Displays 57 (2019) 18–33

Contents lists available at ScienceDirect

Displays

journal homepage: www.elsevier.com/locate/displa

Visual rendering of shapes on 2D display devices guided by hand gestures☆

Abhik Singla[a], Partha Pratim Roy[b], Debi Prosad Dogra[c]

[a] Department of Electronics and Communication Engineering, NIT Kurukshetra, India
[b] Department of Computer Science and Engineering, IIT Roorkee, India
[c] School of Electrical Sciences, IIT Bhubaneswar, India

ARTICLE INFO

Keywords:
Gesture recognition
Shape rendering
Shape matching
3D rendering

ABSTRACT

Designing of touchless user interface is gaining popularity in various contexts. Users can interact with electronic devices using such interfaces even when their hands are dirty or non-conductive. Also, users with partial physical disability can interact with electronic devices with the help of touchless interfaces. In this paper, we propose a Leap Motion controller-based methodology to facilitate rendering of 2D and 3D shapes on display devices. The proposed method tracks finger movements while users perform natural gestures within the field of view of the motion sensor. Then, trajectories are analyzed to extract extended Npen++ features in 3D. These features capture finger movements during the gestures and they are fed to unidirectional left-to-right Hidden Markov Model (HMM) for training. A one-to-one mapping between gestures and shapes, is proposed. Finally, the shapes corresponding to these gestures are rendered over the display using a typical MuPad supported interface. We have created a dataset of 5400 samples recorded by 10 volunteers. Our dataset contains 18 geometric and 18 non-geometric shapes such as “circle”, “rectangle”, “flower”, “cone”, “sphere”, etc. The proposed method has achieved 92.87% accuracy using a 5-fold cross validation scheme. Experiments reveal that the extended 3D features perform better than the existing 3D features when applied for shape representation and classification. The method can be used for developing diverse HCI applications suitable for smart display devices.

1. Introduction

Touchless interactions with electronic display devices have their own benefits. For example, such interfaces can allow surgeons to interact with machines through gestures during surgical operations [1]. They can facilitate the doctors to navigate via complicated and delicate instrument panels to find control buttons during surgery. Moreover, every electronic device may not be equipped with a touch-enabled graphical display interface. Therefore, touchless interfaces can be thought of viable alternatives. However, there are a few challenges that need to be addressed before such interfaces can replace conventional touch-enabled or tactile-enabled display devices.

The question is: Why do away with tactile buttons and touch screens when they are well-established? The answer can be, ”Computers are no longer thought to be used only at homes or on office desks”. These days people travel everywhere with their smart handsets, personal media players, e-books, and tablets. People carry such devices in restaurants, gyms, coffee bars, airport terminals, bus stops, and even inside lavatories. In such diverse operating environments, hands of the users are often occupied, dirty, sweaty or covered with other items. Moreover, it may not be suitable to operate the device through touch screen or display in these conditions. Suppose a user is reading an e-book at the gym while on a treadmill. It may be easier to swipe across the device with a touchless gesture rather than physically contacting a touch screen or hunting down a small button to turn the page.

Gestures and facial expressions are alternate ways to interact with systems that do not support touch or tactile interfaces. Gestures are nothing but meaningful expressions of humans using various body parts. Thus, gesture recognition is termed as understanding the meaning of expressions and a survey on recent developments on this topic can be found in the work of Mitra et al. [2]. Likewise, “thumbs up” representing “best of luck” or “waving our hands” to say “hi” to someone, are examples of a few such gestures. When we interact with machines or computers, raw gestures need to be presented in machine understandable format. Therefore, automatic gesture recognition is being studied in-depth since last 2–3 decades [3]. It has opened-up ample scopes to design interesting applications aimed for human computer interaction [4,5], serious gaming [6], robotics [7], automatic sign language interpretation [8], designing of intelligent machines for serious gaming [9], etc.

Hand gesture recognition in 3D is well studied and some of the recent developments in this field can be found in [10,11]. It has wide

☆ This paper has been recommended for acceptance by Richard H.Y. So.

E-mail addresses: proy.fcs@iitr.ac.in (P.P. Roy), dpdogra@iitbbs.ac.in (D.P. Dogra).

https://doi.org/10.1016/j.displa.2019.03.001
Received 27 November 2017; Received in revised form 22 March 2019; Accepted 24 March 2019
Available online 30 March 2019
0141-9382/

range of applications in virtual reality [12], sign language recognition [8], serious gaming, smart interaction [13], and human computer interaction (HCI) [6]. Gestures can be captured using visible light camera, IR camera or specially designed sensor attachments. Out of these three, recording of gestures through visible light camera is probably the most popular choice. However, vision-based freehand gesture recognition algorithms often suffer from various environmental noises which includes illumination variation and background clutter [14]. Self occlusion of the fingers is an additional challenge. Therefore, applications that use visible light camera for gesture recognition, need controlled or supervised arrangements [15].

Researchers have also used specially designed hardware or software-hardware combinations for designing freehand gesture recognition systems such as wearable glove fitted with inertial measurement unit (IMU) sensors containing accelerometer, gyroscope, and magnetometer [16]. Often such arrangements are preferred over vision-based systems since the signals acquired by wearable sensors are less noisy as compared to signals recorded using normal cameras [2,17]. However, wearable sensors have their own disadvantages such as higher price over optical sensors, large calibration overhead, and more importantly, they may be inconvenient to the end users.

Thanks to the recent development of low-cost and ready-to-use sensors such as Microsoft's Kinect, Intel's RealSense or Leap Motion device, freehand gesture recognition is becoming easier [18–21]. These sensors usually provide a three-dimensional point cloud data that can be processed to understand the underlying gestures. Kinect has been designed for the applications to interpret movements of the whole body [22]. Due to a low-resolution depth map generation (only 640 × 480) of the whole body, it works reasonably well to track large objects (e.g. full human body). Intel RealSense is another sensor that came into existence lately (2014). It can be used for gesture recognition, eye gaze tracking [23], etc. Leap motion controller first came into use in the year of 2012. Since then, it has been used in human-computer interface designing [5], rehabilitation [19], etc.

It has been reported by the manufacturer that the theoretical accuracy of Leap motion is 0.01 mm (can track the movement of both hands and all 10 fingers with up to 1/100th millimeter accuracy and no visible latency) [24]. However, Weichert et al. [25] have shown that the standard deviation remains below 0.7 mm per axis when moving to discrete positions on a path. Therefore, it is not possible to achieve the theoretical accuracy under real conditions, though a high precision (an overall average accuracy of 0.7 mm) with regard to gesture-based user interfaces, can be achieved.

1.1. Motivation of the work

It is believed that, touchless navigation, designing of sign language interface, 3D air painting, augmented reality, serious gaming, physical rehabilitation, consumer electronics interfaces, interactive live performance or real-time pencil rendering, are going to be more fun and interesting in coming days.

Using Leap Motion's API, gestures can be turned into computer commands. Vinayak et al. [26] have proposed a method of 3D modeling that is referred to as "Shape-It-Up" using Kinect. A recently released free 3D modeling app popularly known as "Freeform" [27] applies a gesture control interface for clay-like virtual modeling. However, the actual interface of "Freeform" is similar to the interface of "Sculptris" or "Leopoly" [28]. 3D analysis of gestures can allow manipulation of virtual materials as reported by Vinayak et al. [26]. They emphasize that video game developers, design engineers, and architects will benefit the most from freehand gesture interfaces.

However, mapping of regular or irregular shapes with gestures must be accurate. Otherwise, they cannot be used in high-level tasks such as virtual clay modeling or interactive gaming. Also, the setups need to be simple. This has motivated us to adopt an easy-to-use setup such as Leap Motion and develop an acceptable methodology of freehand gesture recognition. In our study, we have chosen 36 shapes for freehand drawing using hand gestures. These shapes have been selected carefully to represent wide range of variations. For example, 21 out of these 36 shapes can be drawn using single finger movements and the rest can be drawn using movements of multiple fingers. Our dataset covers regular geometrical shapes such as "cone", "sphere", "cube", "rectangle", "triangle", "circle", "cube", "cylinder", "hemisphere", or "pyramid". In addition to that, we have kept direction signs such as "left", "right", "up", and "down" in the dataset. A few commonly known irregular non-geometric shapes such as "house", "heart", "flower", or "moon" have also been included. Commonly used symbols such as "∗", "@" or "+" available on standard keyboards are also included in our dataset. Even we have kept a scientific symbol e.g. "ω" to make the dataset diverse. Though it is a challenging task to prepare an exhaustive set, however, we believe the selected shapes are diverse enough, thus can be used in wide range of applications.

1.2. Research objectives

Motivated by the aforementioned facts, we set the following research objectives while developing the system:

- We are interested in designing a highly-accurate system that can be used to recognize freehand gestures performed over the field of view of the sensor such that rendering of virtual objects on display can be done.
- Conceptualization of shapes can be achieved through natural process driven by instinct. However, rules to represent the concepts need to be specified. Thus, one of our research goals is to define a set of freehand gestures that humans usually apply while interacting with the outside world to closely represent regular or irregular 3D shapes representing common objects.

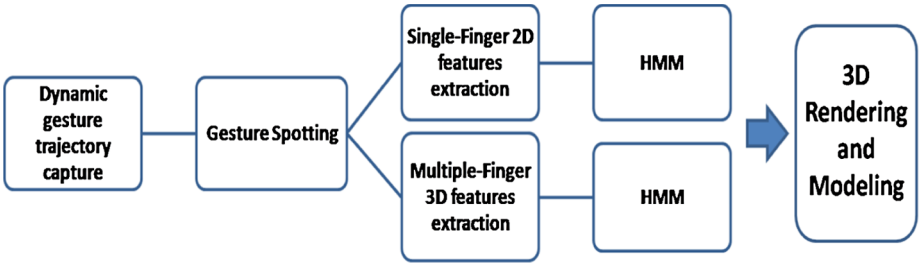


Fig. 1. Block diagram of the proposed shape retrieval and rendering method.

- Our final goal is to map these set of gestures with commonly known shapes such as “cube”, “bottle”, “hemisphere”, or “heart” and design an easy-to-use interface. Users can experience like playing with virtual clay. This has applications in interactive gaming, human computer interface or user interface designing.

1.3. Contributions of the paper

Block diagram of the proposed system is depicted in Fig. 1. Our main contributions are as follows:

- Designing of a gesture recognition system for mapping of stored regular/irregular and geometric/non-geometric shapes with corresponding gestures.
- Designed a GUI interface to retrieve and render shapes on display based on user’s free-hand gestures performed within the field of view of the sensor.
- Our final contribution is creation of a large gesture dataset comprises of 5400 gestures and making it publicly available to the research community.[1]

Rest of the paper is organized as follows. A report on state-of-the-art is presented in Section 2. Proposed methodology is presented in Section 3. Experiment results are presented in Section 4. Finally, we conclude in Section 5 by highlighting some of the possible future extensions of the present work.

