ABSTRACT Gesture-based human-robot interaction has been an important area of research in recent years. The primary aspect for the researchers has always been to create a gesture detection system that is insensitive to lighting and backdrop surroundings. This research proposes a 3D gesture recognition and adaptation system based on Kinect for human-robot interaction. The framework comprises the following four modules: pointing gesture recognition, 3D dynamic gesture recognition, gesture adaptation, and robot navigation. The proposed dynamic gesture recognition module employs three distinct classifiers: HMM, Multiclass SVM, and CNN. The adaptation module can adapt to new and unrecognized gestures applying semi-supervised self-adaptation or user consent-based adaptation. A graphical user interface (GUI) is built for training and testing the proposed system on the fly. A simple simulator along with two different robot-navigation algorithms are developed to test robot navigation based on the recognized gestures. The framework is trained and tested through a five-fold cross-validation method with a total of 3,600 gesture instances of ten predefined gestures performed by 24 persons (three age categories: Young, Middle-aged, Adult; each with 1,200 gestures). The proposed system achieves a maximum accuracy score of 95.67% with HMM for the Middle-aged category, 92.59% with SVM for the Middle-aged category, and 89.58% with CNN for the Young category in dynamic gesture recognition. Considering all the three age categories, the system achieves average accuracies of 94.61%, 91.95%, and 88.97% in recognizing dynamic gestures with HMM, SVM, and CNN respectively. Moreover, the system recognizes pointing gestures in real-time.

INDEX TERMS Kinect, pointing gesture recognition, 3D dynamic gesture recognition, gesture adaptation, robot navigation, human–robot interaction.

I. INTRODUCTION

Human-Robot Interaction (HRI) is a multidisciplinary field of modern science and technology in which the interactions between humans and robots are mainly studied. Along with the vast evolution of science and technology, the use of robots has changed significantly over time. Earlier, mass automation was the main criterion for human–robot interaction. Currently, robots not only serve industrial purposes but are also more efficacious in our day-to-day lives, for example, drone cameras serve food in restaurants and deliver parcels. Therefore, interacting with robots using gestures has been considered as one of the most demanding research fields in the past decade. As various sensing devices are readily available, gesture recognition is a prominent method for robot navigation. Pointing or directing gestures are considered more robust and efficient for robot navigation than voice recognition, which depends on the frequency of voices and accents.

A. MOTIVATION

Hand gestures are not only a part of nonvocal interaction but are also a ubiquitous part of our spoken language [1].
As the use of robots to perform complex tasks is becoming more popular day by day, providing a natural way of interaction between humans and robots has become the primary concern of human-robot interaction studies. Because humans typically use pointing gestures to point to a direction or object, pointing gestures can be used to navigate a robot, thus providing a natural method of interaction. Other dynamic gestures can also be used to provide various commands to robots. Furthermore, the pointing patterns in different regions and cultures are invariant. This paper also presents a combined framework of pointing gestures and dynamic gestures, which is a flexible way to interact with robots. However, a limited number of gestures are inadequate for employing robots in various tasks. If the operation of a robot changes, the corresponding gestures also change according to the task to be performed and the environment surrounding the robot. As a result, a robust gesture recognition system is necessary that can adapt to changes and be applied for robot navigation directly. This study also provides an efficient and robust gesture recognition system capable of adapting new gestures by combining supervised and semi-supervised learning methods.

Apart from these, dynamic gesture recognition tasks and its applications have many obstacles. A fundamental problem arises when it comes to interaction devices. Several methods exist for interacting with robots, such as data gloves [2], markers [3], and gesture controllers [4]. However, these methods are not natural ways to interact with humans or robots, and they are not cheap as well. Since there are various degrees of freedom and hand features are sometimes unobvious, hand gesture recognition depending on skin color has some limitations [5]. Template matching is unstable owing to its dependence on the background contrast. However, time-of-flight (ToF) cameras can be used for robust gesture recognition in the presence of any background color, but they are costly and the viewing angle is narrow [6]. In this study, we used Kinect [7] which houses RGB and depth cameras in a compact sensor. It can be used to track human body joints and to recognize 3D gestures using these joints. This leads to an effective robot navigation system that uses hand gestures. The points mentioned above motivated us to address these issues of human-robot interaction.

**B. OBJECTIVES**

The main objective of this study is to develop a hand gesture-based natural interaction system between humans and robots. The targeted system should provide an adaptive, fast, efficient, robust, yet simple gesture recognition and robot navigation system. The system should also provide flexibility and no additional constraints should be imposed when a person performs a gesture, which makes the system highly usable and robust. To perform this, we use Microsoft Kinect sensor as the medium of interaction. We also aim to use pointing and dynamic commanding gestures in 3D for interacting with the robots. We validate our system with our own gesture dataset that includes pointing gestures aimed at any direction within Kinect’s field of view and predefined ten 3D dynamic gestures to give various commands to robots. The number of dynamic gestures is not limited to 10. Furthermore, in our proposed system, any number of new gestures can be added dynamically. The system adapts to new gestures automatically, and it can be trained with the new gesture data on the fly. As the number of gestures increases, they represent increasingly complex patterns. To address this, five out of ten predefined gestures of our gesture domain constitute complex patterns. We also implement supervised and semi-supervised learning methods in our system to make it adaptable to new unlabeled data. As it is infeasible to label a large amount of data with human supervision, our system aims to label them automatically based on its previous knowledge.

To maintain the performance of the system up to the mark, we aim to impose weights on the labeled and unlabeled data. We also aim to implement three different classifiers for gesture recognition, so that a comparative study can be conducted based on their accuracy, advantages, and disadvantages.

**C. KEY CONTRIBUTIONS**

This paper is an extended version of a conference paper that appeared as [8]. The key additions to this journal version are summarized as follows.

- The proposed system implements a real-time method for estimating the pointing direction of a pointing gesture performed by the right hand, using geometric calculations.
- The proposed system uses 3D hand joint coordinates as the main feature for 3D dynamic gesture recognition. The feature vector for a gesture is generated by capturing the hand joint coordinate sequence, which is translated using a reference point such that every gesture has the same starting position. It provides flexibility to the users, allowing them to perform gestures while standing anywhere within Kinect’s field of view.
- The proposed system uses three classifiers for classifying the 3D dynamic gestures: Hidden Markov Models, Multi-class Kernel Support Vector Machines, and Convolutional Neural Networks.
- A system combining supervised and semi-supervised learning methods that can adapt to new and recognized gestures has been proposed. The system implements self-labeled semi-supervised learning, which solves the false-positive issues associated with traditional semi-supervised learning.
- Two algorithms have been proposed for robot navigation using pointing gestures. The first proposed algorithm (Method 1) considers a boundary around the Kinect, and the robot can only move within the boundary. It then determines the spot on the boundary where the user is pointing and navigates the robot straight towards that point. The second proposed algorithm (Method 2) initially adjusts the robot’s current position such that it aligns with the pointing direction and then moves along the pointing line following a straight path.
• This study also develops an easy-to-use graphical user interface (GUI) and a simple simulator to test the robot navigation methods using recognized dynamic gestures.

The remainder of this paper is organized as follows. Section 2 presents related works on 2D and 3D hand gesture recognition approaches. In Section 3, the proposed system is described. Section 4 discusses the experiment setup for the 3D gesture recognition, adaptation, and robot navigation. Section 5 presents an analysis of the performance of the proposed system. Section 6 concludes the paper by highlighting the limitations of the proposed system as well as the scope for future work.

II. RELATED WORKS

Human beings use gestures not only for nonverbal communication but also along with speech. Hand gestures are used to express ideas or meanings, to demonstrate or accentuate something in the context. So recognizing hand gestures has always been a prominent research domain for researchers. Much research has been conducted on various methods of hand gesture recognition and classification.

A. COMPUTER VISION BASED APPROACH

A webcam-based 3D hand gesture recognition method was proposed by Rodriguez et al. [9]. In order to develop a robust hand detection method that is less dependent on illumination and behavioral changes, the AdaBoost algorithm trained with Haar-like features has been used. Murthy et al. [10] proposed a model that relies on a feed-forward neural network. The hand gestures performed by the user are recorded via a Fronttech e-cam camera and stored on a disk. The backpropagation algorithm has been used to train the neural network, which classifies hand gestures into ten categories. In human-machine interaction (HMI), a gesture recognition method was proposed by Ke et al. [11]. A convolutional neural network with an RGB-D model was used in this study for motion detection. A computer vision-based approach is presented by Zhu et al. [12] for gesture recognition. In 3D context with clustered background, the proposed model achieved high accuracy.

Liu et al. [13] used LD-ConGR RGB-D video dataset for gesture recognition which contains long distance gesture instances. Their proposed system achieved 89.66% accuracy with 3D ResNeXt for recognizing gestures using the LD-ConGR RGB-D video dataset. Manssor et al. [14] used a 3D CNN to recognize real-time hand motions that are invariant to different lighting conditions. They trained the model using a large number of videos of humans performing various movements under different lighting conditions. The trained model can recognize real-time webcam hand motion. With minimum preprocessing, they achieved 76.40% accuracy on the training set and 66.56% accuracy on the validation set.

Chen et al. [15] suggested a local- and dual-attention 3D Convolutional Network for gesture recognition. RGB and depth data from IsoGD and Briareo were utilized, along with hand moves made from RGB data. The network employed the I3D model with dual spatiotemporal attention to extract RGB and depth data characteristics. Final categorization was achieved by multiplying and fusing the retrieved characteristics. This approach achieved an accuracy of 68.15% for the IsoGD test set. Rahman et al. [16] offered a rudimentary gesture recognition system with 3D CNN that interprets “doing other things”, “swiping down”, “swiping left”, “zooming out with two fingers”, and “drumming fingers” from Jester dataset. They employed spatiotemporal filter to extract features from RGB video data. The model can interpret hand motions with 85.7% accuracy and 0.4% loss.

B. MARKER AND GLOVE BASED APPROACHES

A marker-based emotion identification method was proposed by Kapur et al. [17] that can detect four emotional states (sadness, joy, anger, and fear) from body movements. They used a VICON motion-capturing camera and 14 reference points for capturing the gesture data. Five different classifiers were used: Logistic Regression, Naive Bayes, Decision Tree based on the C4.5 algorithm, Artificial Neural Network, and Support Vector Machine trained with sequential minimal optimization (SMO). Siam et al. [18] proposed a desktop-manipulation system based on markers. They used a webcam for image acquisition, and template matching for marker detection. The input image was converted into a hue saturation intensity (HSI) model to reduce the effect of different light intensities. To track the marker, a Kalman filter was used for smooth detection. Pławiak et al. [19] proposed a system for gesture recognition in which gloves were used along with ten sensors. The captured gestures were classified using different machine learning algorithms, such as the Probabilistic Neural Network, Support Vector Machine, and K-Nearest Neighbors algorithm. In order to translate American sign language, Abhishek et al. [20] developed a glove model using touch sensors.

C. KINECT BASED APPROACH

Gongfa Li et al. [21] proposed a dynamic gesture detection framework for the Internet of Things devices. They used Kinect Skeletal Tracking data by employing a Kinect sensor and implied HMM model to perform the task. They achieved an average accuracy of 91.6% and maximum accuracy of 96% for a specific gesture. However, their pre-defined ten gestures are very simple compared to our predefined 10 gestures. Moreover, our proposed system performs better in terms of accuracy and flexibility. Miao et al. [22] proposed multimodal gesture recognition based on a ResC3D network. They utilized the ResC3D model for learning and feature extraction, and achieved an accuracy of 67.71.

Yi Li [23] proposed a Kinect-based gesture recognition method to recognize popular gestures and numbers such as “Start Gesture”, “Thumb-up”, “victory”, “One” to “Nine”. Hand detection was conducted using K-means clustering and the convex hull of the hand clusters was calculated using the Graham Scan algorithm. A deep CNN-based model had been proposed by Liu et al. [24] for surgery. For the input,
a Microsoft Kinect sensor was used to generate images that were classified using a deep CNN network. A combination of Kinect and sEMG signal for gesture recognition was presented by Sun et al. [25]. This was done using Weighted D-S evidence theory.

Heickal et al. [26] proposed a recognition system to aid the umpires in decision-making in a cricket game. First, the body joints were extracted, and the distance between the user and Kinect was measured. For classification, Naive Bayes and Back Propagation Neural Network had been used. Xu et al. [27] implemented Hidden Markov Models (HMMs) to classify the hand gesture feature vectors and the Baum-Welch algorithm to train the classifier. Ren et al. [28] introduced a part-based hand gesture recognition system that uses a new distance metric named Finger-Earth Movers Distance (FEMD) to measure the difference between hand shapes using the finger parts. Modified Minimum Near-Convex Decomposition has been used for filtering out the non-finger parts and Thresholding Decomposition has been used for finger detection in the time-series curve. Gestures are recognized by template matching based on Finger-Earth Movers Distance.

