Hand Gesture Detection Using Haar Classifier with Appropriate Skin Color, Kernal Sizing & Auto Thresholding

Sadman Shahriar Alam, Akib Jayed Islam, Nahid Nasrin, Khandoker Tanjim Ahammad

Abstract— Hand gesture detection & recognition intends to detect & recognize some meaningful hand sign. It is an utmost challenge to design such intelligent human-computer interface. 3D hand gesture recognition, one of the most advanced technologies for building smart communication method with computers, has been developing an enormous research interest in computer vision, pattern recognition and human-computer interaction (HCI) system. The developing depth sensors enormously propelled different hand motion identification methodologies and applications, which were extremely restricted in the 2D area with conventional cameras. In this paper, we provided a productive method for image preprocessing and detection of certain gesture, advance classifier theory based on genetic algorithm, boosting algorithm & Fischer's linear discriminant algorithm. Improvising the traditional preprocessing of image for gesture detection where skin color detection, kernel matrix evolution & auto-thresholding of image are the most challenging part of this thesis paper. This paper additionally displays a review of the state-of-the-art research for 3D hand motion recognition in four perspectives: 3D hand modeling, basic sensors capable for detecting hand gesture, static hand motion acknowledgment, hand direction motion acknowledgment, continuous hand gesture recognition and related applications of hand gesture.

Index Terms— Boosting algorithm, genetic algorithm, hand gesture, Haar like features, Human-Computer Interaction(HCI), Fischer's Linear Discriminant Algorithm, input test set.

1 INTRODUCTION

Hand gestures are basic movements of a person's hands and are the atomic correspondence parts representing the thoughts of a person [1]. Evolutionary anthropologists inform us the use of hand gestures has been used since the beginning of human history and are much older than speech [2]. Moreover, hand gestures are common, pervasive and important part of spoken language, and researchers have asserted that gesture and sound frame a firmly integrated system during human cognizance [3]. Inspired by human interaction mainly by vision and sound, the use of hand gestures is one of most effective and productive routes in Human-Computer Interaction (HCI) [4].

There are three fundamental sorts of sensors which are capable for detecting hand gestures: mount based sensors, multi-touch screen sensors and vision based sensors. Accelerometers or gyros are mainly used to capture the movement of hands and fingers [5]. Multi-touch screen sensors "[6], [7]" are appropriate for cell phones yet restrict the separation amongst users and computers. Moreover, there are some advantages of vision based sensors "[8], [9], [10]" since they can be uncomfortable than the mounted sensors due to no physical contact with users. Vision based sensors also provide much larger working distance than multitouch screen sensors. However, the computational complexity is quite high for conventional vision based hand detection and tracking [11]. To handle this challenge, colored markers or data gloves have been utilized to simplify the vision tasks [12]. Considered as a kind of human motion [13] in our daily life, hand gestures have been reviewed by human motion analysis "[14], [15, [16]" which pays more attention to the full-body human poses and activities [17].

The main purpose of this paper is to develop an improvised hand gesture detecting system by the use of mathematical equations and formulas. For communication system, human can use both verbal and gesture methods. To utilize the gesture method, digital images are one of the most common and convenient ways to transmit information. To extract the information containing in an image, techniques like storage capability, processing, transmission, reorganization and interpretation are required. Digital image processing enables us to convert an image into the matrix format. After capturing an image, it goes through the process of algorithm applications. Each image should be evaluated with regard to its general characteristics such as noise, blur, brightness and contrast, background intensity variations and general pixel value distribution. Measuring these parameters, a suitable classifier will be constructed and can be used for further processing of that particular image.

In this paper we also provided an idea of constructing a Haar feature based classifier which will work on the basis of genetic algorithm. A Haar-like classifier is utilized amid the instating of the framework to acquire the user's hand color. Different from above literatures giving broad and general gesture recognition reports, we mainly focus on giving an effective survey on 3D hand motion recognition in four perspectives: 3D hand modeling, basic sensors capable for detecting hand gesture, continuous hand gesture recognition and related applications of hand gesture.

2 Research Methodology

2.1 Gesture Detection in OpenCV

For Detection & recognition purpose, we designed an ideal model with which an efficient detection algorithm construction is possible. To implement such, we chose the software OpenCV which has inbuilt functions mainly aimed at real time image processing. Now it has several hundreds of image processing and computer vision algorithms which make developing advanced computer vision applications easy and efficient.