2. Related works

Rendering of 3D shapes has many applications. For example, this can be used for designing therapeutic interfaces [29], interactive pedagogy [30,31], shape learning by kindergarten students [32], learning art and music [33], virtual and augmented reality applications [34], etc. Users can perform rendering and transformation of 3D shapes by means of interactive participation through natural gestures.

Lately, gesture based 3D shape creation and recognition has been a topic of research. The VIDEODESK system proposed by Krueger et al. [35] can be considered as the pioneering work in this field. They have designed a system that allows a user to control an object’s shape by using his/her own hands. Whereas utilization of both hands of the user is of good advantage, however, their system uses predefined points of a user’s hands. Thus, the system fails to take advantage of full expressive power of both hands. Researchers have shown that the accuracy can be improved using both hands [36]. Though the improved framework can assign different responsibility to each hand and the method suggests to use bi-manual actions than Krueger’s method, the object deformation is only controlled by position and orientation of both hands. Therefore, shape of the hand is not used in true sense. This has been improved by Nishino et al. [37]. They have shown that the representation of 3D objects can be improved through bi-manual actions. However, aforementioned methods assume palm and finger detection is accurate.

Visible light cameras have found ample applications in the various domains like plant disease identification [38], fight detection in hockey [39], gesture recognition, etc. However, they have certain disadvantages as compared to IR camera-based or sensor-based systems. For example, vision-based gesture recognition system proposed by Zariffa et al. [40] suffers from segmentation error. The method proposed by Chiang et al. [6] assumes a simple background to avoid segmentation error, which is not realistic. On the other hand, sensor-based systems are more accurate since the signals acquired by IMU sensor are less affected by variations in illumination or segmentation error [41].

Despite good accuracy of the sensor-based systems, they have certain drawbacks; (i) Contact-based systems are a burden to the users because often they feel uncomfortable with such artificial attachments [42] (ii) Some of the existing systems require external power through battery [43]. Therefore, contact-less vision-guided systems are preferred for such applications. Researchers have shown that Leap Motion can be successfully used for palm rehabilitation [19], upper limb rehabilitation [44], stroke rehabilitation [45], etc.

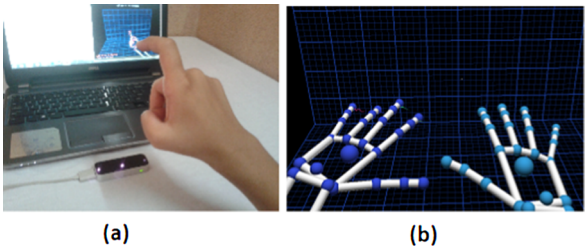


Fig. 2. (a) Leap motion setup used for recording and spotting single finger gestures. (b) Gestures are tracked in 3D interface using double hand.

2.1. Leap motion vs similar technologies

- Webster et al. [46] have measured Normalized Root Mean Squared Error (NRMSE) in position data captured by Kinect as compared to a research-grade OptiTrack motion capture system. As reported by the authors, NRMSE in position vary between 0.53 cm to 1.74 cm when initial calibration is conducted via the OptiTrack system. This is lower than the accuracy of Leap Motion. We have also observed that, it may be difficult to detect smaller body parts e.g. fingers or palm from the low-resolution images that are captured using Kinect. Therefore, it may not always be possible to represent complex articulations of fingers during freehand gestures captured using Kinect. On the other hand, Leap Motion provides real-time tracking of hand and finger movements in 3D. Also, Kinect is marginally expensive as compared to Leap Motion.
- To the best of our knowledge, there is no published research work that compares RealSense and Leap Motion barring a few online surveys. Therefore, it is difficult to quantify the difference between these two devices. However, we can emphasize that RealSense has not gone through a time tested evaluation process since the device is relatively new as compared to the Leap Motion. In addition to that, RealSense device uses RGB cameras. Thus, signals captured using such cameras are usually prone to illumination variation.
- Leap Motion’s inbuilt software, unlike “Leopoly” [28], allows users to change materials (clay, glass or plastic) and choose from more than one brush to work with. It also provides the user with an option of continuously rotating the ball to shape it as the user would do on a pottery wheel.

3. Proposed method

Conceptualization of shape by humans is a natural choice. Though we begin to learn shapes through various daily life activities, however, their conceptualization is a well defined scientific process.

Representation of shapes through natural gestures requires some level of training or prior information. A gesture can be performed in various ways, e.g. single-finger, multiple-finger, single-hand or multiple-hand, etc. In this work, we have tested our algorithm on 18 regular and 18 irregular shapes. Though we have hand-picked these shapes, however, the dictionary can be extended as long as we define a unique gesture for every shape. Similar approach has already been adopted by Horvath et al. [47] to represent shapes through natural gestures. For example, the authors have used both hands to perform gestures that represent typical 3D shapes such as “cylinder”, “cone”, “sphere”, “ellipsoid”, etc.

[1] https://drive.google.com/file/d/1qlEAthZ1m-eixy9btku-DxBP_Ou1cXts/view?usp=sharing.

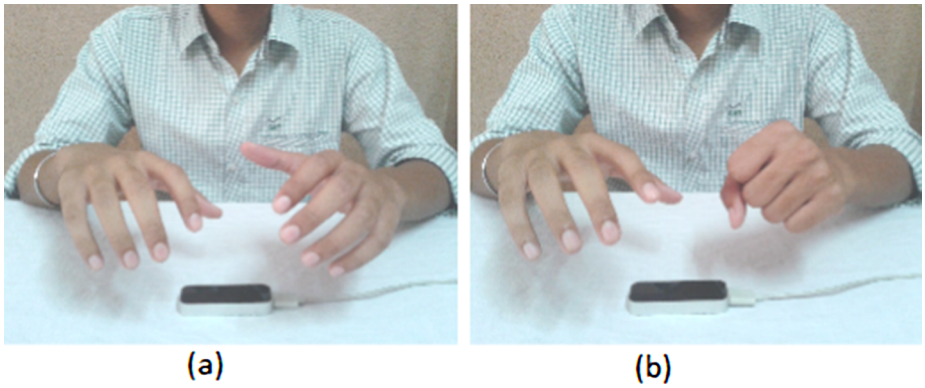


Fig. 3. Leap Motion-based setup used for spotting and recording of multiple fingers-based gestures, (a) capturing off (b) capturing on.

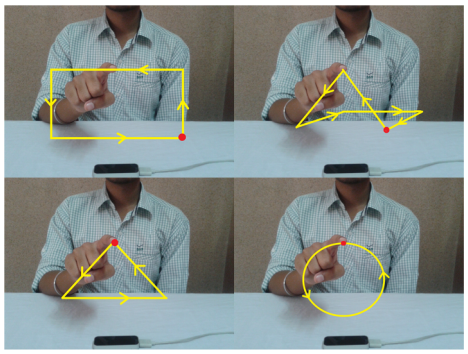


Fig. 4. Single finger-based gestures representing 2D and 3D shapes such as "rectangle", "pyramid", "triangle", and "circle", respectively. Red dots depict the starting point of the gesture. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

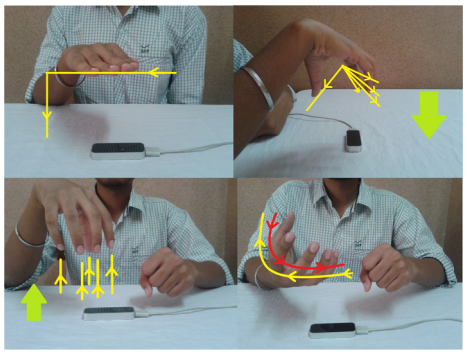


Fig. 5. Multiple fingers-based gesture representing "cuboid", "cone", "cylinder", and "sphere", respectively.

3.1. Single finger gesture recording

In our proposed work, single-finger gestures are captured by tracking the right hand's index-finger when the user draws a shape on the vertical plane or the plane perpendicular to the upper surface of the device. The interface captures the finger-tip position when the right hand is closed (except the index finger is out of the fist) in order to draw a shape as depicted in Fig. 2(a). Capturing is stopped to prevent noise when the user closes his/her right hand including the index finger. However, these rules are design-specific and can easily be modified as per the requirement of an application.

3.2. Multiple fingers gesture recording

A different heuristic has been used for multiple fingers-based gesture spotting and recognition. Capturing routine starts and records tip positions of the right hand fingers when a user closes the left hand and makes a fist. Right hand is used to perform gesture-related movements. The data acquisition stops when the user opens the left fist as depicted in Fig. 3(a). We track the positions and orientations of the fingers. Since the Leap Motion SDK handles occlusion with the help of a human hand model, we get accurate data. The heuristic can be modified as per the requirement of the application. In Fig. 2(b), we show samples of tracking for double-handed gestures.

Some of the shapes can be drawn using single finger-based gestures and others by using multiple fingers-based gestures. For example, "triangle", "circle", or "rectangle" being 2D shapes, can be drawn with single finger-based gestures using right hand index finger over a vertical plane. Even some of the 3D shapes can be drawn by single finger-based gestures with the help of isometric projections, As an example, to draw a solid shape like "pyramid", users can draw a "triangle" on the vertical plane and a "rectangle" on the horizontal plane in continuation as depicted in Fig. 4. This is acceptable since a user normally sees the frontal view of the "pyramid" as a "triangle" and top view as a "rectangle".

Natural way of performing multiple fingers-based gestures to render complex 3D shapes can be done as follows. The user can assume holding a solid sphere by the right hand's palm and gradually rotating the hand around its surface in circular direction and then back to the normal position while drawing a "sphere" gesture. A "cylinder" can be drawn by moving the fingers around the outer surface of a virtual cylinder (assuming there is a cylindrical object placed over the surface vertically) followed by moving the hand in upward direction. Similarly, the user can draw a "cone" by moving the fingers around the outer slant surface of the "cone" (assuming there is a cone-shaped object placed over the surface vertically) followed by moving the hand in downward direction from the top to the circular base. To draw a "cube", a user can assume the palm as a face of the cuboid and mimic it over top and right surfaces in a predefined order. Some possible depictions to draw a few of the above mentioned 3D shapes are presented in Fig. 5.

3.3. Naturalness and conceptualization of gestures

Naturalness of the gesture is very important while designing human-computer interfaces. Any random sequence of finger or hand movements may not be suitable. Thus, we have carried out a set of experiments to understand the naturalness of the gestures and their relevance with the conceptualized shapes. Five volunteers (not involved during recording of the experimental gesture dataset) were involved in this study. We recorded the videos of the gestures while they were performed by users. The volunteers were asked to carefully observe the video recordings of the gestures and label them as per their understanding. Out of all 36 gestures, 17 gestures were uniquely decoded by every volunteer and they understood the shapes. Out of the remaining

Table 1
Correct or incorrect conceptualization of gestures and their naturalness.