III. PROPOSED SYSTEM OF GESTURE RECOGNITION

This research proposes and implements a Kinect-based 3D gesture recognition and adaptation system, along with robot navigation, combining pointing and dynamic commanding gesture recognition. The system takes pointing and dynamic commanding gestures as input through Kinect, and after the gesture recognition and adaptation process, they are translated into instructions for a robot. Subsequently, the robot navigates using two different navigation algorithms implemented by the system. A block diagram of the proposed system is shown in figure 1.

The system consists of four modules: pointing gesture recognition, 3D dynamic gesture recognition, gesture adaptation, and robot navigation. At first, the required body joints are acquired using Kinect Skeletal Tracking. In the pointing gesture recognition module, when the right hand is kept stretched directing to any point, it is considered as a pointing gesture. The feature vector is formed using the right-hand joint coordinates. The joint coordinates are projected onto the X-Z plane to estimate the pointing direction.

After detecting the gesture starting point using the left hand, a feature vector is formed using the right-hand joint coordinates in the dynamic gesture recognition module. The gesture coordinates are translated using a reference point and then normalized before training and classification. 3D dynamic gesture recognition is performed using three different classifiers- Hidden Markov Models (HMM), Multiclass Support Vector Machines (SVM), and Convolutional Neural Networks (CNN). The Baum-Welch algorithm is used to train the HMM models. The HMM determines the likelihood of a particular sequence and determines the gesture class label of a captured gesture based on that likelihood estimation. One-against-one Multi-class Kernel Support Vector Machine Classifier is also used for gesture recognition. A Gaussian kernel function is applied for data transformation. A CNN with six convolutional layers is also used for dynamic gesture recognition. In the training phase, back-propagation with Stochastic Gradient Descent (SGD) is used. Two fully connected layers are used to classify the gestures. The Rectified Linear Unit function (ReLU) is used as an activation function to represent nonlinearities.

The proposed gesture adaptation module is a combined system of supervised and semi-supervised learning. If a performed gesture cannot be recognized three times by the system, the semi-supervised self-adaptation module is activated, and the unrecognized gesture is labeled automatically based on the best match to the existing gesture class labels implying a self-labeled semi-supervised learning method. If the unrecognized gesture does not match with any existing gesture class, user consent-based adaptation is activated. The system asks user interactions to label the data. To maintain performance, we imposed different weights on the training data for load balancing.

In the robot navigation subsystem, recognized pointing and 3D dynamic gestures are used to navigate a robot. Two different simple algorithms are developed for robot navigation which is tested in our simulator. To evaluate the system’s usability and flexibility, the training data is divided.

FIGURE 1. Block diagram of the proposed system.
into three sub-categories based on age which shows how the performance of the proposed system varies from person to person based on their age. The estimated age ranges are Young (below 25 years), Middle-aged (25-45 years), and Adult (above 25 years).

A. SKELETAL TRACKING USING KINECT SENSOR
As we mentioned earlier, Kinect is such an input device that is able to capture data and that can be used to recognize both static and dynamic gestures. It was first developed by Microsoft and released in 2010. It combines depth and RGB cameras into a compact sensor. Moreover, a human body can be represented by a number of joints using Kinect Skeletal Tracking. Thus, hand gesture recognition can be performed by tracking hand joints using a Kinect sensor.

The proposed framework’s first task is to track some specific human body joints of the user standing within the field of view of the Kinect sensor. We used Kinect Skeletal Tracking to track the required joints. If multiple people are present in front of the Kinect, the system tracks the first scanned skeleton and discards the rest. For the tracked skeleton, an array of joints provides the positions of the recognized human body joints in the 3D coordinate space. In the 3D coordinate system used by Kinect, one unit refers to 1 meter. For convenience, before using the coordinate value, we transformed it into centimeters.

Figure 2 shows the 2D image of a human standing in front of the Kinect along with his/her skeleton tracked by Kinect Skeletal Tracking.

![Figure 2. Skeletal tracking using Kinect.](image1)

B. POINTING GESTURE RECOGNITION
Our proposed system uses pointing gestures to navigate a mobile robot. After scanning the joints of the person standing in front of the Kinect, specific joints are used to determine whether the tracked person is performing a pointing gesture or not. If the pointing gesture detection is successful, the pointing direction is estimated using the coordinates of the hand and shoulder joints. This estimation is then used to navigate the mobile robot, which is described in detail in later sections.

1) POINTING GESTURE DETECTION
When someone’s right hand is kept stretched and directing any point, it is referred to as a pointing gesture in our framework. Only the right-hand along with elbow and shoulder joints are employed to identify pointing gestures. A person may not always perform pointing gestures while standing within Kinect’s field of view. To differentiate a pointing gesture from random hand movements, the right arm must maintain this specific posture.

Anyone is considered to be performing a pointing gesture only when he/she stands within the Kinect’s field of view, fully stretching their right arm, and keeping it parallel to the ground. To detect whether the arm is fully stretched or not, the hand, elbow, and shoulder joints are checked to determine whether they are on the same vertical level. Furthermore, a user may not always hold their body joints mentioned on exactly the same vertical level while performing a pointing gesture. Considering this fact, the above-mentioned criterion is relaxed to some extent. The person is still considered to be making a pointing gesture not only when the hand and shoulder joints maintain a vertical distance within 10 cm but also when the elbow and shoulder joints maintain a vertical distance within 5 cm. An example of how a user can perform a pointing gesture is shown in figure 3.

![Figure 3. A user is performing a pointing gesture.](image2)

2) POINTING DIRECTION ESTIMATION
When a user performs a pointing gesture, the hand and shoulder joint positions on the X-Z plane are used to estimate the direction in which the user is actually pointing. This is done by drawing a straight line across the hand and shoulder joint coordinates on the X-Z plane. We use the X- and Z-coordinates of the joints and discard the Y-coordinate because the height of the joints is unnecessary.
for estimating the direction. $X_s$ and $Z_s$ refer to the X- and Z-coordinates of the shoulder joint respectively and $X_h$ and $Z_h$ denote the X and Z coordinates of the hand joint respectively. Subsequently, we calculate the slope $m$ and $Z$ intercept $c$ of the line which goes through the hand and shoulder joints.

$$m = \frac{Z_s - Z_h}{X_s - X_h} \quad (1)$$

$$c = Z_h - (m \times X_h) \quad (2)$$

Equation (3) refers to the line that corresponds to the pointing direction on the X-Z plane.

$$Z = mX + c \quad (3)$$

C. 3D DYNAMIC GESTURE RECOGNITION

In order to control the robot, our suggested method employs 3D dynamic gestures. The Kinect sensor is able to control different aspects of the robot using predefined 10 commanding gestures: “Forward”, “Backward”, “Speed Up”, “Speed Down”, “Return”, “Stop”, “Follow”, “Unfollow”, “Free Movement” and “Go Behind”. The system is not limited to these ten dynamic gestures. The user can add any number of gestures to the system.

For dynamic gesture recognition, Kinect Skeletal Tracking is used to track left-hand, left-elbow and right-hand joints of the user. The coordinates of these joints are extracted, and the state of the left hand is also monitored using Kinect Skeletal Tracking. The state of the left hand determines the start and end of a gesture in the proposed system. The right-hand coordinates of the person performing the gesture are recorded between the starting and ending phases of the gesture. The recorded coordinates are then transformed into a 3D coordinate space by a reference point, so that every gesture sequence begins at that reference point in the 3D coordinate space. This task is accomplished so that a person can execute gestures while standing anywhere within Kinect’s range of view. Moreover, a person can stand in the same place but can start performing the same gesture from different points within their reach. If we use only the raw coordinates of the hand to detect a gesture, the same gesture may produce very different coordinate sequences that cannot be efficiently used in any classifier to classify the gesture. To address this issue, this module uses a reference point to convert all hand joint coordinates, such that comparable motions yield nearly related coordinate sequences.

The proposed system uses (210, 210, 210) (cm) as the reference point to translate the coordinates. It can be safely stated that a human standing in a fixed position and performing a hand gesture cannot move the hand by more than 200 cm in a single direction. Therefore, at any time, the coordinates of the hand joint on any axis are not expected to be less than 10 or greater than 410.

We translate the captured hand joint coordinates through the following calculations. We denote the new translated value by $T$ and the reference point (210, 210, 210) by $R$.

$$T = \text{First point in the sequence} - R$$

For every point in the generated sequence, we delete $T$ to translate these points by the reference points.

$$\text{New Point} = \text{Old Point} - T$$

The hand joint coordinate sequence of a simple gesture after translation by the reference point (210, 210, 210) is shown in Table 1.

| Raw X Coordinate | Translated X Coordinate | Raw Y Coordinate | Translated Y Coordinate | Raw Z Coordinate | Translated Z Coordinate |
|------------------|------------------------|------------------|------------------------|------------------|------------------------|
| 157.5            | 250.9                  | 164.1            | 157.5                  | 180.0            | 164.1                  |
| 159.5            | 250.9                  | 166.1            | 157.5                  | 180.0            | 164.1                  |
| 154.5            | 250.9                  | 166.1            | 157.5                  | 180.0            | 164.1                  |
| 155.5            | 250.9                  | 166.1            | 157.5                  | 180.0            | 164.1                  |

2) COORDINATE TRANSLATION

A reference point is used to translate the captured hand joint coordinate sequence. This is done so that a user can perform a gesture at any point within the Kinect’s field of view. It is possible for a person to stand at different positions in front of the Kinect while performing a single gesture. Moreover, a person can stand in the same place but can start performing the same gesture from different points within their reach. If we use only the raw coordinates of the hand to detect a gesture, the same gesture may produce very different coordinate sequences that cannot be efficiently used in any classifier to classify the gesture. To address this issue, this module uses a reference point to convert all hand joint coordinates, such that comparable motions yield nearly related coordinate sequences.

The proposed system uses (210, 210, 210) (cm) as the reference point to translate the coordinates. It can be safely stated that a human standing in a fixed position and performing a hand gesture cannot move the hand by more than 200 cm in a single direction. Therefore, at any time, the coordinates of the hand joint on any axis are not expected to be less than 10 or greater than 410.

We translate the captured hand joint coordinates through the following calculations. We denote the new translated value by $T$ and the reference point (210, 210, 210) by $R$.

$$T = \text{First point in the sequence} - R$$

Then for every point in the generated sequence, we delete $T$ to translate these points by the reference points.

For every point in the sequence, New Point = Old Point - $T$

The hand joint coordinate sequence of a simple gesture after translation by the reference point (210, 210, 210) is shown in Table 1.
3) COORDINATE NORMALIZATION
To normalize these coordinate values into discrete values, we divide these values using a fixed integer number, which is 20. As a result, we can obtain at most 21 discrete values ranging from 0 to 20 for every coordinate on the X, Y, and Z axes for all hand points in the sequence. We denote the normalized coordinate value as \( V_{\text{normalized}} \) and the original coordinate value \( V_{\text{original}} \), and the normalization of coordinates can be expressed using equation (4).

\[
V_{\text{normalized}} = \frac{V_{\text{original}}}{20} \tag{4}
\]

The reference point (200, 200, 200) could have been used instead of (210, 210, 210). However, this would have made the feature vectors of gestures that do not involve hand movements along a certain axis inconsistent, as Kinect can detect even the slightest hand movements. Consider a gesture that does not involve hand movements along the X-axis. Therefore, the recorded X-axis coordinates of the gesture should be approximately 200. But humans cannot keep their hands stationary. Thus, while performing that gesture, the recorded X-coordinates of the hand joint are often approximately 197-203. As we divide these coordinate values by 20.0, for normalization, using 200.0 as the reference point would make the feature vector have a combination of 10s and 9s as the recorded hand coordinates along the X-axis may read different values varying from 197-203. This makes the classification difficult for the classifier. To ensure that the feature vector stays consistent, a reference point (210, 210, 210) has been selected so that values between 207-213 would result in 10 after normalization, thus removing the effect of slight hand movements.

Using equation 4, three different vectors \( V_x \), \( V_y \), and \( V_z \) containing the normalized values of the hand joint coordinates on the X, Y, and Z axis respectively of a gesture sequence are generated. Table 2 shows the different values generated after normalizing the translated hand joint coordinates from table 1.

| X Coordinate | Normalized X Coordinate \((\lambda)\) |
|--------------|----------------------------------|
| 116.64       | 10                               |
| 116.43       | 10                               |
| 116.63       | 10                               |
| 116.62       | 10                               |
| 116.61       | 10                               |
| 116.60       | 10                               |
| 116.69       | 10                               |
| 116.68       | 10                               |
| 116.67       | 10                               |
| 116.66       | 10                               |
| 116.65       | 10                               |

4) FEATURE VECTOR GENERATION
A feature vector is referred to as a collection of numerical values, where each value represents a measurable characteristic of an object. The normalized coordinate sequence from the previous module is utilized to generate the feature vector \([29]\). From Table 2, all the normalized X, Y, and Z coordinate values are used to construct \( V_x \), \( V_y \), and \( V_z \), respectively, as mentioned before. Then, these vectors are appended together to form a new vector \( V_{\text{feature}} \) which is the feature vector used for model training or classification.

