \subset ((s_{pca} - p_o)p_s)P_t$. An ANN classifier is then used as described in Section III-A, but instead of x_t, a vector r_t is used as input: $r_t = (c_t, c_{t-1}, ..., c_{t-(w_d-1)})$.

C. Gesture Recognition Module Architecture

An architecture of a gesture recognition module is shown in Figure 6. (1) is the input of the module, a vector of features s_{pca}. A PCA transformation is applied to this vector resulting in c_t, marked as (2). The last w_d values of c_t are accumulated in a *recognition window*. The contents of this window are then concatenated into a single vector, r_t (6), which is submitted as input to the ANN. The output of the ANN, (8), is a vector of label probabilities.This vector is interpreted by the label assigning module, which assigns a label,(9), to the current frame. This label is one of the outputs of the recognition module.

Parallel to the "label" data flow depicted in the upper portion of Figure 6, the same features (1) can also be used to obtain smoothed signals representing finger position and posture. A subset s_t of values from input vector s_{pca} (1) is split into two vectors: spatial coordinates of the centroid (4) and remaining features from s_t. The centroid is processed by the Kalman filter resulting in (5) – a vector of smoothed centroid coordinates. Other features are smoothed using LOWESS based on w_s last feature vectors, accumulated in the *smoothing window*. These smoothed signals are re-concatenated with (5) to produce a vector (7) which contains a smoothed version of s_t. This vector, marked as (10), is also an output of this module.

IV. Experimental Results

For ANN Classifier training, we recorded a relatively small (370,108 frames) dataset of various gestures performed by users on a touchpad and manually labeled each frame with the gesture used. Using *repeated random sub-sampling validation*, the dataset was split into K folds combined in different ways into training, validation, and testing subsets for training and measuring performance of our gesture classifier.

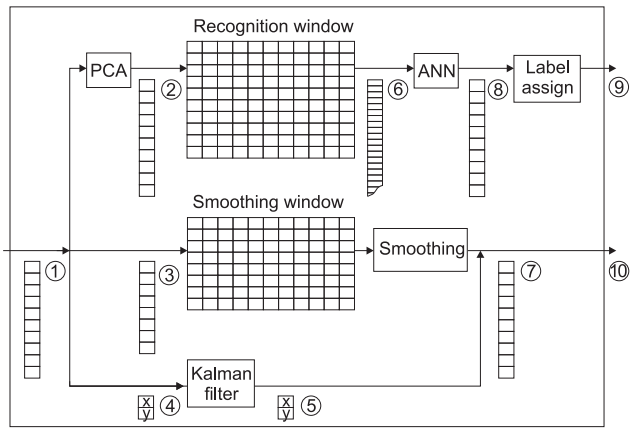

Fig. 6. Gesture Recognition.

The performance of our ANN classifier was sufficient for gesture recognition in real-time on a regular consumer-level PC at a input frame rate of 100 FPS.

To tune classifier parameters, we evaluated results with such metrics as *precision* and *recall* (*micro* and *macro* averaged) and were able to achieve gesture recognition with a miss rate below 1 percent.

To demonstrate applications for 3D gestures, we implemented gesture control for a variety of applications: Microsoft Excel, Google Earth, Wolfram Mathematica, and the OWI robotic arm. These and a large number of other applications have been explored by NRI [7]–[12], [14]–[16], [18], [19].

Videos [22], [23] have been published online showing working examples of this system.

ACKNOWLEDGEMENTS

I would like to thank the New Renaissance Institute for giving me the opportunity to collaborate on this project. The work was funded, in part, by NSF grants IIP-0741237 and IIP-0923986 and based on earlier and recent work by Lester F. Ludwig who has also provided guidance, inspiration, and suggestions.

I also wish to thank Seung Lim for early software designs, optical sensor modifications, and her help together with Eric Sum on experimental aspects, ANN training, and robotics interfacing.

Dr. Ludwig also asked that I acknowledge the earlier NSF SBIR Phase I team, including contributing leader Steven H. Simon and members Amory Schlender, Andrew Graham, and Tony Ricciardi.

REFERENCES

[1] BISHOP, C. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[2] CLEVELAND, W. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association* (1979), 829–836.

[3] DAVIES, E. *Machine vision: theory, algorithms, practicalities*. Signal Processing and its Applications. Elsevier, 2005.

[4] FLUSSER, J., SUK, T., AND ZITOVÁ, B. *Moments and Moment Invariants in Pattern Recognition*. John Wiley & Sons, 2009.

[5] GIBSON, C. *Elementary Euclidean geometry: an introduction*. Cambridge University Press, 2003.

[6] HEALTH AND SAFETY EXECUTIVE STAFF. *Ergonomics of Using a Mouse Or Other Non-Keyboard Input Device*. Research Report Series. HSE Books, 2002.

[7] LIM, S. E. U.S. Patent Application 12/511930: Control of Computer Window Systems, Computer Applications, and Web Applications via High Dimensional Touchpad User Interface.

[8] LIM, S. E. Provisional U.S. Patent Application 61/371,153: High-Dimensional Touchpad Game Controller with Multiple Usage and Networking Modalities.

[9] LIM, S. E. U.S. Patent Application 12/502230: Control of Computer Window Systems, Computer Applications, and Web Applications via High Dimensional Touchpad User Interface.

[10] LUDWIG, L. F. U.S. Patent Application 11/761978: High Parameter-Count Touchpad Controller.

[11] LUDWIG, L. F. U.S. Patent Application 12/875128: Interactive Data Visualization Utilizing HDTP Touchpad, HDTP Touchscreens, Advanced Multitouch, or Advanced Mice.

[12] LUDWIG, L. F. U.S. Patent Application 12/618698: Electronic Document Editing Employing Multiple Cursors.

[13] LUDWIG, L. F. U.S. Patent Application 12/724413: High-Performance Closed-Form Single-Scan Calculation Of Oblong-Shape Rotation Angles From Binary Images Of Arbitrary Size Using Running Sums.

[14] LUDWIG, L. F. U.S. Patent Application 13/026097: Window Manger Input Focus Control for High Dimensional Touchpad (HDTP), Advanced Mice, and Other Multidimensional User Interfaces.

[15] LUDWIG, L. F. Provisional U.S. Patent Application 61/482,248: Simple Touch Interface and HDTP Grammars for Rapid Operation of Physical Computer Aided Design (CAD) Systems.

[16] LUDWIG, L. F. U.S. Patent 7620915: Electronic Document Editing Employing Multiple Cursors.

[17] LUDWIG, L. F. U.S. Patent 6570078: Tactile, Visual, and Array Controllers for Real-Time Control of Music Signal Processing, Mixing, Video, and Lighting.

[18] LUDWIG, L. F. U.S. Patent 7408108: Multiple-Parameter Instrument Keyboard Combining Key-Surface Touch and Key-Displacement Sensor Arrays.

[19] LUDWIG, L. F., AND HU, V. U.S. Patent Application 13/026248: Enhanced Roll-Over, Button, Menu, Slider, and Hyperlink Environments for High Dimensional Touchpad (HTPD), other Advanced Touch User Interfaces, and Advanced Mice.

[20] LUDWIG, L. F., AND LIM, S. E. U.S. Patent Application 12/418605: Multi-Parameter Extraction Algorithms For Tactile Images From User Interface Tactile Sensor Arrays.

[21] NGUYEN, N., AND GUO, Y. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th international conference on Machine Learning* (2007), ACM, pp. 681–688.

[22] NRI. NRI HDTP gesture recognition results (video). http://www.youtube.com/user/nriresearch#p/u/4/ryqNNfVsXeQ, 2011. [Online; accessed 12-July-2011].

[23] NRI. NRI HDTP gesture recognition results (video file). http://www.crocodile.org/lord/video/HDTP-demo-01.mp4, 2011. [Online; accessed 17-July-2011].

[24] ROGERS, S., WILLIAMSON, J., STEWART, C., AND MURRAY-SMITH, R. Anglepose: robust, precise capacitive touch tracking via 3d orientation estimation. In *Proceedings of the 2011 annual conference on Human factors in computing systems* (2011), ACM, pp. 2575–2584.

[25] ROUDAUT, A., LECOLINET, E., AND GUIARD, Y. Microrolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *Proceedings of the 27th international conference on Human factors in computing systems* (2009), ACM, pp. 927–936.

[26] SIMON, S., AND GRAHAM, A. U.S. Patent Application 13/009845: Use of Fingerprint Scanning Sensor Data to Detect Finger Roll and Pitch Angles.

[27] SIMON, S. H., AND GRAHAM, A. U.S. Patent Application 12/541948: Sensors, Algorithms and Applications for a High Dimensional Touchpad.

[28] STATSOFT. Electronic statistics textbook. http://www.statsoft.com/textbook/, 2011. [Online; accessed 1-July-2011].

[29] WACHS, J. P., KÖLSCH, M., STERN, H., AND EDAN, Y. Vision-based hand-gesture applications. *Commun. ACM 54* (February 2011), 60–71.