	Correctly Guessed 2D Gestures	Wrongly Guessed 2D Gestures	Correctly Guessed 3D Gestures	Wrongly Guessed 3D Gestures
Volunteer 1	11	10	9	6
Volunteer 2	13	8	7	8
Volunteer 3	11	10	8	7
Volunteer 4	12	9	9	6
Volunteer 5	10	11	8	7
Best Matching	10		7	

19 gestures, 9 were recognized correctly on a majority voting scheme. Outcome of this experiment is presented in Table 1.

3.4. Feature extraction

Feature selection and extraction, often requiring extensive domain knowledge, are essential steps in a gesture recognition system [48,49]. In our proposed method, we have taken the normalized sequence of captured 3D space coordinates $[x(t), y(t), z(t)]$ as input and computes a sequence of features along the trajectory. Then, the feature vector is used for training and recognition. This section presents the features with descriptions. Inspired from the efficient performance of the 2D features as proposed by Jaegar et al. [50] in online 2D text recognition, we have extended their feature-set in our application related to single as well as multiple fingers-based gesture recognition and shape rendering. Single finger-based gestures are projected on X-Y plane and 2D features are extracted as proposed in Npen++ recognizer [50]. 2D features are extended in 3D and used in training and recognition phases.

3.4.1. Gesture direction

In 2D, local writing direction of point $P(t)$ can be described using (1) and (2) as mentioned in the work proposed by Jaeger et al. [50].

$$\cos\theta = \frac{\Delta x(t)}{\Delta s(t)} \tag{1}$$

$$\sin\theta = \frac{\Delta y(t)}{\Delta s(t)} \tag{2}$$

In 3D, the feature is extended with z dimension and cosines of angle α, β and γ with respect to x, y and z axes can be computed using (3)–(5),

$$\cos\alpha = \frac{\Delta x(t)}{\Delta s(t)} \tag{3}$$

$$\cos\beta = \frac{\Delta y(t)}{\Delta s(t)} \tag{4}$$

$$cos\gamma = \frac{\Delta z(t)}{\Delta s(t)} \tag{5}$$

where $\Delta s(t)$, $\Delta x(t)$, $\Delta y(t)$, and $\Delta x(t)$ are defined in (6)–(9) such that $x(t)$, $y(t)$, and $z(t)$ represent the coordinates of the point under consideration at time t.

$$\Delta s(t) = \sqrt{\Delta x^2(t) + \Delta y^2(t) + \Delta z^2(t)} \tag{6}$$

$$\Delta x(t) = x(t-1) - x(t+1) \tag{7}$$

$$\Delta y(t) = y(t-1) - y(t+1) \tag{8}$$

$$\Delta z(t) = z(t-1) - z(t+1) \tag{9}$$

3.4.2. Curvature

The curvature (in 2D) at a point $P(t)$ can be derived using the sequence of three consecutive points [50], e.g. $P(t-2) = [x(t-2), y(t-2)]$, $P(t) = [x(t), y(t)]$, and $P(t+2) = [x(t+2), y(t+2)]$ as given in (10) and (11)

$$\cos\beta = \cos\alpha(t-1) \times \cos\alpha(t+1) + \sin\alpha(t-1) \times \sin\alpha(t+1) \tag{10}$$

$$\sin\beta = \cos\alpha(t-1) \times \sin\alpha(t+1) + \sin\alpha(t-1) \times \cos\alpha(t+1). \tag{11}$$

Cosine and sine values are calculated using the precomputed values of the direction of writing as mentioned in (1) and (2). Similarly, curvature $K(t)$ of a 3D point, say $P[x(t), y(t), z(t)]$, can be computed using Eq. (12),

$$K(t) = \left| \frac{\overrightarrow{dT}}{dP} \right| \tag{12}$$

where the rate of change of gradient with respect to the rate of change of distance at time t are given in (13) and (14)

$$\overrightarrow{dT(t)} = \overrightarrow{T(t+1)} - \overrightarrow{T(t-1)} \tag{13}$$

$$dP(t) = |P(t+1) - P(t-1)|. \tag{14}$$

$\overrightarrow{T(t+1)}$ and $\overrightarrow{T(t-1)}$ are the gradients $C_r(t)$ at time $t+1$ and $t-1$, respectively. They can be estimated using (15) and (16)

$$\overrightarrow{T(t+1)} = C_r'\left(t+1\right) = \frac{\overrightarrow{dP(t+1)}}{d(t)} = \frac{\overrightarrow{P(t+2)} - \overrightarrow{P(t)}}{d(t)} \tag{15}$$

$$\overrightarrow{T(t-1)} = C_r'\left(t-1\right) = \frac{\overrightarrow{dP(t-1)}}{d(t)} = \frac{\overrightarrow{P(t)} - \overrightarrow{P(t-2)}}{d(t)}. \tag{16}$$

3.4.3. Aspect

The aspect of the trajectory in the vicinity of a point, say $[x(t), y(t)]$, can be described by $A(t)$ as proposed in [50]. It is calculated using (17)

$$A(t) = \frac{\Delta y(t) - \Delta x(t)}{\Delta y(t) + \Delta x(t)}. \tag{17}$$

The aspect of the trajectory characterizes the height-to-width ratio of the bounding box constituting the neighboring points of $[x(t), y(t)]$ as depicted in Fig. 6. However, its 3D extension has the following three values, e.g. $A_1(t)$, $A_2(t)$, and $A_3(t)$, respectively as defined in (18)–(20), where $\Delta y(t)$, $\Delta z(t)$, and $\Delta x(t)$ are height, width, and length of the cuboid as shown in Fig. 6.

$$A_1(t) = \frac{2 \times \Delta y(t)}{\Delta x(t) + \Delta y(t)} - 1 \tag{18}$$

$$A_2(t) = \frac{2 \times \Delta z(t)}{\Delta y(t) + \Delta z(t)} - 1 \tag{19}$$

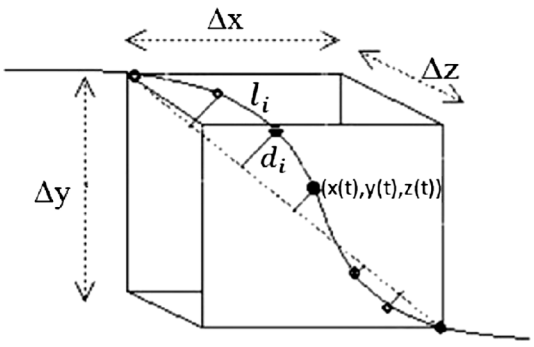


Fig. 6. Bounding box of a point $[x(t), y(t), z(t)]$ enclosing its three preceding and succeeding points.

$$A_3(t) = \frac{2 \times \Delta z(t)}{\Delta z(t) + \Delta x(t)} - 1 \tag{20}$$

3.4.4. Curliness

Curliness feature as denoted by $C(t)$ measures the deviation from a straight line in the vicinity of $P(t)$ in 2D. It is formally defined using (21),

$$C(t) = \frac{L}{max(\Delta x, \Delta y)} - 2 \tag{21}$$

where Δx and Δy represent the width and height of the bounding box containing all points in the vicinity of $P(t)$ and L denotes the sum of lengths of all segments, i.e. length of the trajectory in vicinity of $P(t)$. In 3D, it can be defined using (22)

$$C(t) = \frac{L}{max(\Delta x, \Delta y, \Delta z)} - 2. \tag{22}$$

3.4.5. Slope

Slope is another important feature that is defined by the tangent of the angle subtended by neighboring points [50]. In 3D, slope is defined as the direction ratios represented by l, m, and n as given in (23) with respect to x, y, and z dimensions of the straight line joining the start and end points within the bounding box as shown in Fig. 6.

$$l = \frac{a_1}{s},\ m = \frac{b_1}{s},\ n = \frac{c_1}{s} \tag{23}$$

In the above formulations, values of a_1, b_1, c_1, and s are defined as $a_1 = x(t+3) - x(t-3)$, $b_1 = y(t+3) - y(t-3)$, c_1, and $s = \sqrt{a_1^2 + b_1^2 + c_1^2}$, respectively.

$$= z(t+3) - z(t-3)$$

3.4.6. Lineness

Lineness $L(t)$ [50] is defined as the average square of distance between every point in the cubical box of P and the straight-line joining the first and last points in the box as shown in Fig. 6. It can be calculated using (24), where d_i is the length of the i^{th} segment from the diagonal of the cube and N represents the total number of segments inside the bounding box.

$$L(t) = \frac{1}{N} * \sum d_i^2 \tag{24}$$

3.5. Gesture recognition

Let, a 3D gesture or pattern (G) of length n performed by a user (u) be represented using (25), where $d_i = (x_i, y_i, z_i)$ denotes the instantaneous position of the user's finger in 3D.

$$G_u = [d_1, d_2, d_3, ..., d_n]^T \tag{25}$$

The raw gesture as given in (25) is then converted into a time series representation of high-level features as described in Section 3.4 using (26), where n represents the length of the gesture in number of samples.

$$f_D = [f_1, f_2, f_3, ..., f_n] \tag{26}$$

High-level feature vector of the i^{th} point of multiple fingers-based and single finger-based gestures can be described using (27) and (28), where $f_{D=12}$ and $f_{D=7}$ represent corresponding 12-dimensional and 7-dimensional feature vectors used in 3D and 2D analysis, respectively.

$$f_{12} = [\cos\alpha, \cos\beta, \cos\gamma, K, A_1, A_2, A_3, C, L, l, m, n]_i \tag{27}$$

$$f_7 = [\cos\theta, \sin\theta, \cos\beta, \sin\beta, A, C, m]_i \tag{28}$$

Let the training set contains gestures of U distinct users. Now, recognition of a given test gesture (s) is done using HMM classifier as discussed in the following sections.

3.5.1. Classification using HMM

HMM is a well known classifier to analyze sequences. The feature vector sequence is thus processed using left-to-right continuous density HMMs [51]. One of the important features of HMM is its capability to model sequential dependencies. The basic models considered in this approach are character models as adopted in [52–54]. HMMs can be defined by initial state probabilities π, state transition matrix $A = [a_{ij}]$, $i, j = 1, 2, ...S$, where a_{ij} denotes the transition probability from state i to state j, and observation probability $b_j(O_k)$ modeled with continuous output probability density function. The density function is written as $b_j(x)$, where x represents k dimensional feature vector. Separate Gaussian Mixture Model (GMM) is defined for each state. Formally, the output probability density of state j can be defined using (29),

$$b_j(x) = \sum_{k=1}^{M_j} c_{jk} \aleph\left(x, \mu_{jk}, \Sigma_{jk}\right) \tag{29}$$

where M_j is the number of Gaussian assigned to j, and $\aleph(x, \mu, \Sigma)$ denotes a Gaussian with mean μ, co-variance matrix $\sum$, and c_{jk} represents the weight coefficient of the Gaussian component k of state j. For a model λ, if O is an observation sequence, e.g. $O = (O_1, O_2, ...O_T)$ is assumed to be generated by a state sequence $Q = Q_1, Q_2, ...Q_T$ of length T, we calculate the probability of observation or likelihood as given in (30), where π_{q_1} is initial probability of state 1.

$$P\left(O, Q \middle| \lambda\right) = \sum_Q \pi_{q_1} b_{q_1}(O_1) \prod_T a_{q_{T-1}q_T} b_{q_T}(O_T) \tag{30}$$

In the training phase, features are extracted on each point of a gesture and the feature vector sequence is classified by the trained model. We have constructed separate HMM model for each gesture. Training and recognition steps are described in Algorithm 1.