5) GESTURE CLASSIFICATION USING HIDDEN MARKOV MODEL
Hidden Markov Model (HMM) is a statistical probabilistic model. It can be applied to perceive sequences in homogeneous datasets, learn patterns, and recognize them. HMM is convenient for modeling systems that are considered to reside in a particular state from a set of defined states, and the states periodically switch from one state to the next. HMM has a property called the Markov property which implies that the probability distribution of future states depends only on the current state of the process instead of the previous states [30]. Hidden Markov Model has the term “Hidden” in its name because the state of the system is not explicit. At a given period, neither the current state of the system nor the state transitions that have already occurred can be observed directly. As a result, the actual states of the model need not be specified rather than the number of required states. The model determines the initial states and the sequence of states by itself to satisfy the training data.

A Hidden Markov Model consists of some parameters. If we denote a HMM as \( \lambda \), then we need to know the following parameters to define \( \lambda \) [31].

1) \( N \): Number of states \( S = \{ S_1, S_2, S_3, \ldots, S_N \} \), Here \( S \) is the set of states
2) \( M \): Number of observation symbols \( V = \{ v_1, v_2, v_3, \ldots, v_M \} \). Here \( V \) is the set observation symbols
3) \( X \): State transition probabilities \( X = x[i, j] \) where \( x[i, j] = P(Q_{t+1} = S_j | Q_t = S_i) \); \( i, j \in [1, N] \) Here, \( Q_t \) = state at time \( t \)
4) \( Y \): State emission probabilities \( Y = y(k) \) where \( y(k) = P(G_t = V_i | Q_t = S_j) ; j \in [1, N] \) and \( k \in [1, M] \) Here, \( Q_t \) = state at time \( t \) and \( G_t \) = observation at time \( t \)
5) \( \pi \): Initial state probabilities \( \pi = p_i \) where \( p_i = P(Q_1 = S_i) ; i \in [1, N] \) Here, \( Q_1 \) = initial state of the system

Formally a Hidden Markov Model can be interpreted through the following expression.

\[
\lambda = (\pi, X, Y) \tag{5}
\]

a: TRAINING AND CLASSIFICATION
In order to train an HMM, the model parameters \( \lambda = (\pi, X, Y) \) are optimized using a set of observation sequences so that the model can describe those sequences efficiently. These observation sequences are used to train the model and so can be named as training sequences. In this paper, the Baum-Welch estimation algorithm [32] is employed to train the model which uses the training set of observation or gesture sequences and apply the Expectation-Maximization algorithm to a HMM(\( \lambda \)) to estimate the parameters of \( \lambda \). Given an
gesture classes. Therefore, we got a set of
represented in figure 5, let’s assume the problem domain has
arate HMMs are used for every gesture class and trained
they can classify any input sequence. As stated earlier, sep-
likelihood can be used to derive the class label of that gesture
is created and trained entirely using training sequences from
P
so that
a linear model that does not depend on the sample of data.
nonlinear transformation to interpret a nonlinear model into
This operation is often computationally more efficient than
has a clear dividing margin between classes of data [33].
trick transforms nonlinear data into another dimension that
of algorithms that are used for pattern analysis. The kernel
for dynamic gesture recognition. Kernel methods are a class
space. We incorporate Multi-class SVM with Gaussian kernel
plane, they become linearly separable in a higher-dimensional
However, when the data are not linearly separable in the X-Y
hyperplane are known as the support vectors. When the train-
by a hyperplane. The vectors (instances) that define the
Multi-class Kernel Support Vector Machine Classifier for
Our proposed framework implements One-against-one
SUPPORT VECTOR MACHINE
Gesture classification with HMM.

6) GESTURE CLASSIFICATION USING MULTI-CLASS SUPPORT VECTOR MACHINE
Our proposed framework implements One-against-one Multi-class Kernel Support Vector Machine Classifier for dynamic gesture recognition. SVM is a discriminative classifier that classifies the given labeled data separated by a hyperplane. The vectors (instances) that define the hyperplane are known as the support vectors. When the training data are linearly separable, the hyperplane divides the plane into two parts, containing only one class on each side. However, when the data are not linearly separable in the X-Y plane, they become linearly separable in a higher-dimensional space. We incorporate Multi-class SVM with Gaussian kernel for dynamic gesture recognition. Kernel methods are a class of algorithms that are used for pattern analysis. The kernel trick transforms nonlinear data into another dimension that has a clear dividing margin between classes of data [33]. This operation is often computationally more efficient than explicit computation of the coordinates [34]. It follows a fixed nonlinear transformation to interpret a nonlinear model into a linear model that does not depend on the sample of data. Therefore, any learning theory that exists for linear models automatically transforms to a nonlinear version [35]. One-against-one Multi-class Kernel Support Vector Machine classifier is used for gesture recognition in this study. We trained N(N-1)/2 binary classifiers for N different gesture classes. Each classifier is trained with samples of class pairs from the original training data so that they can distinguish between the two gesture classes. When a new gesture is performed, it is applied to all N(N-1)/2 classifiers, and a voting scheme is also applied [36]. The gesture class with the highest number of votes is the result of the combined SVMs.

a: TRAINING AND CLASSIFICATION
Let
• x be a feature vector (i.e., the input of the SVM), \(x \in \mathbb{R}^n\),
  where \(n\) is the dimension of the feature.
• y be the class (i.e., the output of the SVM), \(y \in \{-1, 1\}\),
  i.e. the classification task is binary.
• w and b are the parameters of the SVM: we need to learn
  them using the training set.
• \((x(i), y(i))\) be the \(i^{th}\) sample in the dataset. Let’s assume
  we have \(N\) samples in the training set.
In classification phase, SVM tries to satisfy two requirements.
1. SVM should maximize the distance between the two decision boundaries. Which means the distance between the two hyperplane defined by \(w^T x + b = -1\) and the hyperplane defined by \(w^T x + b = 1\).
2. SVM should correctly classify all \(x(i)\), which means:
   After introducing Slack variables for error calculation (when a data point is on the wrong side) and a kernel function \(\phi\) that maps the original feature space to a higher dimensional feature space, non-linear decision boundaries are possible. quadratic optimization problem becomes:

\[
\min_{w, b} \frac{\|w\|^2}{2} + C \sum_{i=1}^{N} \xi(i), \\
\text{s.t. } y(i)(w^T \phi(x(i)) + b) \geq 1 - \xi(i), \quad \forall i \in \{1, \ldots, N\} \\
\xi(i) \geq 0, \quad \forall i \in \{1, \ldots, N\} 
\]
After learning \(\alpha(i)\) from the training set, we can predict the class of a new sample with the feature vector \(x^{test}\) as follows:

\[
y^{test} = \text{sign}(w^T \phi(x^{test}) + b) = \text{sign}\left(\sum_{i=1}^{N} \alpha(i)y(i)\phi(x(i))^T \phi(x^{test}) + b\right)
\]

7) GESTURE CLASSIFICATION USING CNN
Convolutional Neural Network is the most popular deep learning architecture and the reason is it can detect important features without any human supervision [37]. It can learn distinctive features for each class all by itself. The convolution and pooling layers are the main features of CNN that make it different from other deep learning architectures [38].
Suppose, CNNs consists of convolutional layers which are defined by an input map $I$, a set of filters $K$ and biases $b$. The output from this convolution procedure is given in equation 6.

$$\sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} K_{m,n} \cdot I_{i+m,j+n} + b$$ (6)

For a total of $P$ predictions, the predicted network outputs $y_p$ and their corresponding targeted values $t_p$ the mean squared error is given by equation 7.

$$\frac{1}{2} \sum_{p} (t_p - y_p)^2$$ (7)

We used 3,600 instances of 10 pre-defined gestures each age range having 1,200 training instances. In order to train CNN with our training instances, we configured the parameters as follows.

- Length of feature: 90
- Batch size: 10
- Iteration: 360
- Epoch: 3
- Learning Rate: 0.01
- Optimizer: Stochastic Gradient Descent

In this system, six convolutional layers are used and the network is trained and the backpropagation is performed with stochastic gradient descent (SGD). We selected our batch size to be 10, which means randomly 10 gesture data will be selected from the training data for training. This is the ‘stochastic’ part of SGD. During a single forward and backward pass of training, the network updates its parameters according to the elements in the batch. Since SGD is noisy, it helps to reduce the chance of overfitting. Moreover, we performed some preprocessing tasks on the data; therefore, data augmentation is not necessary in our case. Here, a regularization method softmax is used, and the dropout probability was 0.8. A neuron is dropped at each iteration which gets activated in the next step after resampling. It also contributes to lessening overfitting issues, as each neuron can contribute independently. Two fully connected layers are used to classify the gestures.

D. GESTURE ADAPTATION

In order to make a gesture recognition system applicable to robots in various fields, the number of gestures should not be limited. The proposed system implements gesture adaptation using supervised and semi-supervised learning. A user can add any number of gestures, and new gesture data can be included in the existing gesture training data through user consent; this is referred to as automatic adaptation. A self-labelled semi-supervised classification technique is also implemented for the growth of the training data. Weights are imputed on the training data in various phases to maintain the gesture subclasses and system accuracy. Figure 6 shows the block diagram of the proposed gesture adaptation module.

1) ADAPTATION THROUGH SUPERVISED LEARNING

As supervised learning methods use labeled data for training, this adaptation module requires complete user interaction. Users can add a new instance of an existing gesture class or a new gesture class label. In both cases, the user must label data on the fly using the developed Graphical User Interface (GUI). This framework comes of ten pretrained 3D dynamic gestures, where five gestures are simple and the rest are complex. The user can add any number of gestures to the system using the GUI. It provides a dedicated button for adding a new gesture. The user is prompted to enter the class label and gesture name before performing a new gesture. Subsequently, the user must perform a gesture in front of Kinect, and a new feature vector is generated from the captured gesture data. It is added to the training data automatically with the maximum weight for that gesture class. Then, the classifiers are trained again on the entire training data and become eligible to recognize new gesture classes afterwards.

2) ADAPTATION THROUGH SEMI-SUPERVISED LEARNING

In real world, the scarcity of labeled data is acute and labeling with human supervision is even more difficult and costly. So, semi-supervised learning methods have proven useful. Clustering is the most common semi-supervised procedure for labeling unlabeled data. Owing to excessive false positives, the performance of classifiers gets hampered. Therefore, a self-labeled semi-supervised method is implemented, which does not make any specific assumptions about the input data. It aims to obtain one or several enlarged labeled sets based on their most confident predictions to classify the unlabeled data. This semi-supervised-based adaptation module is activated in our system if a performed gesture cannot be recognized by a classifier.
a: SELF ADAPTATION

When a user performs three unrecognized gestures, the system asks if they want to add that gesture as a new gesture to the system. The gestures are not necessarily to be performed consecutively. Our adaptation module tracks only unrecognized gestures. Whenever the number of performed unrecognized gestures becomes three, the system asks to add the gesture as a new one. The self-adaptation module is activated after the consent of the user. New feature vectors are generated for the three gestures, and the proposed adaptation module recreates a new HMM-based classifier model with the three unrecognized gestures. The classifier is trained with them using 3-fold cross-validation method.