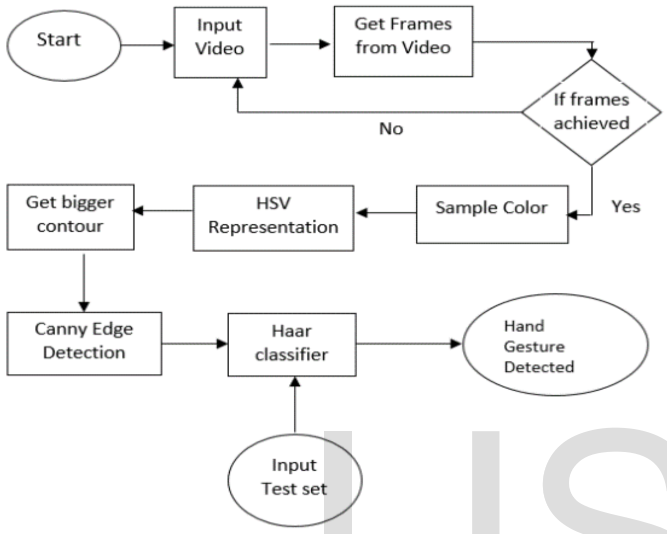
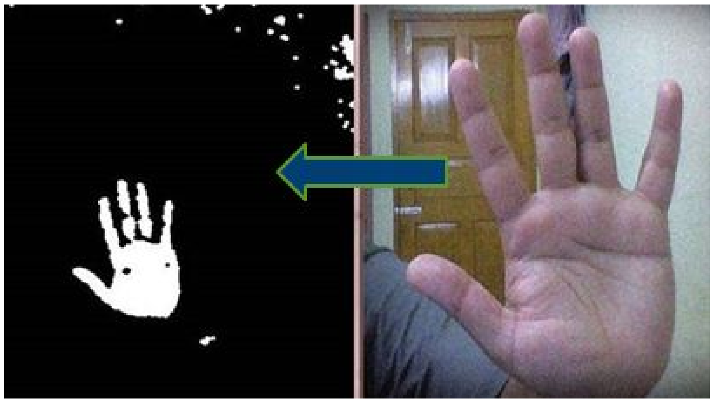


Figure 1: Proposed model for hand gesture detection & recognition.

According to the proposed model, at first we performed the HSV transformation of the input video feed, then we had done the color sampling which is very important for detection purpose because we need to convert the image to such that it represents binary data formulation through each pixel. Then we detected the biggest contour from that frame & eliminated other contours. After that, we implied Canny's edge detection



algorithm to detect the edges.

Figure 2: HSV representation of image after determining the biggest contour.

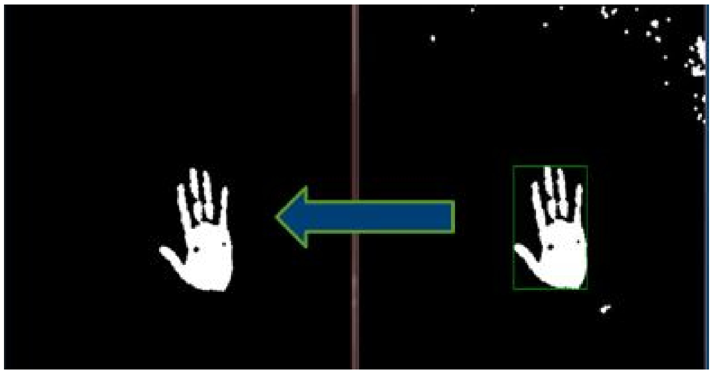


Figure 3: Reduction of background noise after determing the biggest contour.

2.2 Developing Haar Classifier

Haar features can easily be scaled by increasing or the size of pixel group being examined. This allows features to detect features on specific gestures. The variances of contrasts between the pixel groups are used to determine relative light & dark areas [1]. The feature value f of any individual Haar feature with k rectangles can be represented as following equation [2]-

$$I = \sum_{i=0}^{K} W^i \ \mu^i \qquad (1)$$

Haar like feature based classifier provides both high accuracy & speed. It needs less microprocessor instructions & has much less false detections [3]. Use of integral images causes high speed of evaluation while rectangular property of the haar like features characterize non symmetrical properties of Gesture appearance [4], so it is perfect for Gesture detection procedure.