Algorithm 1. Recognition of 3D Gestures using HMMs

Input: $s \in S_{test}$ is a given test gesture, S_{train} = set of training sequences, C = number of classes or users, S = number of states, O = number of observation symbols.

Output: c_j(Class of s)where $c_j \in C$.

1: **Training:** All training samples with chosen feature set.
2: Initialize $\theta = \{\pi, A, b_j \in B\}$, where B is the observation matrix.
3: Train and fix the model (θ) using training data.
4: **Recognition:** Pass a test gesture (s) through all trained models and find a local maxima θ^* out of all models.
5: Return c_j as the class of the test signature.

3.6. Shape retrieval and rendering

Once a gesture is recognized and perceived through HMM-based classifier, it is followed by rendering of a geometrical shape representing the gesture. However, rendering of shapes are done through retrieval. We have preserved the basic shape for every gesture in a dictionary and retrieve the best matching based on the label recognized by the classifier. Rendering has been performed using the MuPAD note book. This has been found to be a convenient interface for rendering 3D shapes with variable parameters. An example of rendering a 3D shape (heart) is depicted in Fig. 7.

4. Results and discussions

4.1. Dataset acquisition details

In order to evaluate the performance of the proposed dynamic gesture recognition system, we created a large dataset consisting of various gesture classes. Each gesture was performed by 10 different

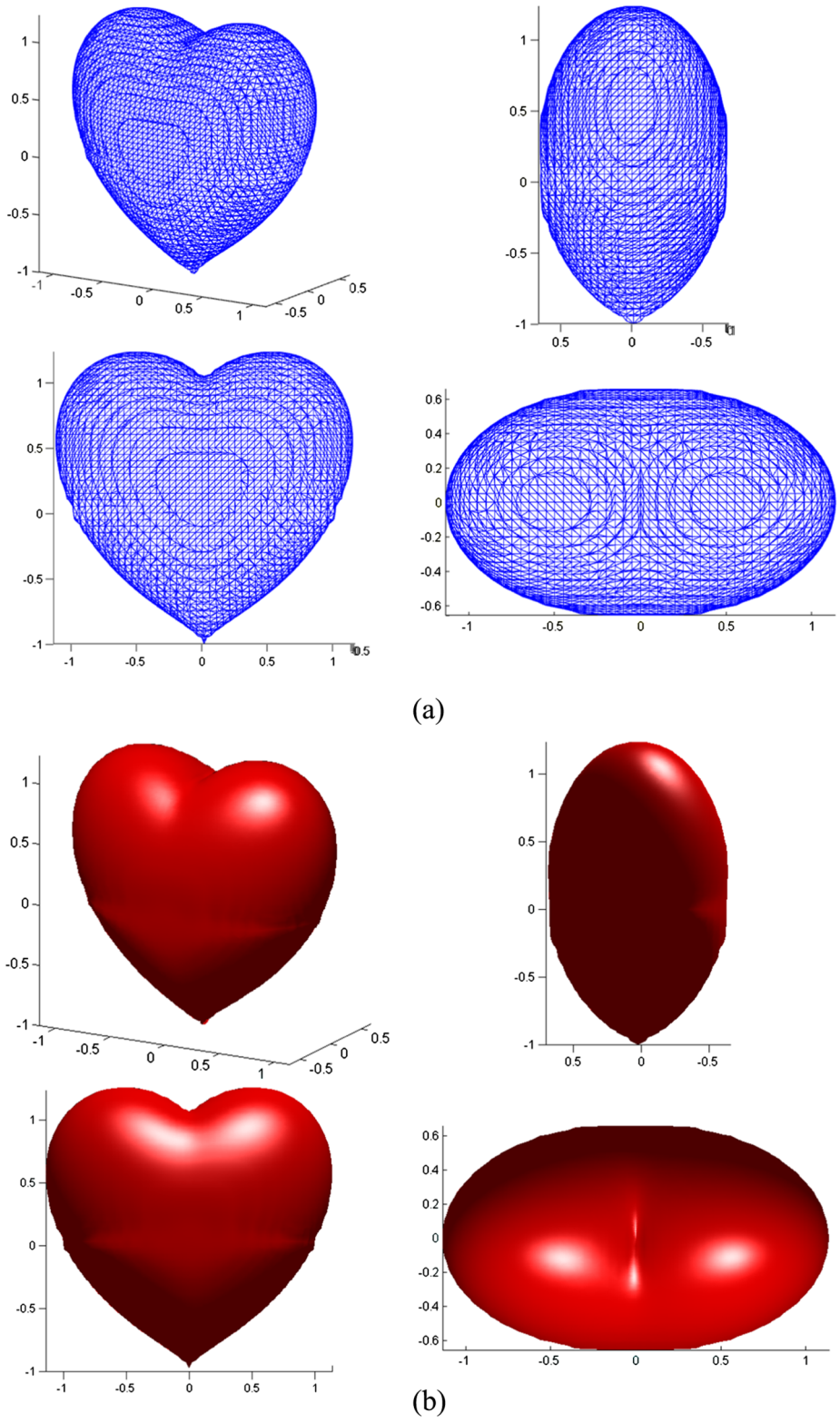


Fig. 7. Different orientations of the 3D rendering of "heart" shape using MuPAD note book interface.

volunteers in varying illumination conditions. The volunteers performed gestures within the field of view of the Leap Motion device. Its SDK processed the sequence of stereo images from the two IR cameras and provides 3-D coordinates of the finger tips. We stored this sequence of 3-D trajectories of the finger tips, from the sensor, for the classification task. Single finger-gesture were done using the index finger and multiple fingers-gesture were done using all right hand fingers as depicted in Figs. 2 and 3. Our dataset consists of 36 geometric as well as non-geometric shapes, which constitutes 21 single finger-based and 15 multiple fingers-based gestures. We asked volunteers to perform a gesture 15 times so as to record the possible variations during action. We collected a total of 5400 samples. Out of these 5400 samples, 3150 were from single finger-based gestures and remaining 2250 samples were from multiple fingers-based gestures. Lastly, high-level features

Table 2
Division of the dataset for evaluation using 5-fold cross validation method.

	Single-Finger Gesture	Multiple-Finger Gesture
Total samples	3150	2250
Used for training	2520	1800
Used for testing	630	450

Table 3
Shapes included in our experiments.

Single-finger	Bag	Circle	Cross
	Diamond	Flower	Heart
	Up	Down	Right
	Left	Pyramid	House
	Pentagon	Moon	Omega
	Triangle	Star	Plus
	Rectangle	@	Leaf
Multiple-finger	Cone	Balloon	Cloud
	Bottle	Hemisphere	Heart
	House	Sq. Pyramid	Spiral
	Pipe	Pyramid	Tree
	cube	Sphere	Cylinder

were computed from the stored 3-D trajectories and fed to the proposed HMM classifier for the training.

The gesture recognition has been evaluated using 5-fold cross validation method. For this purpose, the dataset has been divided into 5 subsets, of which 4 subsets were used for training and rest for testing. This procedure has been repeated 5 times. Hence, each time, training and test sets have been prepared with 2520 and 630 samples for single finger-based recognition. Similarly, 1800 and 450 samples have been used as multiple fingers-based gestures during training and testing. The recognition rates for all the test subsets have been averaged to calculate the final recognition accuracy. Dataset division is described in Table 2. The complete list of different gestures and shapes used in our analysis, is presented in Table 3. A sample video for data collection of "diamond", "star", "cross", "sphere", "cylinder", and "spiral" gestures is available here.[2]

4.2. Gesture type recognition

Before actual recognition of a gesture, its type has been determined first. Involvement of the left hand during recording has played a key role in this phase. As stated earlier, left hand can serve as a virtual switch for capturing multiple fingers-based gestures. If the left hand is closed during the process, it is termed as a multiple fingers-based gesture, otherwise it is termed as a single finger-based gesture.

4.3. Experiments by varying training data

During training of HMMs, the number of samples per gesture class has been varied to study the dependency of recognition performance on the amount of training content. We have varied the size of training samples and obtained an idea about the minimum amount of data required to get satisfactory results. Fig. 8 shows the relation between these two parameters. It may be observed that the recognition performance does not improve significantly between 30 and 150 training samples per gesture class. Whereas, the gestures performed using single finger show improvement in performance with the increase of training data. When the number of samples in training increases from 30 to 90, the performance improves slightly. However, with 150 samples per gesture class, the recognition performance has improved significantly.

[2] http://www.iitr.ac.in/media/facspace/proy.fcs/LeapMotionGesture.mp4.

4.4. Results by varying HMM parameters

During training of HMMs, parameters such as the number of states and the number of Gaussian distributions have been varied. Fig. 9 (a) and (b) present detailed analysis of such experiments. We have noted that increasing the number of Gaussian improves the recognition performance with single finger gesture. However, recognition rate of multiple fingers-based gestures maximizes with 64 Gaussian parameters. After several experiments and validations, we have decided 256 Gaussian parameters and 7 states for single finger-based gestures and 64 Gaussian parameters and 8 states for multiple fingers-based gestures.

4.5. Comparison with other classifiers

$K-$NN is a classification method that uses similarity measure. Similarity between two gesture sequences is measured using Dynamic Time Warping (DTW) [55–57]. The sequences are warped in each point non-linearly along temporal domain to determine a measure of their similarity independent of existing non-linear variations across time-axis. This technique is widely used in many applications such as speech recognition, signatures recognition, and robotics. In our experiment, a gesture is represented by sequence of 3D (x, y, z) space coordinates. The implementation of DTW-based technique to measure similarity between two sequences has been carried with Sakoe-Chiba band [58] to speed up the computation. We have used DTW to estimate similarity between two patterns across all three dimensions. The distance between two signals, e.g. S_1 and S_2 can be evaluated using the matrix D as given in (31), where $d(x_i, y_i)$ can be computed using (32)

$$D\begin{pmatrix} i, j \end{pmatrix} = min\begin{Bmatrix} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{Bmatrix} + d\begin{pmatrix} x_i, y_i \end{pmatrix} \tag{31}$$

$$d\begin{pmatrix} x_i, y_i \end{pmatrix} = \sum_{k=1}^{3} (f_k(S_1, i) - f_k(S_2, j))^2. \tag{32}$$

The matching distances obtained are summed up to get a cumulative distance, and it is considered as the final matching cost. The DTW $+K-$NN-based classifier as described in Algorithm 2 has been used to find the class of a given test signature s.