If all three unrecognized gestures are different, that is, unmatched, then the system will reject the gesture adaptation, considering it as a random or mistakenly performed gesture. On the other hand, if all the three gestures match, this means that the gestures were not performed mistakenly. Therefore, new feature vectors are generated, and the gestures are classified using any primary classifiers (HMM, SVM, CNN) that were trained with the original training data. The primary classifier compares the log-likelihood values of the new unlabeled gesture data with existing gesture data. But this time the log-likelihood threshold is more relaxed so that the unrecognized gesture can be classified. The new gesture is then assigned to one of the existing class labels that has the maximum match with the new gestures. New gestures are added as a subclass of the main gesture class with a lower weight value. If the log-likelihood value is below the relaxed threshold during the second classification phase, the new gesture cannot be added to the existing gesture classes. At this point, user consent-based adaptation module is activated.

b: USER CONSENT BASED ADAPTATION

When an unrecognized gesture cannot be added through self-adaptation, user consent-based adaptation is automatically triggered. Our system will require little user interaction to adapt to the gesture. When a user performs three new gestures, a secondary HMM is trained with those using 3-fold cross-validation method. If the performed three gestures match with each other, that means they were performed intentionally. Now, if they fail to go through the second classification phase of the self-adaptation module, it will refer the gestures as completely new. The system will ask the user to input the new gesture label. If a user inputs a new gesture label, the gesture is immediately added to the current training data under the gesture label with the highest weight. If the user inputs an existing gesture label, it means that the user insists on adding a completely new gesture to an already existing class. So, the gesture is added to that class with a lower weight so that the performance of the system is maintained.

c: LOAD BALANCING

The training data will continue to increase as users continue to use the system, and new gesture instances will be added through self-adaptation with lower likelihood values. Moreover, the number of adapted instances is much greater than that of the original gesture instances, which can lead to a data imbalance problem [39]. There are several methods to classify imbalanced data [40]. However, we have implemented a different and more effective method. Weights are used to maintain the subclasses of the gesture, which helps maintain the accuracy of the system. The original training data have a weight value of 3. When a user performs a gesture and it gets recognized, the gesture is added to the training dataset with a certain weight \( w_r \). In the gesture adaptation module, an unrecognized gesture is tested again using a relaxed log-likelihood threshold. If the unrecognized gesture gets recognized in the adaptation module, it is added to the training data. If the gesture belongs to a pre-existing gesture class, it is associated with a discounted weight value of \( w_r \). This is because it may decrease system accuracy if it obtains the same priority as the original training data. Figure 7 shows the abstract gesture classes of the proposed system.

Each gesture data has a weight value \( N \) associated with it and the same repeated gesture data is used \( N \) times during training dynamically. So, the original training data are not affected by repeated data, yet the system obtains more data to learn. However, the risk of overfitting data still may exist. So we put a little noise when the classifiers are trained. In this way, the original training data remain noise-free, and the system does not suffer from overfitting.

E. GESTURE DOMAIN

Our ten predefined dynamic gestures are divided into two categories: five simple and five complex gestures, as shown in Figures 8 and 9 respectively. The proposed system was trained using these ten predefined gestures.
FIGURE 8. Simple gesture domain.
FIGURE 9. Complex gesture domain.
**F. ROBOT NAVIGATION**

Both pointing and dynamic gesture recognition are required in order to navigate a robot. At first, a user has to perform any dynamic gesture and give the robot a command using that gesture. The robot will not start moving until it recognizes the pointing direction from the user. When the user points to a direction with their right hand, the robot performs the corresponding action specified by the dynamic gesture after going to that pointed direction. When a pointing gesture is recognized, the robot navigation module navigates the robot using the predicted pointing direction, which is represented as a straight line on the X-Z plane described in equation 3. In our study, two strategies are used for driving the robot in the pointing direction.

1) **METHOD 1**

A boundary is set around the Kinect and the robot is permitted to navigate within it. The intersection point of line (3) and any of the boundary lines that appear first in the route from the user in the pointing direction are then computed, and the robot proceeds towards that point of intersection. Assuming that the boundary lines that are to the left, right, and behind the Kinect are respectively A, B, and D units away from the Kinect. Then, the boundary line to the right of the Kinect is perpendicular to the Kinect’s X-axis, and it can be defined using \( X = A \). The boundary line which is situated to the left of the Kinect is also perpendicular to its X-axis and it can be defined using \( X = -B \). The boundary line behind the Kinect is perpendicular to its Z-axis; therefore, it can be defined using \( Z = -D \).

Then it is determined which of these three lines intersects with line (3) first towards the pointing direction and the robot approached towards the intersection point following a straight path. We denote the X coordinates of the shoulder and hand joints as \( X_s \) and \( X_r \) respectively, and the slope and Z intercept of the line (3) as \( m \) and \( c \) respectively. Figure 10 can be used to visualize this method more clearly.

![FIGURE 10. Robot navigation using method 1.](image)

The overall approach of navigating the robot using this method can be described using algorithm 1.

Algorithm 1 Robot Navigation Using Method 1

```plaintext
while Pointing Gesture is being performed do
    \( dx = X_h - X_s \)
    if \( dx = 0 \) then
        \( X_{\text{intersection}} = X_h \)
        \( Z_{\text{intersection}} = -D \)
    else if \( dx > 0 \) then
        \( Z_{\text{intersection}} = m \times A + c \)
        if \( Z_{\text{intersection}} < -D \) then
            \( X_{\text{intersection}} = (Z_{\text{intersection}} - c) / m \)
        else
            \( X_{\text{intersection}} = A \)
        end if
    else
        \( Z_{\text{intersection}} = m \times (-B) + c \)
        if \( Z_{\text{intersection}} < -D \) then
            \( X_{\text{intersection}} = -D \)
        \( X_{\text{intersection}} = (Z_{\text{intersection}} - c) / m \)
        else
            \( X_{\text{intersection}} = -B \)
        end if
    end if
end while
```

2) **METHOD 2**

In the second method, the robot initially fixes its location by intersecting the line of pointing direction indicated in equation 3. From there, the robot follows the line perpendicular to equation 3. We denote Robot’s current location on the XZ plane as \((X_r, Z_r)\) where \(X_r\) and \(Z_r\) refer X and Z coordinate respectively. Equation (3) can be rewritten as follows:

\[ Z = m_1X + c_1 \] (8)

Equation 9 is perpendicular to equation 8 that passes through robot’s current position:

\[ Z = m_2X + c_2 \] (9)

Here, \(m_2\) and \(c_2\) are the slope and Z intercept of line (9). Where,

\[ m_2 = -(1/m_1) \] (10)
\[ c_2 = Z_r - (X_r \times m_2) \] (11)

The intersection point \((X_{\text{intersection}}, Z_{\text{intersection}})\) between lines ((8) and (9)) is then obtained, and the robot travels towards that intersection point from its present location \((X_r, Z_r)\) until it is less than or equal to 10 cm away. While pointing to one direction, the pointing arm may continuously shake and move left and right along the X-axis of the Kinect. As the Kinect can detect even the slightest hand movements, if the robot is made to move to the exact intersection point, the robot may have to adjust its position every now and then. This
slight relaxation of the rule avoids this scenario. Figure 11 exhibits as described above.

**FIGURE 11.** Robot position modification.

As soon as the robot gets close to the intersection point \((X_{intersection}, Z_{intersection})\), it starts to move towards the actual pointed direction. If the performed gesture is ‘Forward’, the robot goes to destination following line (8). If the performed gesture is ‘Backward’, the robot comes to the user following line (8). These two scenarios are shown in Figure 12 (a, b).

**FIGURE 12.** Robot navigation using method 2.

The overall approach of navigating the robot using this method can be described using algorithm 2.

**Algorithm 2** Robot Navigation Using Method 2  
1: while Pointing Gesture is being performed do  
2: \(X_{intersection} = (c_2 - c_1)/(m_1 - m_2)\)  
3: \(Z_{intersection} = (m_1 \times X_{intersection}) + c_1\)  
4: while Distance between intersection point and robot > 10 do  
5: Move robot towards intersection point \((X_{intersection}, Z_{intersection})\)  
6: end while  
7: if Mode = Forward then  
8: Move robot towards the pointing direction  
9: else  
10: Move robot towards the user  
11: end if  
12: end while

**IV. EXPERIMENT SETUP FOR 3D GESTURE RECOGNITION, ADAPTATION AND ROBOT NAVIGATION**

An extensive experiment was carried out using a computer system equipped with an Intel Core-i5 CPU, 8 GB RAM, and a Windows 10 operating system. To build the system, we chose C# as the programming language, and Microsoft Visual Studio Community 2019 as the integrated development environment. The Kinect Software Development Kit 2.0 was utilized to facilitate gesture recognition utilizing the Kinect sensor. The SDK includes the tools and application programming interfaces (APIs) needed to create Kinect-enabled apps for Microsoft Windows. The Kinect SDK supports the functionality of Kinect, such as color pictures, depth images, voice input, and skeletal data.

**A. SIMULATOR**

We created a simulator to monitor system behavior and assess its performance. The simulator was created in C++ using OpenGL graphics API. OpenGL is a collection of functions that provides the general interface needed to make use of the various features provided by graphics hardware.

**FIGURE 13.** Simulator for robot navigation.

Figure 13 depicts a visual representation of the simulator, with horizontal and vertical lines representing the X- and Z-axes, respectively. On the XZ plane, the red and green squares represent the user and the robot respectively. The red line drawn from the source or user denotes the direction on the XZ plane in which the user is pointing. When Method 1 is used to drive the robot, the cross (X) mark represents the intersection point between the pointing line and a boundary line.

**B. GRAPHICAL USER INTERFACE (GUI)**

A Graphical User Interface (GUI) has been created to allow users to easily interact with the system. Users may utilize the GUI to operate the system and alter many variables inside it [41]. Figure 14 shows the layout of this interface.

The GUI has a dedicated button for turning the Kinect sensor on or off. Once Kinect is turned on, the gesture detection module is activated. For the dynamic commanding gesture recognition module, there are two buttons named “Change to Training Mode” and “Change to Recognition Mode” that can be used to change the mode of operation of the system. When the system is in training mode, the GUI provides a drop-down menu from which a user can select which gesture...
the system is going to be trained; however, in recognition mode, this drop-down menu will not be visible.

Three more buttons named “Save Training Data”, “Load Training Data” and “Add The Recognized Gesture” were included that can be used to save the training dataset in a text file, load previously stored training dataset from that text file and add new gestures during adaptation process respectively. Then a user can chose one of three classifiers to be applied on the training data through the buttons named “Train HMMs”, “Train SVM” and “Train CNN”.

Again, for manual gesture adaptation, the “Add New Gesture” button is to be clicked and it requires user consent to define a new label. After entering the input, it automatically adds the given gesture to the gesture list. Figure 14, 15, and 16 exhibits the screenshots of the interfaces.

To control a robot using a pointing gesture, a drop-down menu titled “Robot Navigation Method” is added at the bottom that may be used to pick any of the robot navigation techniques outlined earlier in this section. If “Method 1” is chosen, the GUI shows three text-fields for adjusting the distance between the boundary lines and the Kinect. These text fields are buried when using “Method 2” to navigate the robot, because they are meaningless in this scenario. There is a button in this drop-down menu called “Start Simulator,” which, when clicked, activates the simulator.

The Current navigation status of the robot and the dynamic gesture recognition module are displayed in the center of the GUI labeled as “Robot Status” and “Gesture Recognition.” When a pointing gesture is recognized, the robot starts moving, and the “Robot Status” label changes the text status to “Moving.” If the robot stays idle, the label shows the text “Idle.” The “Gesture Recognition” label displays “Executing Gesture” as the state when performing a dynamic gesture, and after the gesture gets recognized, it shows the name of that gesture.

V. RESULTS AND PERFORMANCE EVALUATION

To test the system behavior and analyze its performance we used several performance metrics such as average response time, precision, true positive (TP) rate or recall, false positive (FP) rate, F-measure and accuracy. All the four modules (1. Pointing Gesture Recognition Module, 2. Dynamic Gesture Recognition Module, 3. Gesture Adaptation Module, 4. Robot Navigation Module) were tested individually to carry out experiments for the proposed system.

A. PERFORMANCE ANALYSIS OF POINTING GESTURE RECOGNITION

This module extracts necessary joint data using Kinect SDK and tries to detect whether a pointing gesture is being performed by a user standing in front of the Kinect. Whenever a pointing gesture is detected, the module uses the extracted hand and shoulder joint coordinates to estimate the pointing direction using the method and stores the necessary values for use in the robot navigation module.

After detecting the pointing hand instantly by Kinect skeletal tracking, the pointing direction is estimated by simple
geometric calculations performed on the fly. So pointing gesture recognition occurs in real-time.

**B. PERFORMANCE ANALYSIS OF DYNAMIC GESTURE RECOGNITION**

This module is responsible for training HMMs, SVMs and CNNs using training datasets and for classifying gestures using the trained models. This module also executes an action associated with a recognized dynamic commanding gesture.

HMMs, SVMs, and CNNs must be trained first in order to recognize dynamic gestures. We invited 24 volunteers to execute the 10 gestures in our gesture domain, 15 times each to produce the training dataset. Consequently, a training dataset of 3,600 gesture sequences was created. While completing a gesture, the right label of the gesture was picked from the drop-down choice provided by the GUI’s ‘Training Mode,’ and all training gesture occurrences were labeled appropriately for training as a consequence.

This part describes the performance analysis for dynamic gesture recognition. The performance evaluation part is divided into three age categories: Young (below 25 years), Middle-aged (25-45 years) and Adult (above 45 years).