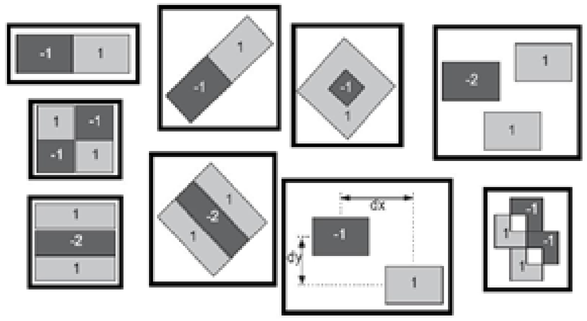


Figure 4: Haar like features shown in the figure with default weighted values of the corresponding rectangles.

The rotated integral image is calculated by calculating the sum of the pixels' intensity values which are located at forty-five degree angle to the left & above for the x value & below for the y value [5].

Animals, human beings included, behave based on environmental perceptions, which are made quickly and often unconsciously. If the decision was wrong, the embedded learning mechanism corrects the decision criterion instantly. Using this principle based on an animal's flexible recognition adaptability, gesture recognition techniques follow the same strategy. In the traditional approach, we tried to obtain more and more detailed information at the beginning and then took decisions based on statistical data processing.

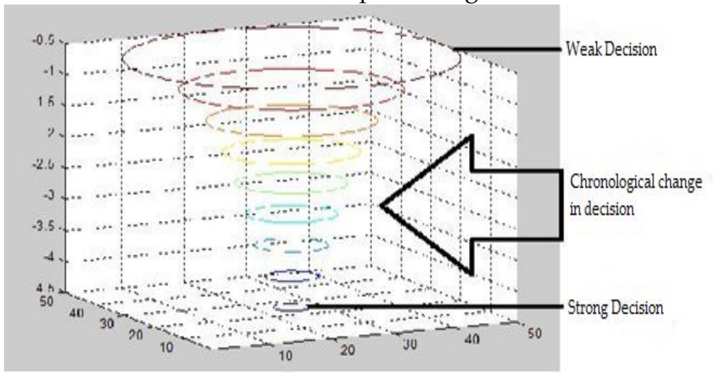


Figure 5: Deducing strong classifier from different iterative states presumed

Training a weak classifier involves determining the following three parameters that give the least error on the training set of object and clutter class images [6]:

- The threshold value θ,
- The polarity term p,
- The weight to be assigned to each rectangle of Haar- like feature 'w'.

3 Algorithm Design

3.1 Brute Force Search Algorithm

Brute Force Search, also known as Exhaustive Search is a very commonly used algorithm in which all the possible test data among a test set is verified one by one whether they match with the given statement or not.

Basic Algorithm can described as below [7]-

- First: Generate a First test solution for P.
- Next: Generate the next test for P after the current one
- Valid: Check whether test C is a solution for P.

Output: use the solution C of P as appropriate to the application.

3.1.1 Training a Single Weak Classifier Using Brute Force Search

```
Input: Training images: $x^i$, i ε {1, … , n}
Input: Training labels: $y^i$ ε {-1, 1}, i ε {1, … , n}
Input: Weights for each training image $z^i$ ε R, i ε {1, … , n}
Input: Weak classifier with k-rectangle Haar like feature: h
Input: Possible weighted values: d
    [1] Set Error of the weak classifier, $\bar{e}_{min} = \infty$.
    [2] For t = 1 to ($^{d}{}_{k}$) Do
    • Set weights ($w^{**}$) to be assigned to the rectangles of h.
    • Find $\theta^{**}$ & $p^{**}$ that minimize the training error ē.
            $[\theta^{**}, p^{**}]$ = arg min ē. Where,
            $\bar{e} = \frac{1}{2}\Sigma_{i=1}^{n} z^i \left|hx^i - y^i\right|$
    • If $\bar{e} < \bar{e}_{min}$ then
      $w^* = w^{**}$, $\theta^* = \theta^{**}$, $p^* = p^{**}$, $\bar{e}_{min} = \bar{e}$
Output: Trained weak classifier: h → $w = w^*$, h → $\theta = \theta^*$,
h→ $p = p^*$
Output: Error of the classifier: $\bar{e}_{min}$
```

3.2 Genetic Algorithm

Given a specific detection problem to solve, the input of GA is a set of solution to that problem which we call Genomes, a quantitative matric element, Fitness Function, which is used to evaluate the quality of the Genomes. Better decision is achieved through different iteration steps by improving the Genomes through genetic operations named as Crossover & Mutation [8]. It seems that as GA comes out with solution after many iteration steps, this algorithm is computationally much expensive; hence it is a much faster technique than the brute force search. So it seems that the proper combination of GA with 30 genomes & 100 Generations & The brute force search algorithm comes out with a greater success rate [9].