Algorithm 2. Recognition of 3D Signatures using DTW $+K-$NN

Input: $s \in S_{test}$ = Set of test sequences, S_{train} = Set of training sequences, $k = |S_{train}|$, C = Number of classes or users.
Output: c_j(Class of s)where $c_j \in C$.
  for l = 1 to k **do**
    $r_l \in S_{train}$.
    $D_l^s = DTW(s, r_l)$.
  end for
  Arrange D_j^s in increasing order of values where $j = \{1, 2, ..., k\}$.
  Apply $K-$NN on D_j^s to find the class (c_j) of test sequence (s).
  Return c_j.

We have compared the proposed HMM-based classifier with DTW $+K$-NN-based approach. DTW evaluates similarities between two time series data that may vary with time or speed. Raw coordinates, 12 dimensional (for 3D), and 7 dimensional (for 2D) high-level feature vectors have been fed to HMMs and DTW $+K$-NN classifiers. Fig. 10 depicts results obtained of the above experiments. It may be observed that DTW $+K$-NN with an average accuracy of 72.6% using raw features is not as good as HMM-based classifier (92.87%) when high-level features are used. This amounts to a substantial difference (20.27%) in accuracy. Such a significant improvement using HMMs is mainly due to its superior capability of sequential analysis.

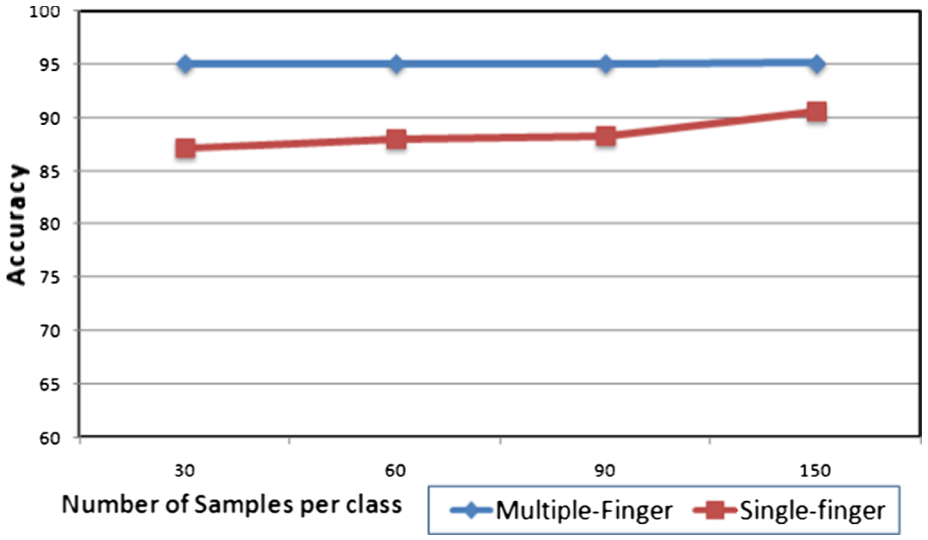
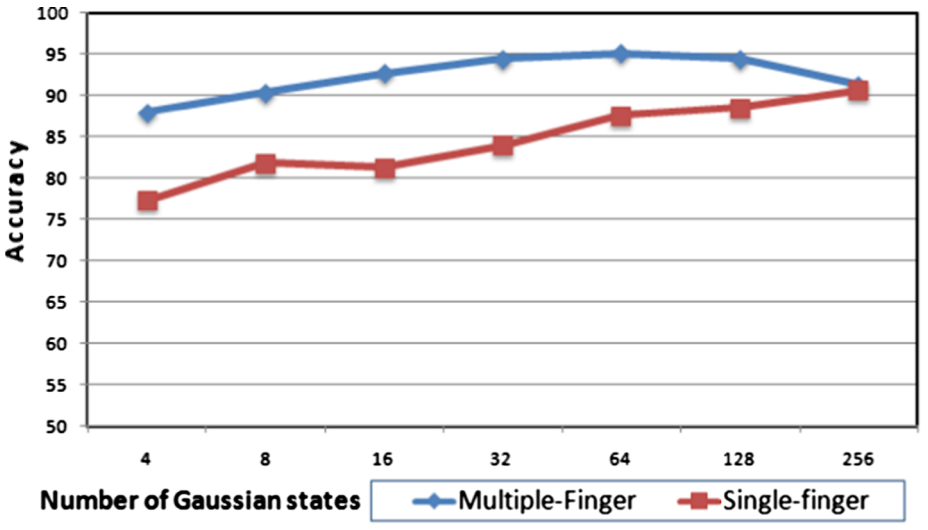
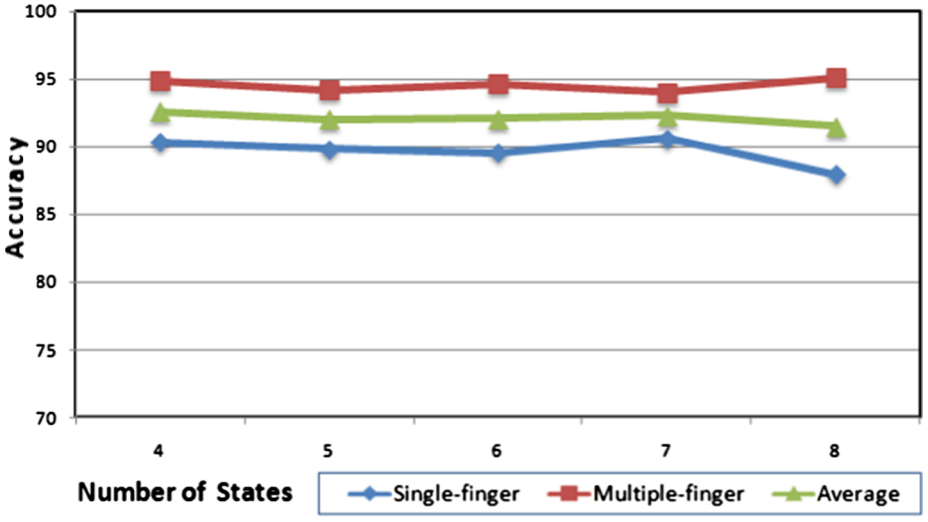


Fig. 8. Performance analysis with number of samples per gesture class.



(a)



(b)

Fig. 9. HMM-based gesture recognition performance against number of (a) Gaussian parameters (b) States.

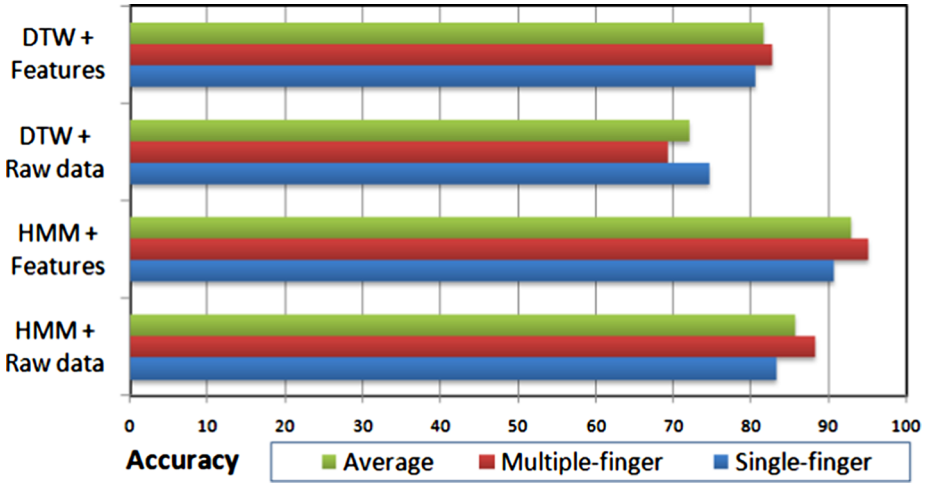


Fig. 10. Gesture recognition performance using HMM and DTW + *K*-NN.

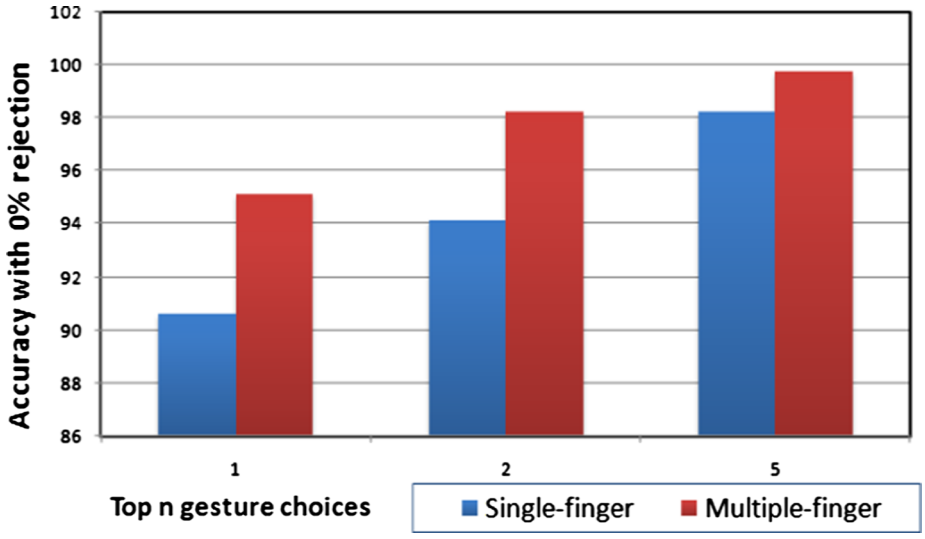


Fig. 11. Recognition results based on different choices when no-rejection was considered.

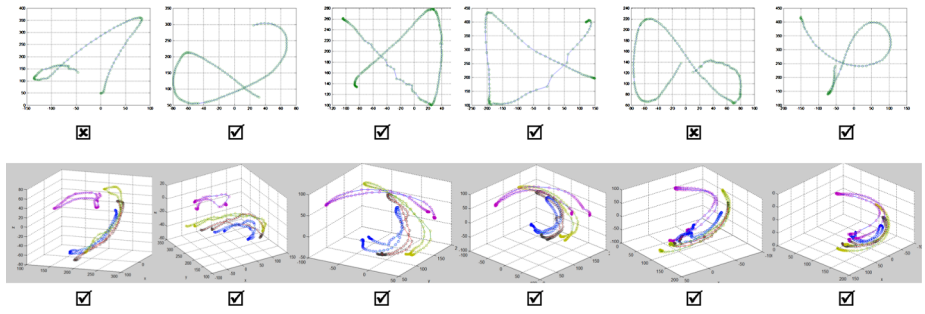


Fig. 12. Some examples of correct and wrong recognized gestures. Top row shows gestures drawn using single finger and bottom row shows gestures using multiple fingers, respectively. (Gestures with the cross mark are wrongly classified.)

4.6. Results of dynamic gesture recognition

As depicted in Fig. 10, recognition accuracy of "3D feature + HMM" on single finger and multiple fingers-based gestures are 95.11% and 90.63%, respectively assuming no-rejection. Some examples of correctly and wrongly recognized gestures are shown in Fig. 12. It can be verified that recognition accuracy of multiple fingers-based gestures is better than that of single finger-based gestures. We have recorded accuracy as high as 98.22% and 94.13% when the first two choices of the results have been considered in multiple fingers and single finger cases. Detailed results of different gestures with varying choices are presented in Fig. 11. Accuracy is increased by 1.56% and 4.12%, respectively while considering the top five choices instead of the top two in multiple fingers and single finger-based gestures. After analyzing the results, we have understood that the improvement obtained by the top-five choices instead of the top-two choices is mainly due to the presence of similarly looking gestures.

Table 4
Error and reliability results of the proposed system with respect to different rejection rates for single finger-based gestures.

Rejection Rate (%)	Error Rate (%)	Reliability (%)
2.3	2.3	98.3
4.7	1.2	98.5
7.1	0.9	99.0
9.5	0.2	99.3

Table 5
Error and reliability results of the proposed system with respect to different rejection rates for multiple fingers-based gestures.

Rejection Rate (%)	Error Rate (%)	Reliability (%)
1.3	3.9	95.9
2.6	3.2	97.0
4.0	1.8	97.9
5.3	1.6	98.8

4.7. Analysis of results

Following statistical measures as described in (33)–(36) have been used for analyzing the results,

$$\text{Recognition rate} = \frac{N_C \times 100}{N_T} \tag{33}$$

$$\text{Error rate} = \frac{N_E \times 100}{N_T} \tag{34}$$

$$\text{Reject rate} = \frac{N_R \times 100}{N_T} \tag{35}$$

$$\text{Reliability} = \frac{N_C \times 100}{N_E + N_C} \tag{36}$$

where N_C is the number of correctly classified gestures, N_E denotes the number of wrongly classified gestures, N_R is the number of rejected gestures, and N_T represents the total number of characters tested by the classifier $N_T = N_C + N_E + N_R$.

Thus, we have computed the recognition results with different rejection rates. It may be noted that 97.93% (99.02%) reliability with 1.8% (0.9%) error has been obtained when 4.01% (7.14%) rejection has been considered in multiple fingers-based (single finger-based) gestures. 98.84% (99.34%) reliability with 1.6% (0.2%) error has been obtained when 5.35% (9.52%) data has been rejected. Recognition reliability values with different rejection rates are given in Tables 4 and 5. Rejection has been done on the basis of the difference of the optimal likelihood values of the best and the second-best recognized gestures. Using this rejection parameter, the confusing pairs of gestures have been opted out from the test data for reliable experiment.

4.8. Error analysis of classification

It has been observed that the errors mainly occurs due the presence of similarly looking gestures during rendering of the shapes. The confusion matrices of recognition results are given in Figs. 13 and 14 for multiple fingers and single finger-based gestures, respectively. These figures show the confusion among gesture-pairs in the experiments and the confusion error rates are highlighted for clarity. In single finger-based cases, ω (omega) and @ have been confused maximum times and the confusion rate has been as high as 20% over the whole dataset. The next most confusing pair has been diamond and triangle, and they have been confused in 10% of the cases. In multiple fingers-based gestures, bottle and cylinder have been confused maximum times and the confusion rate is as high as 13% when computed over all samples.

4.9. Rendering on displays

A few instances of 3D rendering are displayed in Fig. 15. Recognition is followed by evaluation of various geometric quantities like height, radius, length, area, and volume using the minimum size enclosing box. Quantities are measured as per the finger trajectories while

	cone	ballon	bottle	cloud	heart	hemisphere	house	pipe	pyramid	spiral	sq_pyramid	tree	cube	sphere	cylinder
cone	97%								3%						
ballon		100%													
bottle			97%									3%			
cloud				97%			3%								
heart				6%	88%					3%	3%				
hemisphere						100%									
house							94%				6%				
pipe								90%					10%		
pyramid									90%		10%				
spiral										100%					
sq_pyramid											100%				
tree											10%	90%			
cube													100%		
sphere														100%	
cylinder			13%									10%			77%

Fig. 13. Confusion matrix of multiple fingers-based gesture recognition and shape matching.

	@	bag	circle	cross	diamond	down	flower	heart	house	leaf	left	moon	omega	pentagon	plus	pyramid	rectangle	right	star	triangle	up
@	100%																				
bag		97%							3%												
circle	6%		73%				6%	3%		3%			6%	3%							
cross				79%	3%	3%			3%		3%	3%		6%							
diamond		3%			84%							3%		10%							
down						97%								3%							
flower							97%	3%													
heart					6%		3%	88%						3%							
house		6%							91%					3%							
leaf										100%											
left											97%					3%					
moon				3%								91%		3%		3%					
omega	20%												80%								
pentagon									6%	3%			3%	88%							
plus														10%	90%						
pyramid										3%						97%					
rectangle		10%							3%	3%				3%			81%				
right	3%									3%	6%							85%		3%	
star																			100%		
triangle					10%									3%		10%	3%			74%	
up																					100%

Fig. 14. Confusion matrix of single finger-based gesture recognition and shape matching.

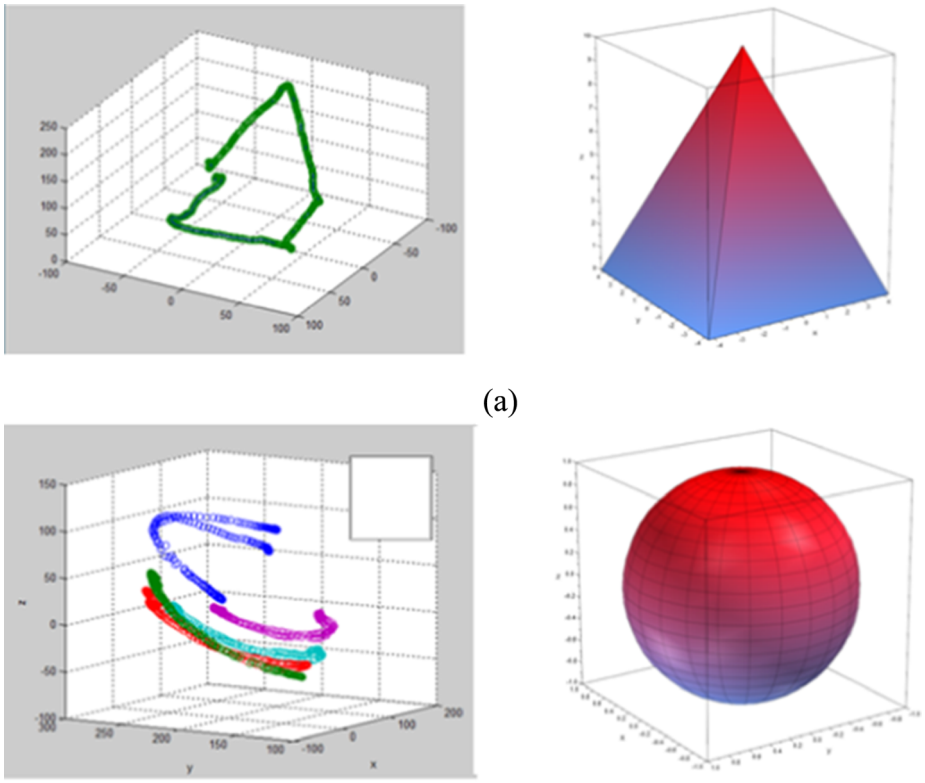


Fig. 15. Two gestures performed with (a) single finger-based (b) multiple fingers-based movements and corresponding 3D shapes rendered using MuPAD Notebook (different colors represent different fingers). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

performing the natural gestures. Considering a “cylinder”, height is evaluated as the difference of the highest and the lowest points traced in the gesture and the diameter be the average distance between the thumb and middle finger. However, depending on their magnitude, they can further be categorized as small, medium or large as depicted in Fig. 17. These quantities are then passed to MuPAD Notebook for 3D rendering on 2D display devices. A few sample shapes that have been rendered using MuPAD interface, are presented in Fig. 16.

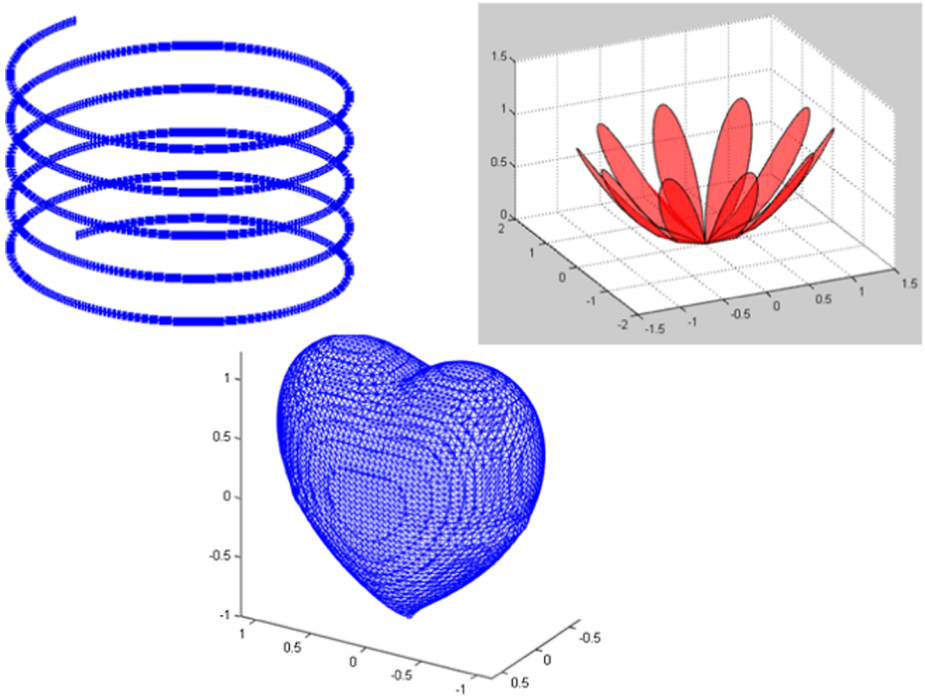


Fig. 16. Examples of shapes rendered after gesture recognition, e.g. “spiral”, “heart”, “flower”.

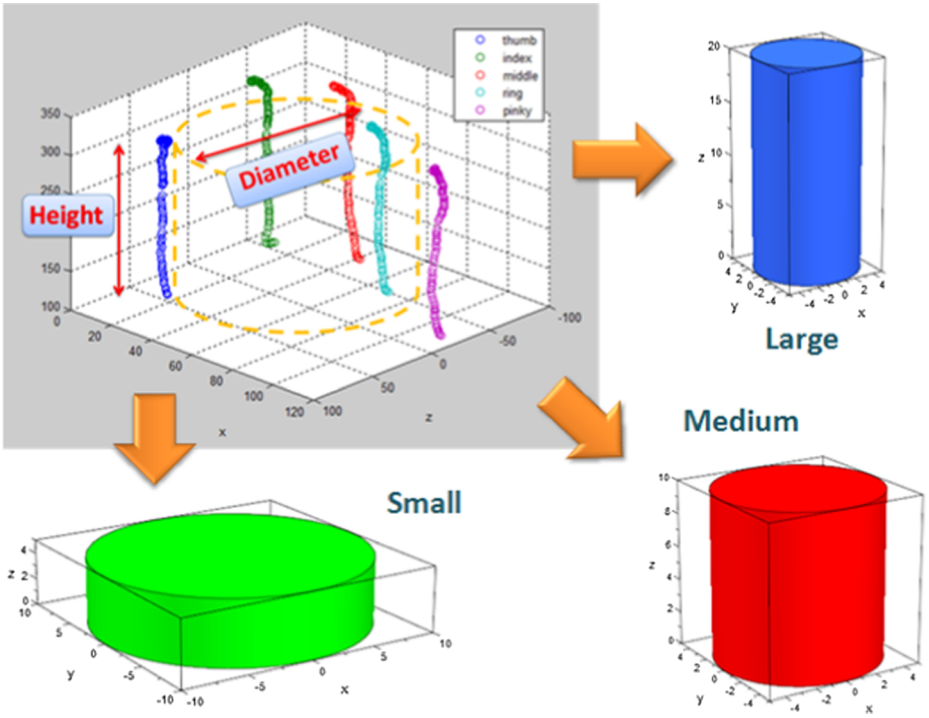


Fig. 17. Variations in 3D shape rendering using parameters extracted from gesture, e.g. “Cylinder”.

Table 6
Comparison of the proposed method with existing hand-gesture recognition systems.

Method	Size of Dataset	Type of Sensor	# of Classes	Accuracy
Kuzmanic et al. [56]	1260	Camera	21	93%
Palacios et al. [5]	810	Camera	6	77.7%
Proposed	**5400**	Leap	36	**92.87%**

Table 7
Computational overhead of the proposed 2D and 3D shape recognition and rendering.

Process	single finger	multiple fingers
Gesture recognition	0.26 s	0.41 s
Rendering	0.03 s	0.03 s

4.10. Comparison with existing methods

We have carried out a meta-comparison with existing methods that are similar in nature. Comparisons have been done mainly based on three parameters e.g. accuracy, type of sensor, number of classes, and size of the dataset. Earlier work by Kuzmanic et al. [56] have used a camera sensor to classify 21 classes and reported a recognition rate of 93% when trained with a dateset of 1260 observations. Also, Palacios et al. [5] have proposed to use camera to classify 10 classes (representation of number 0 to 5 and palm, OK, L and point) with a recognition rate of 77.7% when trained with a dateset of 810 observations, while our proposed method uses Leap Motion Sensor to classify 36 gesture classes and observed a recognition rate of 92.87% when trained with a dateset of 5400 observations. The results of the comparisons are presented in Table 6. It may be observed that the proposed method is superior than the methods mentioned for comparison in terms of accuracy and volume of the data. The superior results are obtained due to better capture of finger articulation using Leap Motion device which is not possible through simple vision-based approach always.

To measure the effectiveness of 3D version of Npen++ features, we have compared it with other existing 3D features. In [48], Haskell et al. have proposed the use of curvature moments associated with 3D curves in both configuration space and velocity space for signature analysis on air. From each signature trajectory, a 6-dimensional feature vector is evaluated comprising of the zero, first and second moments of both position and velocity curvature time series. We have extracted these six features from the samples of our dataset and obtained results. Though the computation time of the 6-dimensional feature vector [48] is low with an average of 0.01 s for both gesture types, the recognition rate is quite low (average 44.14%) when computed for both single finger and multiple fingers-based gestures.

4.11. Analysis of computational complexity

Experiments of the proposed framework have been performed using a desktop computer running with Intel(R) Core™ i5 (1.80 GHz) processor and 4 GB of RAM. We have evaluated the time computation for 2D/3D shape recognition and rendering of test data for both single finger and multiple fingers-based gestures. Table 7 shows the time taken for recognition and rendering process. The recognition and rendering time of all test examples from a class have been averaged to compute the time. It has been observed that the average recognition time for single-finger and multiple-finger type are 0.26 s and 0.41 s per gesture, respectively. The average shape rendering time has been found to be 0.03 s for both single finger and multiple fingers-based gestures.

In this section, we explain the data collection process followed by details on the number of sample points collected and their distributions. We have discussed in detail the performance of the proposed method w.r.t. the essential hyper-parameters of the learning algorithm like number of samples per class, number of gaussian states and number of states. We have also demonstrated a detailed comparison with DTW $+K$-NN method and showcased the effectiveness of the proposed 3-D features explained in Section 3.4. In the end of the section, we have emphasized about the usage of MuPAD notebook for 3D rendering and presented few gestures and the corresponding renderings. We have shown the confusion matrix of all gesture classes and later discussed the inference time and rendering time of the method.

5. Conclusion and future scopes

This paper proposes a new method that can recognize natural gestures and render 2D/3D geometric and non-geometric shapes using Leap Motion device by analyzing the motion of fingers in three-dimensional space. Our system captures trajectories of fingers using a 3D hand tracking methodology developed with the help of Leap Motion device. A simple but effective gesture spotting method has been proposed to ensure real-time execution with minimal effect of noises. A continuous left-to-right HMM has been used to model and classify gestures. Some possible extensions of the work can be, building a stereo vision platform to replace the leap motion for capturing 3D hand tracking and developing new algorithms to improve the recognition accuracy. Besides, the work can be upgraded to recognize more complex geometric shapes and further be integrated with modeling software like AutoCAD and AutoDesk 3D to provide the users a hands-free and complex shape modeling environment.

Disclosure of potential conflicts of interest

The authors declare that they have no conflict of interest.

Appendix A

Some examples of gestures representing 2D and 3D shapes are shown in Fig. 18. Top 17 figures demonstrate gestures using single finger and rest of the 11 figures show the gestures using multiple fingers.

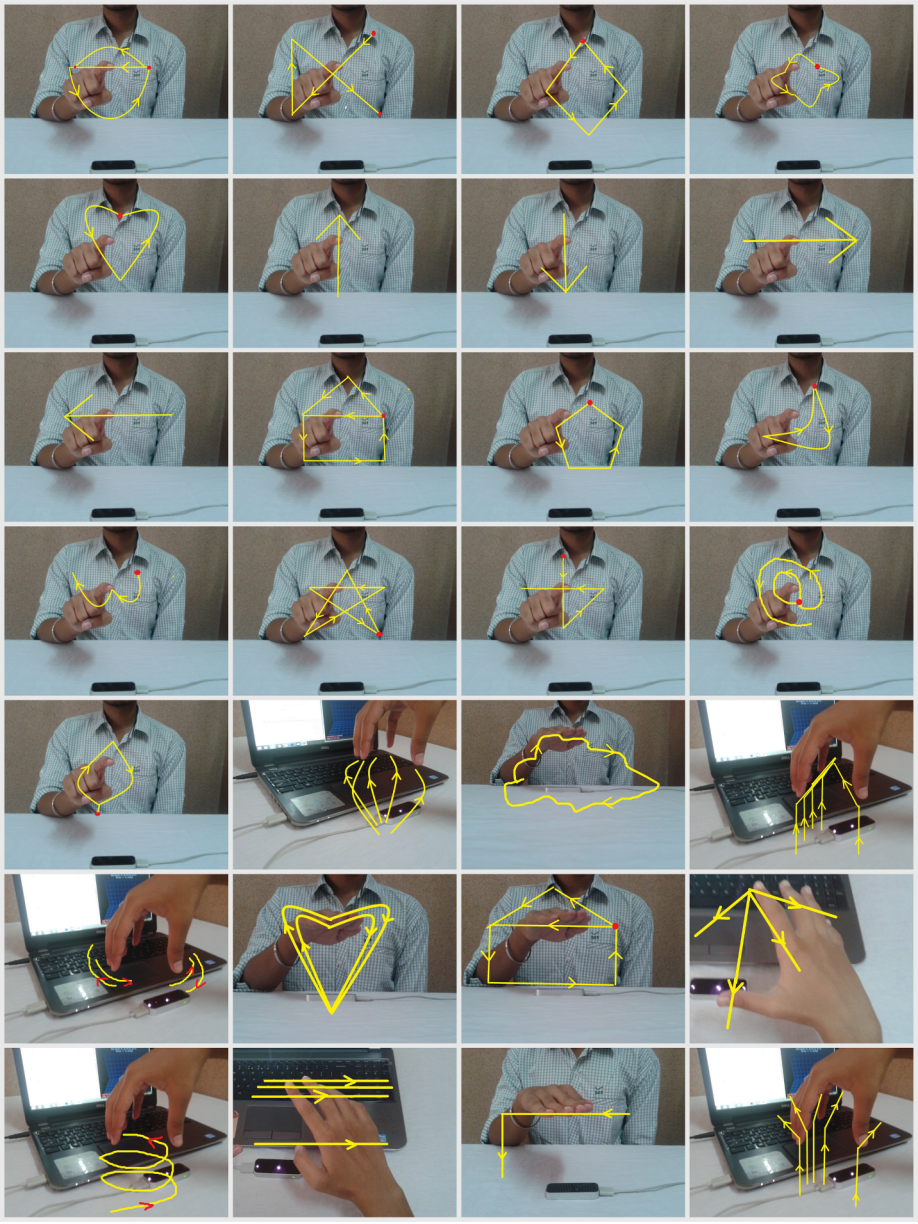


Fig. 18. Examples of gestures representing various regular/non-regular shapes.

Appendix B. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.displa.2019.03.001.

References

[1] G.C.S. Ruppert, L.O. Reis, P.H.J. Amorim, T.F. de Moraes, J.V.L. da Silva, Touchless gesture user interface for interactive image visualization in urological surgery, World J. Urol. 30 (5) (2012) 687–691.

[2] S. Mitra, T. Acharya, Gesture recognition: a survey, IEEE Trans. Syst. Man Cybern. C Appl. Rev. 37 (3) (2007) 311–324.

[3] E. Marilly, A. Gonguet, O. Martinot, F. Pain, Gesture interactions with video: from algorithms to user evaluation, Bell Labs Tech. J. 17 (4) (2013) 103–118.

[4] Y. She, Q. Wang, Y. Jia, T. Gu, Q. He, B. Yang, A real-time hand gesture recognition approach based on motion features of feature points, IEEE Int. Conf. Comput. Sci. Eng. (ICCSE), 2014, pp. 1096–1102.

[5] J. Palacios, C. Sagues, E. Montijano, S. Llorente, Human-computer interaction based on hand gestures using rgb-d sensors, Sensors 13 (9) (2013) 11842–11860.

[6] C. Tan, S. Chin, W. Lim, Game-based human computer interaction using gesture recognition for rehabilitation, IEEE Int. Conf. Control Syst., Comput. Eng. (ICCSCE), 2013, pp. 344–349.

[7] R. Gopalan, B. Dariush, Toward a vision based hand gesture interface for robotic grasping, IEEE Int. Conf. Intell. Robot. and Syst. (ICIRS), 2009, pp. 1452–1459.

[8] D. Kelly, J. McDonald, C. Markham, Continuous recognition of motion based gestures in sign language, IEEE Int. Conf. Comput. Vis. Workshops (ICCVW), 2009, pp. 1073–1080.

[9] M. Rahman, M. Ahmed, A. Qamar, D. Hossain, S. Basalamah, Modeling therapy rehabilitation sessions using non-invasive serious games, IEEE Int. Symp. on Med. Meas. Appl. (ISMMA), 2014, pp. 1–4.

[10] H. Cheng, L. Yang, Z. Liu, Survey on 3d hand gesture recognition, IEEE Trans. Circuits Syst. Video Technol. 26 (9) (2016) 1659–1673.

[11] J. Cashion, C. Wingrave, J. LaViola, Dense and dynamic 3d selection for game-based virtual environments, IEEE Trans. Vis. Comput. Graphics 18 (4) (2012) 634–642.

[12] J. Weissmann, R. Salomon, Gesture recognition for virtual reality applications using data gloves and neural networks, Int. Joint Conf. Neural Netw. (IJCNN), vol. 3, IEEE, 1999, pp. 2043–2046.

[13] D. Lee, J.-S. Kim, H.-J. Lee, Fast hand and finger detection algorithm for interaction on smart display, Displays 55 (2018) 55–63.

[14] S. Kang, A. Roh, H. Hong, Using depth and skin color for hand gesture classification,

IEEE Int. Conf. Consum. Electron. (ICCE), 2011, pp. 155–156.
[15] J. Segen, S. Kumar, Shadow gestures: 3d hand pose estimation using a single camera, IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), vol. 1, IEEE, 1999.
[16] S. Berman, H. Stern, Sensors for gesture recognition systems, IEEE Trans. Syst. Man Cybern. C Appl. Rev. 42 (3) (2012) 277–290.
[17] R.Y. Wang, J. Popović, Real-time hand-tracking with a color glove, ACM Trans. Graphics, vol. 28, ACM, 2009, p. 63.
[18] G. Plouffe, A.M. Cretu, P. Payeur, Natural human-computer interaction using static and dynamic hand gestures, IEEE Int. Workshop Haptic Audio Vis. Environ. Appl. (HAVE), 2015, pp. 1–6.
[19] K. Vamsikrishna, D.P. Dogra, M.S. Desarkar, Computer vision assisted palm rehabilitation with supervised learning, IEEE Trans. Biomed. Eng. 63 (5) (2016) 991–1001.
[20] A.K. Dash, S.K. Behera, D.P. Dogra, P.P. Roy, Designing of marker-based augmented reality learning environment for kids using convolutional neural network architecture, Displays 55 (2018) 46–54.
[21] S.K. Behera, D.P. Dogra, P.P. Roy, Analysis of 3d signatures recorded using leap motion sensor, Multimed. Tools Appl. 77 (11) (2018) 14029–14054.
[22] Y. Li, Hand gesture recognition using kinect, IEEE Int. Conf. Comput. Sci. Automat. Engin. (CSAE), IEEE, 2012, pp. 196–199.
[23] M. Draelos, Q. Qiu, A. Bronstein, G. Sapiro, Intel realsense = real low cost gaze, IEEE Int. Conf. Image Process. (ICIP), 2015, pp. 2520–2524.
[24] Leap motion: Finger tip tracking accuracy, < https://www.leapmotion.com/news/ > (accessed 10 January 2017).
[25] F. Weichert, D. Bachmann, B. Rudak, D. Fisseler, Analysis of the accuracy and robustness of the leap motion controller, Sensors 13 (5) (2013) 6380–6393.
[26] Vinayak, S. Murugappan, H. Liu, K. Ramani, Shape-it-up: hand gesture based creative expression of 3d shapes using intelligent generalized cylinders, Comput. Aided Des. 45 (2) (2013) 277–287.
[27] Freeform: Leap motion, < http://blog.leapmotion.com/designing-leap-motion-sculpting-app/ > (accessed 10 January 2017).
[28] Leap motion's gesture-based 3d modeling app, < http://3dprintingindustry.com/news/leap-motions-gesture-based-3d-modeling-app-21336/ > (accessed 10 January 2017).
[29] Z. Bai, A.F. Blackwell, G. Coulouris, Using augmented reality to elicit pretend play for children with autism, IEEE Trans. Vis. Comput. Graphics 21 (5) (2015) 598–610.
[30] Y. Baek, J. Choi, J. Park, H. Park, 3d interactive whiteboard system using rgb-d camera, IEEE Int. Symp. Consum. Electron. (ISCE), 2014, pp. 1–2.
[31] A. Deligiannakou, A. Papavasileiou, E. Polymeraki, C. Volioti, A. Mavridis, T. Tsiatsos, Y. Revtyuk, L. Kolesnyk, Exploiting 3d virtual environments for supporting role playing games, Int. Conf. Interact. Collabor. Learn. (ICICL), 2012, pp. 1–7.
[32] Y. Akagi, M. Furukawa, S. Fukumoto, Y. Kawai, H. Kawasaki, Demo paper: a content creation system for interactive 3d animations, IEEE Int. Conf. Multimed. Expo Workshops (ICMEW), 2013, pp. 1–2.
[33] J. Mora, W.-S. Lee, G. Comeau, S. Shirmohammadi, A. El Saddik, Assisted piano pedagogy through 3d visualization of piano playing, IEEE Int. Workshop Haptic Audio Vis. Environ. Appl. (HAVE), 2006, pp. 157–160.
[34] N. Haouchine, J. Dequidt, M.-O. Berger, S. Cotin, Monocular 3d reconstruction and augmentation of elastic surfaces with self-occlusion handling, IEEE Trans. Vis. Comput. Graphics 21 (12) (2015) 1363–1376.
[35] M.W. Krueger, Environmental technology: making the real world virtual, Commun. ACM 36 (7) (1993) 36–37.
[36] C.D. Shaw, M. Green, Thred: a two-handed design system, Multimed. Syst. 5 (2) (1997) 126–139.
[37] H. Nishino, D. Nariman, K. Utsumiya, K. Korida, Making 3d objects through bimanual actions, IEEE Int. Conf. Syst., Man, Cybern. (SMC), vol. 4, 1998, pp. 3590–3595.
[38] P. Kumar, S. Mukherjee, R. Saini, P.P. Roy, D.P. Dogra, B.G. Kim, Plant disease identification using deep neural networks, J. Multimed. Inform. Syst. 4 (4) (2017) 233–238.
[39] S. Mukherjee, R. Saini, P. Kumar, P.P. Roy, D.P. Dogra, B.G. Kim, Fight detection in hockey videos using deep network, J. Multimed. Inform. Syst. 4 (4) (2017) 225–232.
[40] J. Zariffa, J. Steeves, Computer vision-based classification of hand grip variations in neurorehabilitation, IEEE Int. Conf. Rehabil. Robot. (ICRR), 2011, pp. 1–4.
[41] T. Brassil, J. Brassil, Hand rehabilitation glove, United States Patent 6454681, Sept 2002. URL < http://www.google.com/patents/US6454681 > .
[42] C. Hao, W. Qingxiang, C. Lixing, Design of the workstation for hand rehabilitation based on data glove, IEEE Int. Conf. Bioinformatics and Biomedicine Workshops (ICBBW), 2010, pp. 769–771.
[43] C. Schonauer, T. Pintaric, H. Kaufmann, S. Jansen Kosterink, M. Vollenbroek-Hutten, Chronic pain rehabilitation with a serious game using multimodal input, Int. Conf. Virtual Rehabil. (VR), 2011, pp. 1–8.
[44] D. Charles, K. Pedlow, S. McDonough, K. Shek, T. Charles, An evaluation of the leap motion depth sensing camera for tracking hand and fingers motion in physical therapy, Int. Conf. Interact. Technol. Games (ICITG), 2013.
[45] M. Khademi, H. Mousavi Hondori, A. McKenzie, L. Dodakian, C.V. Lopes, S.C. Cramer, Free-hand interaction with leap motion controller for stroke rehabilitation, CHI Extend. Abstracts Human Factors Comput. Syst. (HFCS), ACM, 2014, pp. 1663–1668.
[46] D. Webster, O. Celik, Experimental evaluation of microsoft kinect's accuracy and capture rate for stroke rehabilitation applications, IEEE Haptics Symp. (HAPTICS), 2014, pp. 455–460.
[47] I. Horváth, N. Tromp, J. Daalhuizen, Comprehending a hand motion language in shape conceptualization, ASME Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf. (IDETC/CIE), American Society of Mechanical Engineers, 2003, pp. 1047–1061.
[48] R.E. Haskell, D.M. Hanna, K.V. Sickle, 3d signature biometrics using curvature moments, Int. Conf. Artif. Intell. (ICAI), vol. 2, 2006, pp. 718–721.
[49] B.-G. Kim, G.-S. Hong, K.E. Psannis, Design of efficient shape feature for object-based watermarking technology, Multimed. Tools Appl. 76 (21) (2017) 22741–22759.
[50] S. Jaeger, S. Manke, A. Waibel, Npen + +: an on-line handwriting recognition system, Int. Workshop Front. Handwrit. Recognit. (IWFHR), 2000, pp. 249–260.
[51] L. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.
[52] E.R. Justino, A. Yacoubi, F. Ortolozzi, R. Abourin, An off-line signature verification system using hidden markov model and cross-validation, Brazilian Symp. Comput. Graphics Image Process. (BSCGIP), IEEE, 2000, pp. 105–112.
[53] R.S. Kashi, J. Hu, W.L. Nelson, W. Turin, On-line handwritten signature verification using hidden markov model features, Int. Conf. Document Anal. Recognit. (ICDAR), vol. 1, IEEE, 1997, pp. 253–257.
[54] Y. Iwai, H. Shimizu, M. Yachida, Real-time context-based gesture recognition using hmm and automaton, Int. Workshop Recognit., Anal., Track. Faces Gestures in Real-Time Syst. (RATFGRS), IEEE, 1999, pp. 127–134.
[55] C.C. Tappert, C.Y. Suen, T. Wakahara, The state of the art in online handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 12 (8) (1990) 787–808.
[56] A. Kuzmanic, V. Zanchi, Hand shape classification using dtw and lcss as similarity measures for vision-based gesture recognition system, Proc. EUROCON, Comput. Tool (EUROCON), 2007, pp. 264–269.
[57] M. Faundez-Zanuy, On-line signature recognition based on VQ-DTW, Pattern Recognit. 40 (3) (2007) 981–992.
[58] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoust. Speech Signal Process. 26 (1) (1978) 43–49.