1) **CONFUSION MATRIX FOR HMM**

The confusion matrix with ten class labels for three age groups is shown in table 3, 4 and 5. In the confusion matrix, each of the 10 dynamic gestures is represented by a class label. Each row represents the actual class label, while each column represents the predicted class label. As the total number of gesture instances is 3600, dividing them into three age categories results into 1200 gesture instances for each age category. As a result, each gesture class under any age category has 120 gesture instances. Precision, recall, F-measure, specificity, false positive rate and accuracy are calculated considering all the directly recognized gestures as well as self-adapted gesture instances for each gesture class.

**TABLE 3. Confusion matrix for HMM (young; below 25 years).**

|       | Forward | Backward | Speed Up | Speed Down | Stop | Open | Close | Return | Follow | Total |
|-------|---------|----------|----------|------------|------|------|-------|--------|--------|-------|
| Forward | 116     | 4        | 0        | 112        | 2    | 0    | 0     | 0      | 0      | 116   |
| Backward| 4       | 116      | 0        | 0          | 112  | 2    | 0     | 0      | 0      | 116   |
| Speed Up| 0       | 0        | 116      | 4          | 112  | 2    | 0     | 0      | 0      | 116   |
| Speed Down| 0     | 0        | 0        | 116        | 112  | 2    | 0     | 0      | 0      | 116   |
| Stop    | 2       | 0        | 0        | 0          | 116  | 4    | 0     | 0      | 0      | 118   |
| Open    | 0       | 2        | 0        | 0          | 0    | 116  | 4     | 0      | 0      | 118   |
| Close   | 0       | 0        | 0        | 0          | 0    | 0    | 116   | 4      | 0      | 116   |
| Return  | 0       | 0        | 0        | 0          | 0    | 0    | 0     | 116    | 4      | 116   |
| Follow  | 0       | 0        | 0        | 0          | 0    | 0    | 0     | 0      | 116    | 116   |
| Total   | 118     | 118       | 118       | 118        | 118  | 118  | 118   | 118    | 118    | 1200  |

Consider ‘Forward’ gesture in table 3, out of 120 performed gesture instances, 116 instances are labeled as ‘Forward’ indicating 116 true positives and 4 instances falsely labeled as ‘Speed Down’ indicating 4 false negatives. 1 ‘Speed Down’ gesture instance is falsely labeled as ‘Forward’ indicating 1 false positive. The ‘Stop’ gesture is labeled as ‘Stop’ for 120 instances out of 120 indicating 120 true positives and so 0 false negatives. No other gesture is labeled as ‘Stop’ indicating there is no false positive for ‘Stop’ gesture.

Both ‘Follow’ and ‘Return’ gestures start from the same hand position and so they have similarity at the beginning of their gesture sequences. As a result, some misclassification occurred between ‘Follow’ and ‘Return’ in table 4. Due to the simplicity of ‘Stop’ gesture, the volunteers could easily perform the gesture and all the 120 instances got correctly labeled. The number of true positives for rest of the gesture classes are almost similar to the confusion matrix shown in table 3. This signifies that the gesture sequences for young and middle-aged people have similar patterns; hence, they can be modeled invariantly by HMM.

**TABLE 4. Confusion matrix for HMM (middle-aged; 25-45 years).**

|       | Forward | Backward | Speed Up | Speed Down | Stop | Open | Close | Return | Follow | Total |
|-------|---------|----------|----------|------------|------|------|-------|--------|--------|-------|
| Forward | 116     | 4        | 0        | 112        | 2    | 0    | 0     | 0      | 0      | 116   |
| Backward| 4       | 116      | 0        | 0          | 112  | 2    | 0     | 0      | 0      | 116   |
| Speed Up| 0       | 0        | 116      | 4          | 112  | 2    | 0     | 0      | 0      | 116   |
| Speed Down| 0     | 0        | 0        | 116        | 112  | 2    | 0     | 0      | 0      | 116   |
| Stop    | 2       | 0        | 0        | 0          | 116  | 4    | 0     | 0      | 0      | 118   |
| Open    | 0       | 2        | 0        | 0          | 0    | 116  | 4     | 0      | 0      | 118   |
| Close   | 0       | 0        | 0        | 0          | 0    | 0    | 116   | 4      | 0      | 116   |
| Return  | 0       | 0        | 0        | 0          | 0    | 0    | 0     | 116    | 4      | 116   |
| Follow  | 0       | 0        | 0        | 0          | 0    | 0    | 0     | 0      | 116    | 116   |
| Total   | 118     | 118       | 118       | 118        | 118  | 118  | 118   | 118    | 118    | 1200  |

The confusion matrix in table 5 signifies the change in patterns in gesture sequences for comparatively older people. ‘Forward’ and ‘Speed Down’ gestures have similarity in hand motion direction. This similarity was insignificant in table 3 and 4 because of the rigid gesture performance of young and middle-aged individuals. On the contrary, the similarity is notable in 5 because some of the adult people could not differentiate between ‘Forward’ and ‘Speed Down’ gestures properly without any prior training. As a result, the number of true positives for ‘Forward’ gesture got decreased and more ‘Speed Down’ gestures were recognized falsely as ‘Forward’ giving a raise in the false positive for ‘Forward’ gesture. Both ‘Return’ and ‘Speed Up’ gestures require hand motion from bottom to top at the beginning. As a result, a significant number of ‘Return’ gesture was falsely labeled as ‘Speed Up’. A probable reason behind this can be sudden interval while performing ‘Return’ gesture which generates sequences like ‘Speed Up’. Sometimes this sequence has higher likelihood value compared to actual ‘Speed Up’ class, and gets labeled as ‘Speed Up’. This has occurred more for adult people than young or middle-aged people.

2) **PRECISION FOR HMM**

Table 6 shows the comparison of precision of the HMM classifier for the ten gestures performed by the volunteers from three age categories. Precision provides the percentage of the gestures that are labeled as positive, are indeed positive. As an example, there are 120 ‘Backward’ gesture instances in table 3. Out of the 120 instances, HMM labeled 114 instances as ‘Backward’ indicating 114 true positives and 6 instances as ‘Speed Up’ indicating 6 false negatives. 3 other gesture
instance was labeled as ‘Backward’ indicating 3 false positives. As a result the ‘Backward’ gesture achieves a precision score of 0.97 due to the 3 false positives. False negatives do not get reflected in precision table. The middle-aged category has slightly better precision overall.

**TABLE 6. Precision comparison for 10 dynamic gestures using HMM.**

| Gesture          | Young (Below 25 years) | Middle-aged (35-45 years) | Adult (Above 45 years) |
|------------------|------------------------|---------------------------|------------------------|
| Forward          | 0.99                   | 0.96                      | 0.96                   |
| Backward         | 0.97                   | 0.99                      | 0.89                   |
| Speed Up         | 0.91                   | 0.97                      | 0.85                   |
| Speed Down       | 0.90                   | 0.95                      | 0.89                   |
| Return           | 0.96                   | 0.98                      | 0.92                   |
| Follow           | 0.91                   | 0.99                      | 0.88                   |
| U-turn           | 0.99                   | 0.98                      | 0.89                   |
| Free Movement    | 0.96                   | 0.95                      | 0.91                   |
| Go Behind        | 0.98                   | 0.95                      | 0.88                   |
| Stop             | 1.00                   | 1.00                      | 1.00                   |

A precision column chart for 10 dynamic gestures recognized using HMM is shown in figure 17.

3) **RECALL OR SENSITIVITY OR TP RATE FOR HMM**

Table 7 provides recall or sensitivity or true positive rate of dynamic gesture recognition using HMM classifier. According to the confusion matrices in table 3, 4 and 5 some of the ‘Free Movement’ gesture instances were wrongly classified as ‘Return’ and ‘Follow’ due to the complexity and similarity of the three gestures. As a result, ‘Free Movement’ gesture has more false negatives compared to other gestures. As calculation of recall gets affected by false negatives, recall score for ‘Free Movement’ gesture is comparatively low. ‘Go Behind’ gesture has pretty low false negatives despite being a complex gesture as it has less similarity with other gestures that start from the same position and as a result achieved higher recall score.

Figure 18 shows a column chart for the recall scores of dynamic gesture recognition using HMM.

**TABLE 7. Recall comparison for 10 dynamic gestures using HMM.**

| Gesture          | Young (Below 25 years) | Middle-aged (35-45 years) | Adult (Above 45 years) |
|------------------|------------------------|---------------------------|------------------------|
| Forward          | 0.99                   | 0.97                      | 0.93                   |
| Backward         | 0.95                   | 0.97                      | 0.93                   |
| Speed Up         | 0.97                   | 0.95                      | 0.88                   |
| Speed Down       | 0.94                   | 0.96                      | 0.85                   |
| Return           | 0.91                   | 0.97                      | 0.97                   |
| Follow           | 0.95                   | 0.91                      | 0.91                   |
| U-turn           | 0.98                   | 0.98                      | 0.98                   |
| Free Movement    | 0.88                   | 0.95                      | 0.88                   |
| Go Behind        | 0.95                   | 0.93                      | 0.91                   |
| Stop             | 1.00                   | 1.00                      | 1.00                   |

4) **F-MEASURE FOR HMM**

F-measure or F-score is the harmonic mean of precision and recall. It provides a generalized view of precision and recall altogether and signifies the number of positive labeled instances that are actually positive. F-measure takes true positive, false positive, and false negative into account but not true negative. Hence, it is sometimes misleading when there is a class imbalance between positive and negative classes in the dataset. As there is no significant imbalance in our dataset primarily, F-measure gives a clear idea on the performance of the HMM. Complex gestures like ‘Return’, ‘Follow’ and ‘Free Movement’ have lower F-score compared to other gestures in the system. The performance of the HMM for the 10 dynamic gestures based on F-measure is given in table 8.

**TABLE 8. F-measure comparison for 10 dynamic gestures using HMM.**

| Gesture          | Young (Below 25 years) | Middle-aged (35-45 years) | Adult (Above 45 years) |
|------------------|------------------------|---------------------------|------------------------|
| Forward          | 0.99                   | 0.97                      | 0.94                   |
| Backward         | 0.94                   | 0.96                      | 0.96                   |
| Speed Up         | 0.84                   | 0.86                      | 0.84                   |
| Speed Down       | 0.86                   | 0.86                      | 0.86                   |
| Return           | 0.85                   | 0.85                      | 0.85                   |
| Follow           | 0.93                   | 0.90                      | 0.93                   |
| U-turn           | 0.94                   | 0.96                      | 0.97                   |
| Free Movement    | 0.89                   | 0.95                      | 0.89                   |
| Go Behind        | 0.95                   | 0.93                      | 0.91                   |
| Stop             | 1.00                   | 1.00                      | 1.00                   |

A column chart comparing F-measures of the 10 recognized dynamic gestures using HMM is shown in figure 19.

5) **SPECIFICITY OR TN RATE FOR HMM**

Specificity considers true negative and false positive but it does not take false negatives into account. It signifies the number of negatively labeled instances that are actually negative. In our experiment, almost all the gestures have high
specificity, which means HMM can correctly identify the negative instances. Among the ten dynamic gestures, only ‘Return’ and ‘Follow’ gesture instances have slightly lower specificity score due to increased false negatives which is negligible. Table 9 shows the specificity or true negative rate of HMM for the 10 dynamic gestures and indicates that our system has a good performance for negative instances as well.

6) FALSE POSITIVE RATE FOR HMM
False positive rate indicates the number of negative instances that are wrongly labeled as positive. In short it can be referred as false alarm ratio. Table 10 presents the false positive rate of the HMM classifier for the 10 dynamic gestures. It shows that most of the gesture classes have very low false positive rate except for ‘Return’. Volunteers from adult category were mostly aged above 55 years and confused between two gestures ‘Speed Up’ and ‘Return’ as their beginning sequences have partial similarity. In order to lower the false positive rate, the gestures must be performed in a rigid way. Adult volunteers had difficulties in performing some continuous longer gestures without giving a pause which led to some of the ‘Return’ gestures to be classified as ‘Speed Up’. This issue can be solved easily with prior gesture training for the users. Moreover, gesture performing ability will keep growing as the users get used to the system. Which will boost the performance of the system and diminish false positives.

7) AVERAGE ACCURACY FOR HMM
Accuracy reflects the overall recognition rate of the classifier. This indicates how well the classifier recognizes the various gesture instances. Table 11 presents the accuracy of the HMM classifier for the 10 dynamic gestures. The table shows the accuracy of each gesture. ‘Stop’ gesture has the highest accuracy of 0.999, followed by ‘Stop’ gesture with an accuracy of 0.999.

Figure 20 presents a column chart of specificity comparison for the 10 dynamic gestures recognized with HMM.

TABLE 9. Specificity comparison for 10 dynamic gestures using HMM.

| Gesture      | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|--------------|------------------------|---------------------------|-----------------------|
| Forward      | 0.999                  | 0.998                     | 0.995                 |
| Backward     | 0.997                  | 0.999                     | 0.997                 |
| Speed Up     | 0.995                  | 0.996                     | 0.992                 |
| Speed Down   | 0.995                  | 0.999                     | 0.994                 |
| Return       | 0.995                  | 0.995                     | 0.975                 |
| Follow       | 0.999                  | 0.998                     | 0.995                 |
| Close        | 0.999                  | 0.999                     | 0.999                 |
| Free Movement| 0.998                  | 0.999                     | 0.990                 |
| Go Behind    | 0.999                  | 0.999                     | 0.998                 |
| Stop         | 1.00                   | 1.00                      | 1.00                  |

Figure 21 shows a false positive column chart for 10 dynamic gestures recognized using HMM.

TABLE 10. FP Rate comparison for 10 dynamic gestures using HMM.

| Gesture      | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|--------------|------------------------|---------------------------|-----------------------|
| Forward      | 0.001                  | 0.001                     | 0.001                 |
| Backward     | 0.003                  | 0.001                     | 0.001                 |
| Speed Up     | 0.012                  | 0.006                     | 0.006                 |
| Speed Down   | 0.007                  | 0.065                     | 0.014                 |
| Return       | 0.012                  | 0.055                     | 0.035                 |
| Follow       | 0.011                  | 0.012                     | 0.013                 |
| Close        | 0.001                  | 0.002                     | 0.001                 |
| Free Movement| 0.011                  | 0.006                     | 0.010                 |
| Go Behind    | 0.001                  | 0.004                     | 0.002                 |
| Stop         | 0.00                   | 0.08                      | 0.30                  |
accuracy of 100%. The rest of the gestures have satisfactory accuracy considering gesture complexity and duration. Our system achieves higher accuracy than existing systems for complex gesture recognition. The Middle-aged gesture dataset achieves the highest accuracy value for the HMM, which is 95.67%. To the best of our knowledge, the overall accuracy of the HMM combining all three age categories is 94.61%, which is the highest for complex dynamic gesture recognition. It is necessary to note that different people perform the same gesture for different durations. This is why our dataset consists of three different age categories, where people of different ages performed all the ten gestures. It was observed that these age groups took different times to complete a gesture. Consequently, our system obtained the same gesture instances made up of various lengths and patterns. Our proposed system successfully solves these differences and offers good accuracy for all age categories. It clearly signifies the robustness of our system.

TABLE 11. Average accuracy comparison for 10 dynamic gestures using HMM.

|       | Young (below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|-------|------------------------|---------------------------|------------------------|
| Forward | 96.87                  | 94.67                     | 92.30                  |
| Backward | 95.00                  | 97.87                     | 93.35                  |
| Speed Up | 96.87                  | 95.00                     | 88.33                  |
| Speed Down | 94.77                  | 96.87                     | 94.57                  |
| Return  | 90.85                  | 98.97                     | 96.97                  |
| Follow  | 95.00                  | 90.83                     | 91.27                  |
| PullIt  | 97.50                  | 97.87                     | 94.57                  |
| Free Movement | 98.53                | 90.83                     | 87.50                  |
| Go Behind | 95.83                  | 93.83                     | 93.55                  |

The accuracy comparison column chart of HMM for the 10 dynamic gestures is shown in Figure 22. The comparison also highlights the differences in accuracy for different age categories.

FIGURE 22. Accuracy comparison chart for dynamic gesture recognition using HMM.

8) CONFUSION MATRIX FOR SVM
The feature vectors generated from the performed gesture instances must be equal in size in order to make the SVM work efficiently. As all kernels of SVM are based on distance, in order to differentiate between two gestures, it is required that those two gesture sequences are of the same size. Otherwise, the longer gesture sequence will dominate in the computation of kernel matrix. Some of the gestures in our system take longer to perform and produce longer feature vectors than other gestures. So it is necessary to make the length of these gesture sequences equal. This can be achieved by padding the shorter gesture sequences. However, adding additional data based on the existing sequence to the gestures will result in poor performance, as some of the gestures have similar patterns, and as a result, their padding data will be the same. The number of false positives greatly increases. In order to solve the issue, we divided our gesture domain into two sub-domain: simple gestures (‘Forward’, ‘Backward’, ‘Speed Up’, ‘Speed Down’ and ‘Stop’) and complex gestures (‘Return’, ‘Follow’, ‘Unfollow’, ‘Free Movement’ and ‘Go Behind’). We trained the SVM solely using these two different datasets.

9) CONFUSION MATRIX FOR SIMPLE GESTURES
Confusion matrices for five simple gesture recognition with multi-class SVM are shown in table 12 for young category, table 13 for Middle-aged category and table 14 for adult category. In the confusion matrix, the actual gesture class is represented by each row, while the anticipated class label is represented by each column.

TABLE 12. Confusion matrix for SVM (young; below 25 years).

|       | Forward | Backward | Speed Up | Speed Down | Stop |
|-------|---------|----------|----------|------------|------|
| Forward | 112     | 0        | 0        | 0          | 0    |
| Backward | 0       | 115      | 5        | 0          | 0    |
| Speed Up | 0       | 11       | 109      | 0          | 1    |
| Speed Down | 7       | 0        | 0        | 113        | 0    |
| Stop   | 0       | 0        | 4        | 0          | 116  |

The confusion matrix shown in table 12 illustrates the similarity between ‘Forward’ and ‘Speed Down’ gesture. Out of 120 actual ‘Forward’ gesture instances, 112 gestures have been labeled as ‘Forward’ indicating 112 true positives and rest of the 8 ‘Forward’ gestures labeled as ‘Speed Down’ indicates 8 false negatives. On the other hand, 7 ‘Speed Down’ gestures are wrongly recognized as ‘Forward’ gesture, which leads to 7 false positives for ‘Forward’.

TABLE 13. Confusion matrix for SVM (middle-aged; 25-45 years).

|       | Forward | Backward | Speed Up | Speed Down | Stop |
|-------|---------|----------|----------|------------|------|
| Forward | 114     | 0        | 0        | 0          | 0    |
| Backward | 0       | 112      | 7        | 1          | 0    |
| Speed Up | 0       | 11       | 107      | 0          | 2    |
| Speed Down | 8       | 0        | 0        | 112        | 0    |
| Stop   | 0       | 0        | 1        | 0          | 119  |

Table 13 represents the confusion matrix for 5 simple gestures recognized by multi-class SVM. The values are almost similar to table 12 except ‘Stop’ gesture class. 119 ‘Stop’ gesture instances out of 120 are recognized for Middle-aged dataset.

Unlike HMM, all the instances of ‘Stop’ gesture could not be recognized using SVM. The length of gesture sequence plays a vital role here. ‘Stop’ gesture has the lowest motion
change compared to other gestures, it produces the shortest gesture sequence. Additional padding was added to the original sequence based on the last part of the original sequence such that its range becomes equal to the rest of the four simple gestures. As a result, misclassification occurred for some of the ‘Stop’ gesture sequences.

10) AVERAGE ACCURACY FOR SIMPLE GESTURES USING SVM

Table 15 represents the average accuracy for 5 simple gesture recognition using multi-class SVM. The ‘Forward’ gesture achieved 93.33% accuracy for young category while HMM achieved 96.67% accuracy for the same gesture and dataset. Accuracy for rest of the four gestures are satisfactory except ‘Speed Up’. HMM achieved far better accuracy than SVM for ‘Speed Up’ gesture. It happened due to padding.

TABLE 15. Average accuracy comparison for 5 simple dynamic gestures using SVM.

| Gesture         | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|-----------------|------------------------|---------------------------|------------------------|
| Forward         | 93.33                  | 95.00                     | 90.83                  |
| Backward        | 95.83                  | 93.33                     | 96.67                  |
| Speed Up        | 90.00                  | 93.77                     | 93.77                  |
| Speed Down      | 94.17                  | 95.33                     | 97.33                  |
| Stop            | 96.87                  | 99.17                     | 99.83                  |

Figure 23 represents a column chart for five simple gesture recognition with multi-class SVM.

11) CONFUSION MATRIX FOR COMPLEX GESTURES

Confusion matrices for five complex gesture recognition with multi-class SVM classes are included in table 16 for young category, table 17 for Middle-aged category and table 18 for adult category. In the confusion matrix, the actual gesture class is represented by each row, while the anticipated class label is represented by each column.

TABLE 16. Confusion matrix for SVM (young; below 25 years).

| Gesture         | Return | Follow | Unfollow | Free Movement | Go Behind |
|-----------------|--------|--------|----------|---------------|-----------|
| Return          | 109    | 0      | 0        | 11            | 0         |
| Follow          | 4      | 109    | 2        | 3             | 0         |
| Unfollow        | 0      | 11     | 109      | 0             | 2         |
| Free Movement   | 8      | 0      | 0        | 112           | 0         |
| Go Behind       | 0      | 0      | 5        | 0             | 115       |

In table 16, it can be observed intuitively that the number of true positives for complex gestures has fell off compared to simple gestures. Performances of complex gestures do not solely depend on classifiers, it also depends on the participants. The more various hand motions a gesture has, the more differences are created while performing it. Which leads to a few more misclassification.

TABLE 17. Confusion matrix for SVM (middle-aged; 25-45 years).

| Gesture         | Return | Follow | Unfollow | Free Movement | Go Behind |
|-----------------|--------|--------|----------|---------------|-----------|
| Return          | 11     | 2      | 1        | 1             | 0         |
| Follow          | 3      | 109    | 2        | 3             | 0         |
| Unfollow        | 3      | 3      | 109      | 3             | 0         |
| Free Movement   | 2      | 3      | 109      | 2             | 0         |
| Go Behind       | 0      | 0      | 3        | 3             | 108       |

Table 17 represents the confusion matrix for 5 complex gesture recognition with SVM for Middle-aged dataset. There is no significant difference in true positives compared to the previous table 16. It gives a clear idea that gesture differences created by young and Middle-aged users can be invariantly addressed by SVM.

TABLE 18. Confusion matrix for SVM (adult; above 45 years).

| Gesture         | Return | Follow | Unfollow | Free Movement | Go Behind |
|-----------------|--------|--------|----------|---------------|-----------|
| Return          | 11     | 1      | 1        | 1             | 0         |
| Follow          | 2      | 109    | 2        | 3             | 0         |
| Unfollow        | 3      | 3      | 109      | 3             | 0         |
| Free Movement   | 2      | 3      | 109      | 2             | 0         |
| Go Behind       | 0      | 0      | 3        | 3             | 108       |

Table 18 represents the confusion matrix for complex gesture recognition with SVM for adult dataset. It shows that ‘Free Movement’ gesture has slightly low true positive compared to other age categories. It happens because of the major differences created in gesture patterns due to various hand motions and long duration. SVM achieved considerable performance for rest of the gestures.

12) AVERAGE ACCURACY FOR COMPLEX GESTURES USING SVM

Table 19 represents the average accuracy for 5 complex gesture recognition using multi-class SVM. The ‘Unfollow’ gesture achieved 90.83% accuracy for young category while HMM achieved 97.67% accuracy for the same gesture and dataset. Accuracy for rest of the four gestures are lower than HMM. Figure 24 represents a column chart for five complex gesture recognition with multi-class SVM.

TABLE 19. Average accuracy comparison for 5 complex dynamic gestures using SVM.

| Gesture         | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|-----------------|------------------------|---------------------------|------------------------|
| Unfollow        | 90.83                  | 97.67                     | 97.67                  |
| Free Movement   | 90.83                  | 97.67                     | 97.67                  |
| Go Behind       | 90.83                  | 97.67                     | 97.67                  |

Figure 24 represents a column chart for five complex gesture recognition with multi-class SVM.
13) OVERALL ACCURACY COMPARISON FOR SVM

Table 20 shows the overall accuracy of 5 simple gesture and 5 complex gesture recognition with multi-class SVM and categorizing the dataset into young, Middle-aged and adult. Simple gestures have greater accuracy compared to complex gestures as anticipated. Average accuracy is 93.33% for simple gesture recognition and 90.56% for complex gesture recognition. Again, the complex gesture recognition accuracy is higher than existing complex gesture recognition system to the best of our knowledge.

Table 20. Overall accuracy comparison for 5 simple dynamic gestures and 5 complex dynamic gestures using SVM.

|                | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|----------------|------------------------|--------------------------|-----------------------|
| Simple Gesture | 96.00                  | 96.00                    | 92.00                 |
| Complex Gesture| 90.11                  | 91.11                    | 90.33                 |

14) CONFUSION MATRIX FOR CNN

Table 21, 22 and 23 represent the confusion matrices with ten gesture classes for three age categories. In the confusion matrix, each of the ten dynamic gestures is assigned a class label. Each row corresponds to the actual class label, whereas each column corresponds to the anticipated class label.

Table 21. Confusion matrix for CNN (young; below 25 years).

| Gesture       | Number of instances | Predicted Gesture | True Positive | False Positive | True Negative | False Negative | Correct Classification % |
|---------------|---------------------|-------------------|---------------|----------------|---------------|---------------------|-------------------------|
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |
| Stop          | 120                 | Stop              | 120           | 0              | 119           | 1                   | 99.2%                   |

Table 22 shows the confusion matrix for the ten predefined dynamic gesture recognition with CNN for Middle-aged age category. The number of true positives are almost the same as that in table 21. ‘Stop’ gesture achieved 120 true positive likewise HMM. 13 ‘Follow’ gesture instances are wrongly labeled as ‘Free Movement’, 11 instances as ‘Go Behind’ and 5 instances as ‘Return’ indicating 29 false positives. These gestures are complex, and exhibit diverse patterns. If more data can be fed into the system, the number of false positives will diminish.

Table 23 constitutes the confusion matrix for the ten predefined dynamic gesture recognition with CNN for Middle-aged age category. Most of the gestures have similar true positives as before, which did not occur for the HMM and SVM. Gesture recognition with HMM and SVM for the adult age category performed comparatively lower than for...
the young and middle-aged categories. But for CNN, there is a small interval of gesture recognition performance for adult vs young and Middle-aged category. This clearly indicates that CNN can minimize the difference in gesture patterns made by various age classes better than HMM and SVM.

15) PRECISION FOR CNN
Table 24 shows the comparison of precision of the CNN classifier for the ten gestures performed by the volunteers from three age categories. Among the complex gestures ‘Follow’ and ‘Free Movement’ have less precision due to increased false positive. More training instances for the two gestures will improve the performance of the CNN. Rest of the gestures have pretty good precision score but slightly lower than HMM except for ‘Speed Down’ and ‘Return’ for adult category. The performance increased significantly for these two gestures compared that with of the HMM.

16) SENSITIVITY OR RECALL OR TP RATE FOR CNN
Table 25 provides sensitivity or recall or true positive rate of the CNN classifier. The ten dynamic gestures of the proposed system are associated with their recall values in the table. The table shows the number of classes labeled as positive, which are actually positive. It takes true positives and false negatives into account but ignores false positives. The number of true positives is slightly lower for CNN than for HMM because of the higher number of false negatives for some gesture instances.

17) F-MEASURE FOR CNN
Performance of the classifier for the 10 dynamic gestures based on F-measure is given in table 26. The table gives an overall idea of precision and recall altogether. F-score for ‘Follow’ and ‘Free Movement’ gestures are slightly low because of their low precision and recall score.

18) SPECIFICITY OR TRUE NEGATIVE RATE
Table 27 shows the specificity or true negative rate of CNN classifier for the 10 dynamic gestures calculated from the confusion matrix. The table indicates the ability of CNN to label classes negatively which are indeed negative. ‘Speed
TABLE 26. F-measure comparison for 10 dynamic gestures using CNN.

| Gesture      | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|--------------|------------------------|---------------------------|------------------------|
| Forward      | 0.85                   | 0.82                      | 0.81                   |
| Backward     | 0.90                   | 0.91                      | 0.80                   |
| Speed Up     | 0.91                   | 0.91                      | 0.81                   |
| Speed Down   | 0.88                   | 0.88                      | 0.81                   |
| Return       | 0.97                   | 0.93                      | 0.89                   |
| Follow       | 0.79                   | 0.79                      | 0.82                   |
| Get Down     | 0.94                   | 0.94                      | 0.80                   |
| Free Movement| 0.86                   | 0.78                      | 0.77                   |
| Stop         | 1.00                   | 1.00                      | 1.00                   |

Figure 28. F-measure comparison chart for dynamic gesture recognition using CNN.

‘Up’ has higher specificity score with CNN for young category. ‘Return’ has higher specificity score with CNN than HMM for both Middle-aged and adult category. It signifies strong correlation of complex gesture performance invariant of age class with CNN.

TABLE 27. Specificity comparison for 10 dynamic gestures using CNN.

| Gesture      | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|--------------|------------------------|---------------------------|------------------------|
| Forward      | 0.93                   | 0.91                      | 0.85                   |
| Backward     | 0.91                   | 0.91                      | 0.91                   |
| Speed Up     | 0.90                   | 0.90                      | 0.90                   |
| Speed Down   | 0.95                   | 0.95                      | 0.95                   |
| Return       | 0.97                   | 0.94                      | 0.91                   |
| Follow       | 0.97                   | 0.97                      | 0.92                   |
| Get Down     | 0.95                   | 0.95                      | 0.95                   |
| Free Movement| 0.96                   | 0.96                      | 0.96                   |
| Stop         | 1.00                   | 1.00                      | 1.00                   |

Figure 29 presents a column chart regarding specificity of 10 recognized gestures with CNN.

19) FALSE POSITIVE RATE FOR CNN

Table 28 presents the false positive rate of the CNN classifier for the 10 dynamic gestures. CNN has a slightly higher false-positive rate than HMM. ‘Free Movement’ gesture has the highest FP rate for CNN. The FP rate for ‘Go Behind’ gesture recognition with CNN has the maximum difference among the age classes. FP rate of ‘Go Behind’ gesture for young category is 0.002, whereas for adult category it is 0.018. On the other hand ‘Speed down’ gesture for the adult category has a lower FP rate compared to other age categories.

FIGURE 29. Specificity comparison chart for dynamic gesture recognition using CNN.

Insufficient dataset and noisy optimizer played significant role in here. The original gradient descent can be used instead of stochastic when a larger dataset is generated with a higher convergence rate.

TABLE 28. FP rate comparison for 10 dynamic gestures recognized with CNN.

| Gesture      | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|--------------|------------------------|---------------------------|------------------------|
| Forward      | 0.011                  | 0.011                     | 0.012                  |
| Backward     | 0.009                  | 0.009                     | 0.009                  |
| Speed Up     | 0.010                  | 0.010                     | 0.010                  |
| Speed Down   | 0.011                  | 0.015                     | 0.005                  |
| Return       | 0.013                  | 0.011                     | 0.019                  |
| Follow       | 0.002                  | 0.002                     | 0.018                  |
| Get Down     | 0.006                  | 0.006                     | 0.005                  |
| Free Movement| 0.031                  | 0.032                     | 0.034                  |
| Stop         | 0                      | 0                         | 0                      |

Figure 30 shows a column chart of false positive rate comparison among 10 dynamic gestures recognized using CNN.

FIGURE 30. FP Rate comparison chart for dynamic gesture recognition using CNN.

20) AVERAGE ACCURACY FOR CNN

Table 29 represents the accuracy of the CNN classifier for the 10 dynamic gestures. ‘Forward’ gesture has similarity
in pattern with ‘Speed Down’ gesture and ‘Speed Up’ has similarity with ‘Backward’. As a result, ‘Forward’ achieves a maximum accuracy of 95.83% in the young category, whereas a similar ‘Speed Down’ achieves 87.50%. Similar relation found between ‘Speed Up’ and ‘Backward’ as well with an accuracy of 90.00% and 88.33% respectively. ‘Go Behind’ gesture is much similar to ‘Return’ if not carefully performed. Moreover, they have the same starting patterns in their sequences. ‘Go Behind’ achieves the lowest accuracy of 82.50% and 77.50% for young and adult dataset. The accuracy of the CNN is subject to change over time in our system, as more training data will be available to the system. The system is developed with scalability so that continuous addition to the original dataset does not create any performance issues; rather, it enhances the performance of the system.

### TABLE 29. Average accuracy comparison for 10 dynamic gestures recognized by CNN.

| Gesture    | Young (Below 25 years) | Middle-aged (25-45 years) | Adult (Above 45 years) |
|------------|------------------------|---------------------------|------------------------|
| Forward    | 95.17                  | 92.90                     | 94.17                  |
| Reckoned   | 90.50                  | 90.90                     | 88.33                  |
| Speed Down | 97.50                  | 97.50                     | 87.30                  |
| Return     | 94.17                  | 94.17                     | 91.67                  |
| Follow     | 85.83                  | 81.47                     | 85.43                  |
| Call Down  | 80.15                  | 79.25                     | 85.00                  |
| Free Movement | 85.83             | 81.47                     | 81.47                  |
| Go Behind  | 92.50                  | 83.33                     | 77.50                  |
| Stop       | 100                    | 100                       | 100                    |

The accuracy column chart of CNN for the 10 dynamic gestures is shown in Figure 31.

![FIGURE 31. Average accuracy comparison chart for dynamic gesture recognition using CNN.](image)

21) OVERALL ACCURACY COMPARISON OF THE SYSTEM

Table 30 shows the overall accuracy comparison among HMM, SVM and CNN for 10 dynamic gesture recognition in three different age categories. HMM achieves the highest accuracy of 95.67% in Middle-aged category, SVM achieves the highest accuracy of 92.59% in Middle-aged category and CNN achieves the highest accuracy of 89.58% in young category. Taking the average accuracy of the three age categories altogether, HMM, SVM and CNN achieves an overall accuracy of 94.61%, 91.95% and 88.97% respectively.

### TABLE 30. Overall accuracy comparison of the system.

| Category       | HMM  | SVM  | CNN  |
|----------------|------|------|------|
| Young (Below 25 years) | 95.00 | 92.09 | 89.58 |
| Middle-aged (25-45 years) | 95.67 | 92.59 | 89.17 |
| Adult (Above 45 years)     | 93.17 | 91.17 | 88.17 |
| Average Accuracy          | 94.61 | 91.95 | 88.97 |

The gestures performed by the volunteers had no prior training on our predefined gestures. As a result, the same gesture was performed differently by volunteers from different age categories. As a result, each gesture exhibited diverse patterns in its sequence. Even though this has affected the accuracy of gesture recognition, it has made the system more robust for new users. The parameters of the HMM are set through a trial-and-error process to achieve optimal performance in our system. The multi-class SVM with Gaussian kernel performs slightly lower than the HMM. Moreover, it is less suitable for systems such as ours, which involve variable-length complex dynamic gestures. The overall accuracy of the CNN is not optimal in our system. But training time for CNN is very low as Stochastic Gradient Descent results in faster iterations. The proposed system does not depend on static datasets. The more users use the system, the more data it receives. Moreover, new gesture classes can be introduced as well. As a result, the dataset will continue to grow. At some point CNN will have adequate data to train with. In this case, using the original gradient descent instead of stochastic will increase the convergence rate with slower iterations, which is an affordable trade-off. Figure 32 shows the column chart of overall accuracy comparison among HMM, SVM and CNN based on three age categories.

![FIGURE 32. Verall accuracy comparison of the system.](image)

C. PERFORMANCE COMPARISON WITH EXISTING SYSTEMS

Table 31 shows performance comparison of other existing systems with our proposed system in terms of 3D dynamic
gesture recognition accuracy. Li et al. [21] used Kinect Skeletal Tracking data in a 3D coordinate space, which is similar to ours. They applied HMM for recognizing 10 dynamic gestures and achieved an overall accuracy of 91.60%. Their gesture domain is simpler than our predefined gesture domain. Our complex gestures are capable of producing longer gesture sequences as well as variable-length sequences by users from different age categories. Despite having a complex gesture domain, our proposed system achieves a maximum accuracy of 95.67% for middle-aged gesture category and 94.61% on average with HMM. This proves the superiority of our system in Kinect Skeletal Tracking based gesture recognition domain.

Our system uses simple yet effective data pre-processing on the fly without much calculation overhead. Considering a reference point and starting gesture sequences from that point plays a significant role in generating similar sequences for the same gesture class. Moreover, normalization makes the sequences under the same class much similar. These simple factors make our system perform better than others. Moreover, we have optimized the parameters of the HMM based on trial and error process that is best suited for any Kinect Skeletal Tracking dataset. This demonstrates the robustness of the system. The SVM configuration with an appropriate kernel trick in our proposed system also proves to be robust in both simple and complex gesture recognition. CNN in our system will achieve ample performance with the increased data generated by our adaptation module.

Most of the existing gesture recognition systems work with simple motion gestures. Moreover, these systems are limited to a fixed number of gestures, and do not provide any adaptation module. Some of these systems suffer from large calculation overheads owing to the nature of their dataset. To the best of our knowledge, there is no Kinect skeletal tracking-based dataset comprising complex and long gestures such as our gesture domain. The closest method mentioned earlier in [21] performs with less accuracy than ours. The gesture adaptation module is a fundamental requirement in human-robot interaction. Our system provides an effective adaptation system with decent accuracy, which makes it dynamic and adaptive to many new gestures. The system does not depend on any pretrained dataset. Users can use our system from the beginning with their completely new gesture domain and train the system themselves using the GUI. Considering all these factors, our proposed system provides a robust and efficient dynamic gesture recognition and adaptation system with ample performance in Kinect Skeletal Tracking domain.

### D. GESTURE ADAPTATION MODULE

1) **SELF ADAPTATION**

In figure 33, a user is trying to perform the ‘Free Movement’ gesture. However, he did something different from the original. The system does not adapt to a new gesture during the first go. The system will do nothing at the first time and show ‘Unrecognized Gesture’ dialog only. Meanwhile, it tracks the number. When a user performs a similar gesture three times, the system gives a prompt asking whether the user wants to add the new gesture. New or incorrect gestures need not be performed consecutively three times. Even if the user performs another gesture between two incorrect gestures, the system still manages to keep track of the incorrect gestures, and as soon as the count reaches 3, it will show the prompt.

**Figure 33. A user performing modified gesture of ‘Free Movement’.**

Figure 34 demonstrates what happens when the user presses ‘Yes’ to start the adaptation module. First, our system creates new feature vectors using the three newly performed unrecognized gestures. The system trains the HMM using these new feature vectors and compares the log-likelihood values of existing gesture classes. If the likelihood value matches more than 30%, the new gesture is labeled as the corresponding gesture class. In figure 34, the new instance of ‘Free Movement’ passes the threshold and is automatically added to the ‘Free Movement’ gesture class with a lower
weight value. All these occur in the background of the system, and the user only sees the final message on the GUI.

Figure 34. Self adaptation of ‘Free Movement’ gesture by the system.

2) CONSENT BASED ADAPTATION

Figure 35 shows a demonstration of consent based adaptation of our system. When three unrecognized gesture instances match but they do not pass the threshold for self adaptation, the system asks the user to add the new gesture class manually. In the figure, user is adding a new gesture class named ‘Pick Up’. The new gesture class is instantly added to the drop-down list, and is available for further training with new data. ‘Pick Up’ gesture has become the 11th gesture class of the system.

Figure 35. Consent based gesture adaptation by the system.

E. ROBOT NAVIGATION MODULE

This module navigates the robot based on the estimated pointing direction and the previously recognized dynamic gesture as command. This module estimates the path of robot from source to destination based on the algorithms described in earlier section. The process is shown in figure 36.

We asked 3 volunteers to stand in front of the Kinect and navigate the robot using pointing gestures in order to analyze our robot navigation system. The robot started 200 cm behind the Kinect at (0, −200) on the X-Z plane. The volunteers were asked to navigate the robot to a specific location in two different tasks using both Algorithm 1 and Algorithm 2, and the time required to navigate the robot was recorded. The robot speed was kept the same in both scenarios.

1) SCENARIO 1

Three volunteers were instructed to guide the robot to a specified position on the border, as shown by the cross (X) mark in figure 37.

We instructed them to complete the task three times using both navigation techniques, with an average time computed for each person, as shown in table 32.

Table 32. Average time consumed for robot navigation in scenario 1.

|           | Method 1 (sec) | Method 2 (sec) |
|-----------|---------------|---------------|
| Person 1  | 8.92          | 12.06         |
| Person 2  | 8.74          | 12.02         |
| Person 3  | 8.89          | 11.95         |

2) SCENARIO 2

Like scenario 1, we instructed three people to move the robot to a certain position inside the boundary as shown with a cross (X) mark in figure 38.

We instructed them to perform this thrice using both navigation methods. The average time was calculated for every person as shown in table 33.

These results can be visualized using bar diagrams shown in figure 39.
F. LIMITATIONS
The limitations of the proposed system are as follows:

- In order to track the full upper body, the user needs to stand in front of the Kinect maintaining a minimum distance from it. The user is advised to stand while using the system to ensure tracking efficiency. Efficacy might be hampered if input notions are not properly maintained.
- As we are only using the right hand for performing pointing gestures, navigating the robot to the left of the user may become difficult while the user is facing the Kinect.
- Users cannot point directly behind them since the Kinect is unable to track the hand if it is kept behind the body.
- As we have not tracked fingers in this system, a pointing gesture is detected as long as the user keeps his right arm fully stretched and parallel to the ground, regardless of the current state of the hand. In general, pointing gestures include a certain pose of the hand that involves keeping the index finger extended and every other finger closed.
- The performance of this system at outside will degrade as Kinect uses IR sensors and it is affected by direct sunlight.

- Due to the unavailability of a physical mobile robot, we could not calculate the accuracy of the robot navigation system in a real life scenario.
- More data is required to get an optimal performance from Convolutional Neural Network.

VI. CONCLUSION AND FUTURE WORKS
The fundamental goal of this research is to create a system that uses pointing and dynamic commanding gestures to navigate mobile robots under any lighting condition and in any environment. Kinect has been chosen as the sensor to capture gesture instances because it is capable of tracking human bodies and extracting coordinates of different body joints in a 3D coordinate space efficiently. Moreover, this study proposes a robust method for dynamic hand gesture recognition. HMM, multiclass SVM, and CNN-based classifiers have been used to classify dynamic gestures. To analyze the performance of the system, a comprehensive experiment has been carried out, which includes a five-fold cross validation utilizing the training dataset, which comprises 3,600 gesture instances of ten predetermined dynamic gestures. The system is not limited to this ten gestures rather, a user can train the system with a completely new gesture class through the user interface on the fly. The system is tested extensively to correctly calculate the 3D dynamic gesture recognition accuracy of the HMM, SVM, and CNN classifiers. HMM achieves an accuracy of 95% for Young category, 95.67% for Middle-aged category, 93.17% for Adult category yielding an average accuracy of 94.61%. The Multi-Class SVM achieves an accuracy of 92.09% for the Young category, 92.59% for the Middle-aged category, and 91.17% for the Adult category, yielding an average accuracy of 91.95%. The CNN achieves an accuracy of 89.58% for the Young category, 89.17% for the Middle-aged category, and 88.17% for the Adult category, with an average accuracy of 88.97%. Another major part of this work includes the development of a complete gesture adaptation module combining supervised and semi-supervised learning, so that the system can adapt to new and unrecognized gestures. In this study, two different algorithms are also developed for navigating robots using the recognized pointing and dynamic gestures. The first approach considers that the Kinect is confined within a boundary and the robot is only allowed to move inside that boundary, whereas the second approach is based on adjusting the robot’s current position before navigating it towards the pointing direction.

A. FUTURE WORKS
To alleviate the limitations of the current system, this project offers scope for future research. In the future, we intend to improve the accuracy of our proposed system. Future work on this system may include using both hands for pointing gestures. In addition, finger tracking can be added so that a correct pointing pose can be correctly identified. An obstacle detection system can be implemented with this system such that the robot can avoid any obstacle in its path. More data can be collected to obtain the optimal performance of the...
classifiers, so that scalability can be evaluated. Unsupervised learning can be implemented in gesture adaptation module. The system can be trained with any sign-language dataset to be used as a dynamic sign-language translator.

REFERENCES

[1] 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,” in Language and Linguistics Compass, vol. 2, no. 4. U.K.: Wiley, 2008, Art. no. 569588.

[2] TechTarget Contributor. (Aug. 2016). Data Glove, TechTarget. Accessed: Sep. 12, 2022. [Online]. Available: http://www.techtarget.com/what/definition/data-glove

[3] H. Ishiyama and S. Kurabayashi, “Monochrome glove: A robust real-time hand gesture recognition method by using a fabric glove with design of structured markers,” in Proc. IEEE Virtual Reality (VR), Mar. 2016, pp. 187–188, doi: 10.1109/VR.2016.7504716.

[4] Ashley Timms. (Nov. 8, 2019). 9 Gesture Control Devices You Need to See to Believe, Gadget Flow. Accessed: Sep. 12, 2022. [Online]. Available: https://thegadgetflow.com/blog/gesture-control-devices/

[5] D. Xu, Y.-L. Chen, C. Lin, X. Kong, and X. Wu, “Real-time dynamic gesture recognition system based on depth perception for robot navigation,” in Proc. IEEE Int. Conf. Robot. Biomimetics (ROBO), Dec. 2012, pp. 689–694, doi: 10.1109/ROBO.2012.6491047.

[6] M. Van den Bergh, D. Carton, R. De Nijis, N. Mitsou, C. Landsiedel, K. Kuehnelt, D. Wöllhert, L. Van Gool, and M. Buss, “Real-time 3D hand gesture recognition with a glove and vision for understanding directions from humans,” in Proc. RO-MAN, Jul. 2011, pp. 357–362, doi: 10.1109/ROMAN.2011.6005195.

[7] Shawn Hickey. (Aug. 31, 2022). Kinect for Windows, Microsoft. Accessed: Sep. 12, 2022. [Online]. Available: https://docs.microsoft.com/en-us/windows/apps/design/devices/Kinect-for-windows

[8] A. Rahman, J. A. Mahmud, and M. Hasanuzzaman, “Pointing and commanding gesture recognition in 3D for human-robot interaction,” in Proc. Int. Conf. Innov. Eng. Technol. (ICIET), Dec. 2018, pp. 1–10, doi: 10.1109/ICIET.2018.8660913.

[9] S. Rodriguez, A. Picon, and A. Villodas, “Robust vision-based hand tracking using single camera for ubiquitous 3D gesture interaction,” in Proc. IEEE Symp. 3D User Interfaces (3DUI), Mar. 2010, pp. 135–136.

[10] G. R. S. Murthy and R. S. Jadon, “Hand gesture recognition using neural networks,” in Proc. IEEE 2nd Int. Advance Comput. Conf. (IACC), Feb. 2010, pp. 134–138, doi: 10.1109/100.5423024.

[11] Q. Ke, J. Liu, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, “Computer vision for human-machine interaction,” in Computer Vision for Assistive Healthcare, L. Marco, and G. M. Farinella, Eds. New York, NY, USA: Academic, 2018, pp. 127–145.

[12] C. Zhu, J. Yang, Z. Shao, and C. Liu, “Vision based hand gesture recognition using 3D shape context,” IEEE/CCA I. Automat. Sinica, vol. 8, no. 9, pp. 1600–1613, Sep. 2021, doi: 10.1049/IJAS.2019.1911534.

[13] D. Liu, L. Zhang, and Y. Wu, “LD-ContGR: a large RGB-D video dataset for long-distance continuous gesture recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 3294–3302, doi: 10.1109/CVPR52688.2022.00330.

[14] H. Mansoor, N. Kalra, P. Goyal, M. Bansal, and N. Wadhwa, “Hand gesture recognition using 3D CNN and computer interfacing,” in Inventive Systems and Control, vol. 436, V. Suma, Z. Baig, S. K. Shannugam, and P. Lorenz, Eds. Singapore: Springer, 2022.

[15] H. Chen, Y. Li, H. Fang, W. Xing, Z. Lu, and Q. Miao, “Multi-scale attention windows,” in Proc. Int. Joint Conf. Neural Net. (IJCNN), Jul. 2014, pp. 3930–3937.

[16] C. Huang, A. F. Molisch, R. Wang, P. Tang, R. He, and Z. Zhong, “Research on kernel functions of SVM for line-of-sight identification in vehicle-to-vehicle MIMO system,” in Proc. IEEE Symp. Antennas Propag. USNC/URSI Radio Sci. Meeting, Jul. 2019, pp. 2107–2108.

[17] S. A. Jan, Y. D. Lee, J. Shin, and I. Koo, “Segment fault classification based on support vector machine and statistical time-domain features,” IEEE Access, vol. 5, pp. 6882–6890, 2017.

[18] M. M. Kabir, A. Q. Ohi, and M. F. Mridha, “A multi-plant disease diagnosis method using convolutional neural network,” in Computer Vision and Machine Learning in Agriculture. Singapore: Springer, 2021, pp. 99–111.

[19] K. M. Habib, M. A. Habib, N. A. Towhid, and M. I. H. Showrov, “A novel deep learning-based sentiment analysis of Twitter data for U.S. airline service,” in Proc. IEEE Int. Conf. Inf. Commun. Technol. Sustain. Develop. (ICICT SD), Feb. 2021, pp. 450–455, doi: 10.1109/ICICT4SD50815.2021.9396879.

[20] K. M. Habib, M. S. Iqbal, F. M. Shah, J. Al Mahmud, M. H. Popel, M. I. H. Showrov, S. Ahmed, and O. Rahman, “A survey of methods for managing the classification and solution of data imbalance problem,” J. Comput. Sci., vol. 16, no. 11, pp. 1546–1557, Nov. 2020.
[40] K. M. Hasib, M. I. H. Showrov, M. J. Al, and K. Mithu. “Imbalanced data classification using hybrid under-sampling with cost-sensitive learning method,” in Edge Analytics, vol. 869, R. Patgiri, S. Bandypadhyay, M. D. Borah, and V. E. Balas, Eds. Singapore: Springer, 2022, pp. 423–435.

[41] M. F. Mridha, S. C. Das, M. M. