Hand Sign Recognition through Palm Gesture and Movement

Image Processing, EE 368, Spring 2012

Oleg Rumyantsev, Matt Merati, Vasant Ramachandran
Stanford University

olegr@stanford.edu, mmerati@stanford.edu, vasantr@stanford.edu

***Abstract* — We propose and implement a new method of detecting hand signs by combining skin color detection in transformed 2D normalized RGB or YCbCr color space, PCA-based detection of hand gestures and movement tracking of hand centroids. Hand gesture recognition is a field of research with growing applications, and the implemented system uses movement-tracking pairing for more granular recognition and control. Our algorithm was successfully realized in a real-time application for robust recognition of any trainable gesture-movement pairs.**

I. Introduction

Hand gesture recognition is a problem that has elicited significant attention and research as computational capabilities, camera performance, and computer-vision-style learning algorithms have rapidly improved over the past few years. Such research is driven by the tremendous growth and variety in development of applications that require some form of gesture comprehension: these include remote hardware control, game controls, affective computing, and other endeavors in enhanced human-computer interaction [1,2]. There are, however, fundamental limitations to most current systems for gesture detection based off training on a set of pre-defined gestures. Non-uniform lighting conditions and less-than-ideal camera resolution and depth of color limit the number and accuracy of possible gesture classifications in practice [3]. Moreover, the modeling and analysis of hand gestures is complicated by the variegated treatment required for adequate detection of static gestures, which represent a combination of different finger states, orientations, and angles of finger joint that are often hidden by self-occlusion [4,5]. In computer vision (CV) based solutions such as ours, hand gestures are captured by web cameras which offer resolutions that allow only a general sense of the figure state to be detected [4]. On the other hand, modeling gestures as temporal objects (emphasizing an understanding the movement of the hand as a pointer to the nature of the gesture) allows for greater accuracy and better differentiation of gestures [5]. Additionally, the problem of hand-gesture recognition usually occurs in contexts where gestures involving finger conformation are accompanied by movement of the hands relative to the body.

Thus, we hypothesize that a system which can group gestures based both on hand conformation and movement tracked during the timespan of the gesture improves the robustness, accuracy, and detection precision of a system. We implement fast online processing of gestures and movements into a set of gesture/movement buckets from the webcam-MATLAB interface. The detection process uses skin color detection, filtering and hand-postures comparison algorithms similar to [3] in the initial process of identifying or detecting hand gestures to train a principal component analysis based classification system. The PCA algorithm establishes an eigenspace on a training set and performs classification using a Euclidean metric within the eigenspace for the test set from our webcam. The results allow for real-time classification of gesture-movement pairs in eight different directions and any number of trainable hand postures (based on the variety within the training set). Current version is trained on for palm gestures, so the total number of simultaneous gesture-movement pairs performed by both hands is $(8\cdot4)^2=1024$ different hand signs.

II. Algorithms and Implementation

Application Structure

User interface for the hand gesture recognition application was developed using MATLAB GUI (Graphical User Interface). It allows control of skin color calibration (including selection of different color spaces), visualization of palm and face tracking, recording of training palm gestures. During the real-time run the application handles video streaming from the web-camera and timing of frame processing. Frame rate can exceed 20 fps which is more plenty for reliable human gesture detection.

Skin color Detection

There are numerous methods for skin color detection based on different color schemes. Real-time gesture recognition as well as other real-time applications require fast and computationally inexpensive algorithms for skin detection. Another important requirement for the algorithm is to perform well under different illumination conditions, for different skin tones, and be robust with respect to shadows on skin (that are inevitable when a person gesticulates). The most commonly used color spaces for skin detection are YCbCr, RGB, normalized RGB, HSV [1,11]. Based on the limitation of the algorithm run time we suggest using a 2D color space as it can yield real-time performance. Suitable color spaces are YCbCr and normalized RGB as the majority of built-in or external web cameras can capture video in this formats and no additional color conversion step is needed. Also, these color schemes they have shown relatively good performance. To make the application suitable for a variety of illumination conditions a user can choose the color scheme between YCbCr and normalized RGB based on the technical specifications of available web-camera and then perform calibration. At this step a snapshot is taken and a user is prompted to specify a region belonging to skin (see figure 1).

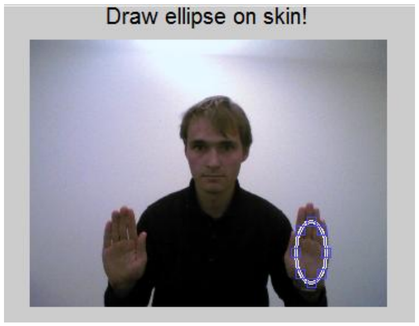

Fig. 1 Skin color calibration. User is prompted to select skin region.

For YCbCr color scheme only the chrominance components are taken into account giving robustness against brightness changes. When normalized RGB color space is used, number of dimensions is also reduced to two by leaving normalized r and g components:

$$r = \frac{R}{R+B+G}, \qquad g = \frac{R}{R+B+G},$$

as the b = 1 – r – g component is redundant. On the next step ‘skin’ pixels of the calibration image are projected into the used 2D space. Skin color model is constructed by fitting the distribution of pixels in the color space by a tilted ellipse. Mathematically, the main axes of such ellipse are in direction of maximal variance in the data distribution, so the tilt angle of the ellipse can be computed using the diagonal elements of the correlation matrix between two observables in the color space:

$$\theta = \arctan(CORR(r,g)_{1,2}) = \frac{COV(r,g)_{1,2}}{\sigma_r \sigma_g}$$
$$or \quad \theta = \arctan(CORR(Cr,Cb)_{1,2})$$

The following angular transformation of the color space yields axes in the direction of maximal variance so that major and minor ellipse half-axis can be computed (same for Cb, Cr):

$$\begin{bmatrix}\alpha \\ \beta\end{bmatrix} = \begin{bmatrix}\cos\theta & \sin\theta \\ -\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}r \\ g\end{bmatrix}$$
$$a = 2std(\alpha)$$
$$b = 2std(\beta)$$

The coefficient 2 in front of the standard deviation was experimentally adjusted to give good estimation of the confidence interval (it basically sets the threshold on how far from the center of the distribution a pixel can be to be classified as ‘skin’). Figure 2 shows an example of skin color calibration. All pixels lying inside the ellipse are classified as skin.

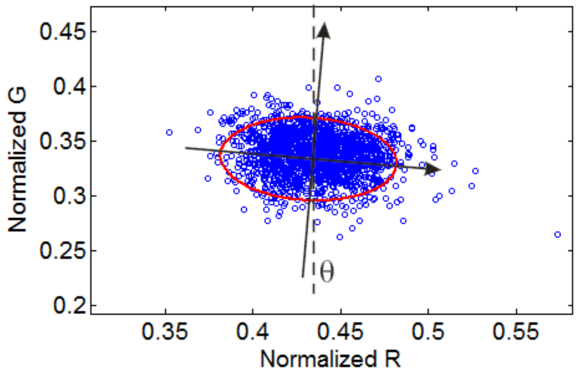

Figure 2. Example of skin color calibration in normalized RGB space (same in CbCr color space). Distribution of skin pixels (blue circles) is fitted with a tilted ellipse (red line) with axes in direction of maximal variance. Ellipse major and minor half axes are set to standard deviation scaled by factor 2 to ensure good separation between ‘skin’ and ‘non-skin’ pixels.

After calibration is performed the following steps are done in real-time:

1) New frame acquired.
2) Masking of skin pixels (color space transformation described above) results in binary mask.
3) Morphological closing is applied to the mask to remove small holes.
4) Larger holes in the mask are removed.
5) Connected regions labeling and filtering by size performed – three largest areas are attributed to face and two palms.
6) Bounding boxes for the regions corresponding to left and right palms are determined.
7) Binary masks for left and right hand are extracted from the large mask and resized to standardized dimensions. (This ensures scale invariance, so that the distance between hands and camera may change).

Palm gesture recognition

Hand gesture detection and recognition has been a fast growing, challenging and interesting area in real-time applications such as sign language recognition and remote control for games, hardware, and virtual reality. A large number of hand recognition algorithms have been developed

from decades. Principal Component Analysis (PCA) is one of the most successful techniques that has been used in hand recognition.
The goal of PCA is to reduce the dimensionality of the image data. The input signals are highly noisy (e.g. the noise is caused by varying lighting conditions, gesture movements, etc.), yet the input images are not completely random and in spite of their differences there are patterns which occur in any input signal. The input image is projected to a new coordinate system containing a set of ordered eigenvectors, and the M highest eigenvectors maintain the most of the variation demonstrated in the original image set.

We project a large 1-D vector of training image vectors constructed from 2-D hand gesture image into the compact principal components of the feature space. This is called the eigenspace projection. The eigenspace is computed by identifying the eigenvectors of the covariance matrix extracted from a set of hand postures training images. These eigenvectors are representatives of the principal components of the training images and are normally ortho-normal to each other.

Eigenvectors and eigenvalues are vectors and numbers associated to square matrices, and together they provide the eigen-decomposition of a matrix which analyzes the structure of this matrix. Even though the eigen-decomposition does not exist for all square matrices, it has a particularly simple expression for a class of matrices often used in multivariate analysis such as covariance matrices. Principal component analysis is obtained from the eigen-decomposition of a covariance matrix and gives the least square estimate of the original data matrix.

PCA based approaches are normally divided into two stages: training and testing. In the training stage, an eigenspace is created from the hand gestures training images using PCA and the hand training images are projected to the eigenspace for classification. In the testing stage, each captured frame of a hand gesture image is projected to the same eigenspace and classified by minimum Euclidean distance [3].

1. Training Stage

The various steps to calculate eigenimages are:

1A. Capture training images data. Assume we have 4 sets of hand training images called 1) Hand fist, 2) Hand Five, 3) Hand Horizontal, and 4) Hand Vertical for each hand (Left & Right). We propose to record 4 video files, 200 frames each with the resolution of 50×50 pixels for each hand gesture. Hands orientation is slightly changed during recording to give robustness against similar changes during the test stage.

1B. Accommodate the data. A 2-D hand gesture image can be represented as 1-D vector by concatenating each row of $N \times N$ training image into a long thin vector $1 \times N^2$. Let's suppose we have M vectors of size N^2 (M =4, N=50) representing a set of sampled images. Then the training set become: $\mathrm{T}_1, \mathrm{T}_2 \ldots \mathrm{T}_M$; where each training image has its subset of $\Gamma_1, \Gamma_2, \ldots, \Gamma_P$ $(\mathrm{P} = 200)$. For this implementation, it is assumed that all images of the training set are stored in a single matrix $\mathrm{T}_{800\times2500}$, where each row of the matrix is an image.

1C. Calculate the mean of training images
In this step we calculate the mean of each training gesture image according to:

$$\bar{T}_m = {}^{1}/_{P} \sum_{p=1}^{P} \Gamma_{m,p},\ m = 1,2,3,4$$

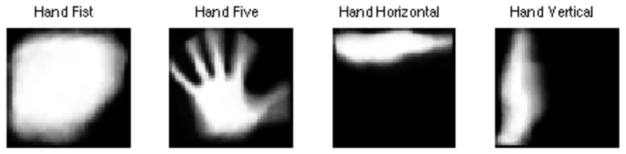

Fig 3. Mean hand gesture training images obtained from 200 frames recordings. During recording hands were slightly rotated to give robustness against similar orientation changes in the test stage.

1D. Calculate the covariance matrix
In the next step the covariance matrix is calculated according to:

$$\mathrm{C}_{i,j} = cov(T_i, T_j) = \mathbb{E}\left[\langle T_i - \mu_i\rangle\langle T_j - \mu_j\rangle\right]$$

where the entries in the row vector $T = [T_1 \ldots T_n]$ are random variables and averaging is done along the columns.

1E. Calculate the eigenvectors and eigenvalues of the covariance matrix according to: $CV = \lambda V$. Each column of the eigenvector is the representative of an eigenimage.

1F. Keep the M highest eigenvectors. For a 50×50 image that means that one must project each training image onto 2500 eigenvectors, which computationally is not very efficient as most of those eigenvectors are not useful for our task. By looking at the eigenvalues plot (see Fig 4.), we can observe that the eigenvalues are approaching very small numbers passing their 10 highest values, hence, we only keep the M=10 highest eigenvectors (see Fig 5.) that can be represented again as images.

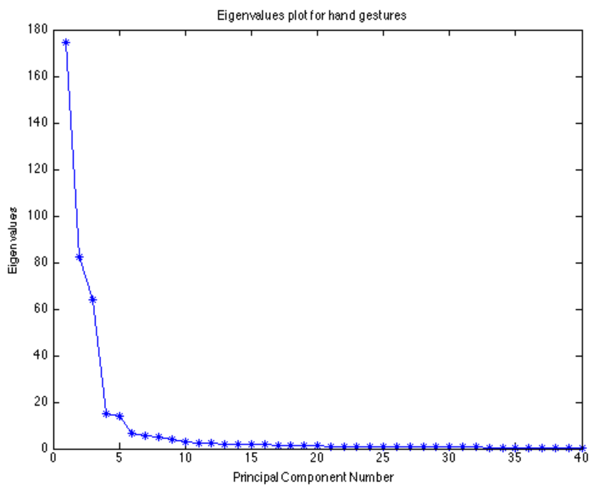

Fig 4. Eigenvalues plot. First 10 eigenvalues are kept for the stage.

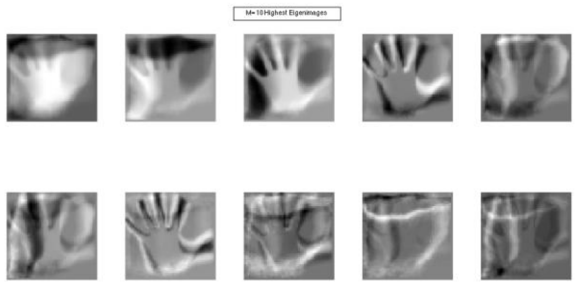

Fig 5. M=10 most Eigenimages

1G. Project the mean of hand gesture training images into the eigenspace and store their weights. In this step hand gesture training images are mapped or projected into the eigenspace, and their weights are stored, which will be used in the testing stage. The weights of every training image are simply the dot product of each mean training image with the M eigenvectors.

$$w_{m,k} = \overline{T}_m \, . \, C_k \, ; \quad m = 1,2,3,4 \; \& \; k = 1,2 \ldots 10$$

2. Testing Stage

The various steps to classify hand gesture captured frames are:

2A. Get the detected hand gesture captured frame. The small image (50 × 50 pixels) that contains the detected hand gesture each frame is passed to the algorithm.

2B. Project the selected hand gesture into the M most eigenvectors to form its weights. The selected small image from part 2A is projected into M=10 highest eigenvectors found in the training stage and weights of each detected hand gesture are stored.

2C. Classify the hand gestures. The Euclidean distance between two weight vectors (training stage weights and testing stage weights) provides a measure of similarity between the corresponding images. The minimum Euclidean distance between the detected hand gesture weights and the training weights of each training image is determined to recognize the hand gesture. If the Euclidean distance between the detected hand gesture weights and other training weights exceeds some threshold τ one can assume that the hand gesture is not defined.

Movement Tracking

The processing algorithm for movement tracking runs every frame iteration and uses the hand centroids obtained in each frame through the skin-detection and face-subtraction process. First, a user-specified window of centroids over the past N frames is stored and continually updated. MATLAB's numerical gradient is computed in both the x and y directions to represent the movement of the centroid in the coordinate plane across the N frames, and the result is averaged across the window to obtain the overall vector of centroid movement over the course of the n-frame-window. This vector is stored in an array that represents a user-specified window of length M of approximate movement vectors in each frame over the past M frames, which is similarly updated each iteration. A final averaging over this array produces the movement vector of length K (in pixels) for the frames, which is then compared (norm-wise) to a threshold value that is designed to filter out jiggles and non-movements. Vectors that fail this threshold test are classified as non-movements, while those that pass are classified into octants (up, up/right, right, down/right, down, down/left, left, and up/left) centered on multiples of $\pi/4$.

Control over the movement detection occurs at three levels. First, the trade-off between lag, detection-failure, and noise-canceling is tuned to the application by specification of the number filtering window length. Secondly, the window size for the averaging approximate movement vectors is also specifiable. Thirdly, the thresholding applied to the final movement vector provides the most explicit layer of control. The results show relatively robust and consistent identification of hand movements.

III. Results

A Matlab based application performing hand gesture-movement recognition was successfully implemented. Advantage of real time processing allows fast and reliable detection of gesture-movement pairs. Figure 6 shows a snapshot of application working and detecting two different hand gestures for left and right hand simultaneously.

A demonstration video of the application run can be seen at http://www.youtube.com/watch?v=mGAAEi3Zf8A&feature=youtu.be.

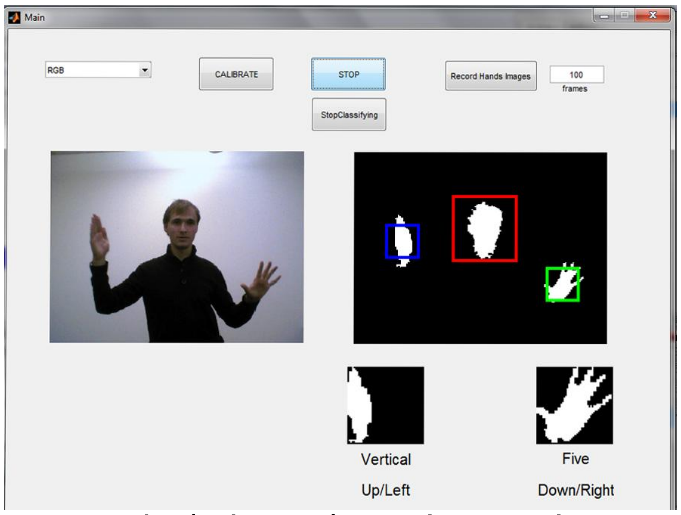

Fig 6. Snapshot of application performing real-time gesture detection. User interface allows skin color calibration, training video recording and video stream control. Even in non-ideal illumination (that can be seen in the snapshot) and poor web-cam quality algorithm performs robust detection.

Overall performance of the algorithm is very good although it has some limitations. One of the main limitations is that a person should be wearing long sleeves; otherwise all arms are detected and treated as palms. Another hardware limitation is related to automatic gain control of the majority of web-cameras. As a result, the color of the scene may significantly fluctuate and be very far from natural color tones. Use of camera with non-adjustable gain and higher dynamic range may be beneficial.

IV. Applications and Future Work

Applications of a sophisticated temporal-gesture detection and recognition system are myriad: we enumerate some of the more salient areas of research and development.

Sign Language Recognition

Some of the more revolutionary developments in computer interfaces and human-computer interaction with speech-impaired or deaf individuals involve sign language recognition. Gesture recognition software to translate sign language to text is increasingly important in such applications[6]. Moreover, as sign language speakers exhibit variable patterns of associated hand-movement based on environmental, cultural, and “dialect”-based factors, a temporal-gesture detection system is particularly relevant[7].

Socially Assistive Robotics and Affective Computing

Socially assistive robotics utilize appropriate sensors for movement and gesture tracking(accelerometers and gyros) that are worn on the body of the patient and provide real-time value readouts. This information allows robots to assist in patient rehabilitation during recovery from motor-sensory trauma or debilitating strokes[8].

Affective computing allows for enhanced computer interaction through the capture of data about the user's physical state or behavior that is analogous to the cues humans use to perceive emotions in others. Gesture recognition software allows the computer to process, interpret, and adapt behavior to the emotional state of the user[9].

Remote Control for Games, Hardware, and Virtual Reality

Gestures can be used to control interactions within video games to try and make the game player's experience more interactive or immersive[10]. Similarly, through the use of gesture recognition, "remote control with the wave of a hand" is possible for a variety of devices.

References

[1] Soontranon, N.; Aramvith, S.; Chalidabhongse, T.H.; , "Improved face and hand tracking for sign language recognition," Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on , vol.2, no., pp. 141- 146 Vol. 2, 4-6 April 2005

[2] Paulraj, M.P.; Yaacob, S.; Desa, H.; Hema, C.R.; Ridzuan, W.M.; Majid, W.A.; , "Extraction of head and hand gesture features for recognition of sign language," Electronic Design, 2008. ICED 2008. International Conference on , vol., no., pp.1-6, 1-3 Dec. 2008

[3] Dardas, N.H.; Petriu, E.M.; , "Hand gesture detection and recognition using principal component analysis," Computational Intelligence for Measurement Systems and Applications (CIMSA), 2011 IEEE International Conference on , vol., no., pp.1-6, 19-21 Sept. 2011

[4] Zhu, Hong-Min, Pun, Chi-Mun, "Movement Tracking in Real-Time Hand Gesture Recognition" Information Technology: Computer and Information Science 2010. IEEE/ACIS 2010. International Conference on , vol.1, no., pp. 135- 137 Vol. 2, 18-20 Aug 2010.

[5] Y. Wu, and T. S. Huang, “Hand modeling, analysis and recognition,” IEEE SIGNAL PROCESSING MAGAZINE, vol. 18, no. 3, pp. 51-60, 2001.

[6] Pavlovic, V., Sharma, R. & Huang, T. (1997), "Visual interpretation of hand gestures for human-computer interaction: A review", IEEE Trans. Pattern Analysis and Machine Intelligence., July, 1997. Vol. 19(7), pp. 677 -695.

[7] Fuller, J., Hollrah, B., Lewis, J. & McCaskill, C. (Eds) 2004-2005. *Black Perspectives on the Deaf Community*. Gallaudet University and the U.S. Department of Education Rehabilitation Services Administration Grant for Region III, CFDA # 84, 160A

[8] Kai Nickel, Rainer Stiefelhagen, “Visual recognition of pointing gestures for human-robot interaction,” Image and Vision Computing, vol 25, Issue 12, December 2007, pp 1875-1884.

[9] Tao, Jianhua; Tieniu Tan (2005). "Affective Computing: A Review". Affective Computing and Intelligent Interaction. LNCS 3784. Springer. pp. 981–995.

[10] Kue-Bum Lee, Jung-Hyun Kim, Kwang-Seok Hong, “An Implementation of Multi-Modal Game Interface Based on PDAs,” Fifth International Conference on Software Engineering Research, Management and Applications, 2007 .

[11] J.C. Terrillon, M.N. Shirazi, H. Fukumachi, and S. Akamatsu, “Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images”, Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition, March 2000, IEEE Comptuer Society, pp 54-61.

[12] N. Dardas, N. Georganas, “Real Time Hand Gesture Detection and Recognition Using Bag-of-Features abd Multi-Class Support Vector Machine” , IEEE Transactions on Instrumentation and Measurement-2011

Appendix

Group Breakdown:
Poster presentation and report writing – all authors.

GUI-interface, video acquisition, timing of frame processing, skin detection, hand masks extraction for detection and tracking – Oleg Rumyantsev

Gesture Recognition, Principal Component Analysis training and testing stages – Matt Merati

Movement detection, tracking, calibration – Vasant Ramachandran