In this paper, we are interested in developing a specific approach to train our weak classifier. This approach should work much better than only with an exhaustive search algorithm over classifier parametric space.

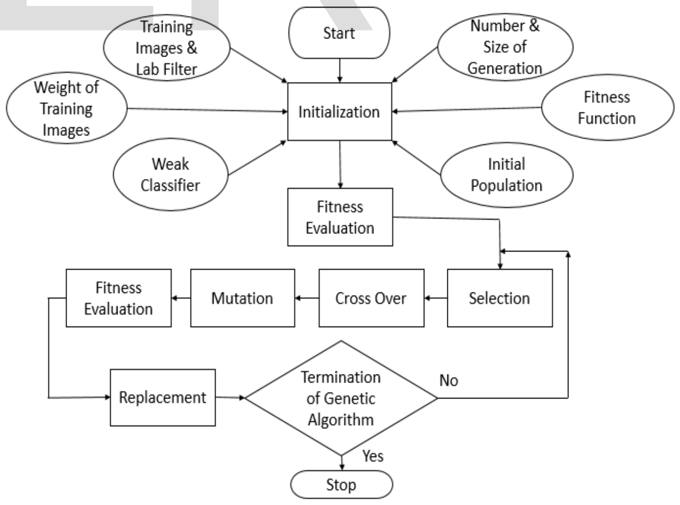


Figure 6: Proposed model for genetic algorithm.

• Sadman Shahriar Alam is currently working as a lecturer at Port City International University, Dept. of EEE, Chittagong, Bangladesh, PH-+8801790742747.
E-mail: sadmanshahriar23@gmail.com
• Akib Jayed Islam is currently working as a lecturer at Port City International University, Dept. of EEE, Chittagong, Bangladesh.
• Nahid Nasrin is currently working as a chairman at Port City International University, Dept. of EEE, Chittagong, Bangladesh.
• Khandoker Tanjim Ahammad is currently working as a lecturer at Port City International University, Chittagong, Bangladesh.

A population is sum of test sets consists of n individuals where N generation of iteration is going to be performed, N is freely chosen by the designer of GAs. Every individual genome contains l number of genes which is ranged from 1 to g. So genome G_t^l will be "[10], [11]"

$$G_t^l = \text{gene } 1 \times \text{gene } 2 \times \text{gene } 3 \times \ldots \times \text{gene } (g - 1) \times \text{gene } (g) \qquad (2)$$

Training data fitness functions are used in gradient descent training methods such as error back propagation for training neural networks & weak classifiers.

In our problem of finding optimal weights to rectangles, the genome is an array, representing the weights to be assigned to the rectangles. A valid genome for a Haar-like feature with k rectangles would be an array of length k, whose elements can take real values. For a given genome and a Haar-like feature, our fitness function builds a weak classifier and returns the error made by the weak classifier on training examples [12]. The lower the error returned by the fitness function, the better the genome represents the desired solution, But we have to keep in mind that we cannot achieve "0" error for a real time test set, in that case the generation number would be infinite, which would make the process huge time consuming the system will not be efficient. For this reason, we have to accept a certain amount of error.

3.2.1 GA Algorithm

Input: Training images: x^i, i ϵ {1, … , n}

Input: Training labels: y^i ϵ {-1, 1}, i ϵ {1, … , n}

Input: Weights for each training image z^i ϵ R, i ϵ {1, … , n}

Input: Weak classifier with k-rectangle Haar like feature: h

Input: Number of generations: N

Input: Size of Generations: m

Input: For targeted error $\bar{e}_{target}$ & No error $\bar{e}_0$, Fitness function f_{avg} & f_{max} determined.

Input: Initial population of genomes G_1^l, l ϵ {1, … , g}

```
1.  Set error of the weak classifier, $\bar{e}_{min} = \infty$.
2.  for t=1 to N do {
3.  for l=1 to g do {
4.  Set the weights to be assigned to the rectangle of current test set h
        1. $h \rightarrow w = G_t^l$
5.  Find θ** & p** that minimize the training error ē.
        [ϴ**, p**] = arg min ē. Where,
        $\bar{e} = \frac{1}{2}\Sigma_{i=1}^{n} z^i \left| hx^i - y^i \right|$
6.  If $\bar{e} < \bar{e}_{min}$ then
        $w^* = w^{**}, \theta^* = \theta^{**}, p^* = p^{**}, \bar{e}_{min} = \bar{e}$
7.  The probability of adaptive crossover $P_c$ calculated by
        $Pc = \begin{cases} \frac{a_1}{f_{max} - f_{avg}}(f_{max} - f), & f \geq f_{avg} \\ a_2, & f < f_{avg} \end{cases}$
8.  If $R < P_c$, (R= Expected ACP for), the operation of crossover.
9.  Replace the worst genomes with the new genomes.
10. End
11. Output: Trained weak classifier: h → w = w*, h → θ= θ*, h → p = p*
12. Output: Error of the classifier: $\bar{e}_{target}$
```

3.3 Fisher's Linear Discriminant Analysis Algorithm

FLDA provides a closed-form solution to find a linear Classifier that best separates two or more classes. Although it provides an optimal solution only when the classes are Gaussian with equal covariance, it is reasonably quick and provides good approximations to the optimal solution in the general case [13]. FLDA outputs a linear projection vector whose slope corresponds to the optimal weights assigned to the rectangles of the Haar-like feature. Once the optimal weights are found, the optimal values for threshold θ^* and polarity term p^* are found by searching exhaustively over all possible solutions.

3.3.1 Training a Single Weak Classifier Using FLDA

Input: Training images: x^i, i ϵ {1, … , n}
Input: Training labels: y^i ϵ {-1, 1}, i ϵ {1, … , n}
Input: Weights for each training image z^i ϵ R, i ϵ {1, … , n}
Input: Weak classifier to be trained:
h begin{
Evaluate the Haar-like feature in h over all object and clutter training images. Each evaluation gives ak-dimensional vector $\mu^{(i)}$. Compute between-class scatter: m_1-m_2 where m_1 & m_2 are the k dimensional means of measurements made from n_0 object class and n_c clutter class training images respectively.

$$m_1 = \sum_{\forall i, y^i = 1} \frac{\mu^i}{n_o}, \; m_2 = \sum_{\forall i, y^i = -1} \frac{\mu^i}{m_c}$$

Compute within class scatter: $S = S_1 + S_2$

$$S_1 = \sum_{(\forall i, y^i = -1)} (\mu^i - m_1)(\mu^i - m_1)^t$$

$$S_2 = \sum_{(\forall i, y^i = -1)} (\mu^i - m_2)(\mu^i - m_2)^t$$

Compute optimal weights

$$w^* = S^{(-1)}(m_1 - m_2)$$

Find θ** & p** that minimize the training error ē.
[θ**, p**] = arg min ē. Where,

$$\bar{e} = \frac{1}{2}\Sigma_{i=1}^{n} z^i \left| hx^i - y^i \right|$$

}
Output: Trained weak classifier: h → w = w∗, h →□= □∗, h → p = p*
Output: Error of the classifier: $\bar{e}_{target}$

3.4 Boosting Algorithm (ADABOOST)

Boosting is a method of finding a highly accurate hypothesis by combining many "weak" hypotheses, each of which is only moderately accurate. It manipulates the training examples to generate multiple hypotheses. ADABOOST iteratively generates a robust final hypothesis by giving increased weight to mis-classified training samples from previous learning rounds [14]. In AdaBoost, a key step is choosing a new distribution on the training examples based on the old distribution and the errors made by the present weak hypothesis [15].

The overall procedure of the boosting GA classifier is depicted in figure 6. The function of boosting is to repeatedly apply a weak learning algorithm on various distributions of the training data and to aggregate the individual classifiers into a single overall classifier [16]. Initially, all training examples are uniformly weighted. After each iteration, the distribution of training examples is changed, based on the error that the current classifier exhibits on the training set. The weight $w^{(i)}$ specifies the relative importance of the i-th training example.

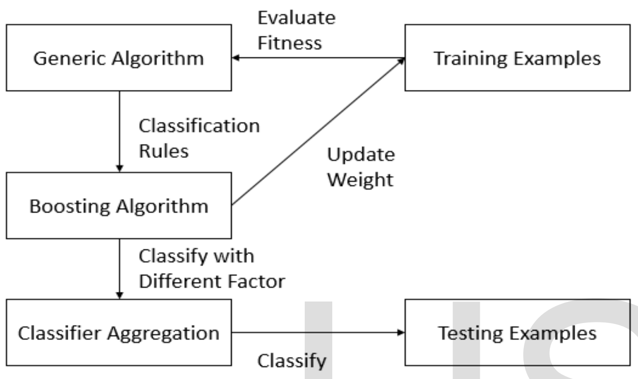


Figure 7: Procedure of boosting GA algorithm.

3.4.1 AdaBoost Algorithm

Input: Sequence of N examples <(x_1,y_1), … ,(x_n,y_n)> with labels $y_i \in Y = \{1, \dots, k\}$;

Input: Distribution D over the examples;

Input: Weak classifier algorithm;

Input: number of iteration T.

Initialize the weight vector: w_i^l = D(i) for i=1,……,N

Do for t=1,2,…………,T_t

1. Set $P^t = W^t / (\Sigma_{(i=1)}^N w_i^t)$

2. Call weak classifier, providing it with the distribution P^t, get back a hypothesis $h_t : X \rightarrow Y$

3. Calculate the error of ,

If $\varepsilon_t > 1/2$ the set T=t-1 and abort loop.

4. Set $\beta_t = \grave{o}_t / 1 - \grave{o}_t$

5. Set the new weights vector to be $w^{(t+1)} = w^t \beta^{(1|h_t x_i \neq y_i|)}$

6. Output: h_f (x)= arg max $\Sigma_{t=1}^T (log 1/\beta_t)|h_t(x) = y|$

3.5 Building a Rejection Classifier

The rejection cascade is built according to algorithm 3.5, the algorithm takes in a database of object and clutter class images $x^{(i)}$, i=1, ... ,n, their labels $y^{(i)}$, i =1,...,n and a set of weak classifiers $h^{(j)}$, j =1,...,m. Given the training data, the algorithm iterates M times to construct the rejection cascade, where M is the desired number of nodes. In each iteration, a node $H^{(u)}(x)$ is built according to Algorithm 4. The clutter training images that were correctly classified by the current node are not considered to build the next node. This ensures that the next node to be built concentrates on rejecting the clutter images that were misclassified by the previously built nodes.

```
Input: Estimated number of nodes in the cascade: M
Input: Training images: $x^i$, i ∈ {1, … , n}
Input: Training labels: $y^i$ ∈ {-1,1}, i ∈ {1, … , n}
Input: Set of weak classifiers: $\mathbf{h} = \{h^{(j)}\}$, i ∈{1, ... ,m}

begin{
    for u=1 to M do
    Build node $H^{(u)}$ according to the adaboost algorithm.
Remove x(i) if $H^{(u)}(\mathbf{x}^{(i)}) = 0$ and $y^i = -1$.
end}
```

Output: Rejection cascade. H(t) of strong classifier where t is a test image:

4 Future Scope in 3D Hand Modeling

4.1 Classification of 3D Modeling

3D hand modeling and tracking estimate the articulated hand poses and motions [22]. These are key technologies in many HCI applications such as robot surgery, virtual keyboard and sign language recognition [23]. According to the taxonomy proposed by [24], hand pose estimation approaches can be classified in two categories: discriminative approaches and generative approaches. Moreover, the hybrid 3D modeling which combines discriminative modeling and generative modeling has been also proposed. Next, we will review various 3D modeling approaches in the context of 3D hand pose estimation and tracking with respect to these three categories.

4.2 Hand Modeling from 2D Images

De La Gorce *et al.* proposed generative approaches [25, 26] to estimate 3D hand poses from monocular 2D images. In [27], they modeled the hand by an articulated kinematic tree with 28 DoF. The original model was rendered by ellipsoids and polyhedral to form the hand surface model. Then, the hand silhouette could be extracted by differentiating the parametrized hand surface model

4.3 Hand Trajectory Gesture Recognition

In contrast to the static hand gesture recognition which works on hand shapes, the hand trajectory gesture recognition considers the sequential data of hand trajectory and explores the temporal character of hand motion. In this section, we will survey the 3D hand trajectory gesture recognition approaches.

4.4 Hand Trajectory Gesture Recognition

For general hand trajectory gesture recognition, we usually assume the hand gesture video clip has been well segmented in the temporal domain. The focus is to recognize the single meaningful hand gesture in the video. However, the practical HCI applications require the continuous hand gesture recognition in video streams. This means that both the spatial segmentation and temporal segmentation are necessary to recognize the sequential hand gestures.

4.4 Depth Sensors

Various 3D depth sensing technologies have been well reviewed by previous literatures [15], [16], [21], which mainly introduce the working mechanisms of different depth sensors1. In this survey, we focus on three popular depth sensors for hand gesture recognition. In next subsections, we will briefly review Kinect, Leap Motion and Time of Flight sensors.

5 Application and typical systems

The 3D hand gestures recognition approaches are mainly used in five application domains: sign language recognition, virtual manipulation, daily assistance, palm verification, gaming and the emerging human-robot interaction. Sign language recognition with hand gestures has been investigated for the ASL as mentioned in previous sections. For disable people who are deaf mute, a recognition system of sign language can greatly help them to keep in touch with others. Virtual manipulation is the popular application of 3D hand gesture recognition due to its natural user interface for HCI tasks. Fraunhofer FIT developed the 3D Gesture Based Interaction System7 which is the noncontact gesture and finger recognition system. The system detects hand and finger positions in real-time and translates these into appropriate interaction commands. Daily assistance using hand gestures can help older people to perform Activities of Daily Living (ADLs) such as hand-washing [18]. The important senior gestures such as eating and drinking can also be monitored [19]. Palm verification is a key biometric technology for many security applications. The shape of the hand can be easily captured in a relatively user friendly manner by using 2D/3D cameras thus acceptable by the public. Amayeh *et al.* [168] proposed a component based approaches to hand-based verification and identification. Their approaches utilized a 2D camera plus a lighting table to decomposite the hand silhouette into different regions corresponding to the back of the palm and the fingers. Gaming with interactive hand gestures have been significantly promoted by next-gen game consoles. Sony's Play Station and Nintendo's Wii are equipped with handy controllers (PS Move and Wiimote, respectively) for the player's hand tracking. Human-Robot Interaction (HRI) is the most important function for the emerging social robot which is able to interact with people using the natural gestures [20]. Hand gesture based interface offers a way to enable human to interact with robots more easily and efficiently [21].

6 Conclusion

In this paper, we introduced some concepts to improve a gesture detection system from many aspects. The application of genetic algorithm makes the system more smart & intelligent; whereas the improvisation of the test set preprocessing makes it easygoing for the system. We propose assigning proper optimal weighted value of Haar like feature so that weak classifiers developed based on them comes out with best possible decision. The optimal weights were computed and classified by using Brute Force Search Algorithm, Genetic Algorithm & Fisher's Linear Discriminant Analysis Algorithm. We suggest GMM & Adaptive Skin Color segmentation for skin color detection, fuzzy set II for Auto thresholding & semi definite programming (SDP) and objective functions for kernel matrix determination for time varying input and our logic indicates that by using these proposed features, better gesture detectors, both in term of accuracy and speed will be constructed. One of the major challenges is the online recognition of 3D hand gestures. The absence of explicit begin/end hints in practical scenarios will degrade the performance of traditional static and trajectory approaches. Hence, the continuous hand gesture recognition will attracted more attention due to its applicability. Other challenges include different meanings of similar hand gestures which need to be distinguished by fine-grained gesture recognition. In the future, we expect tinier finger gestures can be well recognized leveraging more accurate depth sensors. Finally, we envision the booming of intelligent products using 3D hand gesture recognition for human-robot interaction purpose. Most of existed 3D hand gesture recognition works employ depth sensors with fixed position. However, users may move freely and disappear during the interaction with robots. We believe there will be much research room of interactive hand gesture recognition for human-robot interaction.

7 Appendices

7.1 Code for Edge Detection

```
#include "opencv\cv.h"
#include "opencv\highgui.h"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdlib.h>
#include <stdio.h>
using namespace cv;

/// Global variables

Mat src, src_gray;
Mat dst, detected_edges;

int edgeThresh = 1;
int  lowThreshold ;

int const max_lowThreshold = 100;
int ratio = 3;
int kernel_size = 3;
char* window_name = "Edge Map";

/**
 * @function CannyThreshold
 * @brief Trackbar callback - Canny thresholds input with a
ratio 1:3
```

```
 */
void CannyThreshold(int, void*)
{
  /// Reduce noise with a kernel 3x3
  blur( src_gray, detected_edges, Size(3,3) );

  /// Canny detector
  Canny( detected_edges, detected_edges, lowThreshold,
lowThreshold*ratio, kernel_size );

  /// Using Canny's output as a mask, we display our result
  dst = Scalar::all(0);

  src.copyTo( dst, detected_edges);
  imshow( window_name, dst );
 }

/** @function main */
int main( int argc, char** argv )
{
  /// Load an image
  src = imread( "hand1.jpg" );

  if( !src.data )
  { return -1; }

  /// Create a matrix of the same type and size as src (for dst)
  dst.create( src.size(), src.type() );

  /// Convert the image to grayscale
  cvtColor( src, src_gray, CV_BGR2GRAY );

  /// Create a window
  namedWindow( window_name, CV_WINDOW_AUTOSIZE
);

  /// Create a Trackbar for user to enter threshold
  createTrackbar( "Min Threshold:", window_name,
&lowThreshold, max_lowThreshold, CannyThreshold );

  /// Show the image
  CannyThreshold(0, 0);

  /// Wait until user exit program by pressing a key
  waitKey(0);
  return 0;
}
```

REFERENCES

[1] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," ACM Comput. Surv. vol. 43, no. 3, pp. 16:1–16:43, 2011.

[2] R. Lefevre, *Rude Hand Gestures of the World: A Guide to Offending without Words*. San Francisco: Chronicle Books, 2011

[3] S. D. Kelly, S. M. Manning, and S. Rodak, "Gesture gives a hand to language and learning: Perspectives from cognitive neuroscience, developmental psychology and education," *Language and Linguistics Compass*, vol. 2, no. 4, pp. 569–588, 2008.

[4] H. Cheng, Z. Dai, and Z. Liu, "Image-to-class dynamic time warping for 3D hand gesture recognition," in IEEE ICME, 2013.

[5] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang, "A framework for hand gesture recognition based on accelerometer and emg sensors," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 6, pp. 1064–1076, 2011.

[6] D. Kammer, J. Wojdziak, M. Keck, R. Groh, and S. Taranko, "Towards a formalization of multi-touch gestures," in *ACM International Conference on Interactive Tabletops and Surfaces*, 2010.

[7] E. Hoggan, J. Williamson, A. Oulasvirta, M. Nacenta, P. O. Kristensson, and A. Lehti¨o, "Multi-touch rotation gestures: Performance and ergonomics," in *ACM SIGCHI*, 2013.

[8] X. Zabulis, H. Baltzakis, and A. Argyros, "Vision-based hand gesture recognition for human-computer interaction," 2009, Chapter 34, The Universal Access Handbook.

[9] P. Buehler, M. Everingham, D. Huttenlocher, and A. Zisserman, "Upper body detection and tracking in extended signing sequences," *International Journal of Computer Vision*, vol. 95, no. 2, pp. 180–197, 2011.

[10] S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, pp. 1–54, 2012.

[11] X. Suau, J. Ruiz-Hidalgo, and J. Casas, "Real-time head and hand tracking based on 2.5D data," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 575–585, 2012.

[12] R. Y. Wang and J. Popovic, "Real-time hand-tracking with a color glove," in *ACM SIGGRAPH*, 2009.

[13] R. Poppe, "Vision-based human motion analysis: An overview," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 4–18, 2007.

[14] M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, and J. Gall, "A survey on human motion analysis from depth data," *Lecture Notes in Computer Science*, vol. 8200, pp. 149–187, 2013.

[15] L. Chen, H. Wei, and J. Ferryman, "A survey of human motion analysis using depth imagery," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1995–2006, 2013.

[16] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with Microsoft Kinect sensor: A review," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.

[17] Y. Song, J. Tang, F. Liu, and S. Yan, "Body surface context: A new robust feature for action recognition from depth videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 6, pp. 952–964, 2014.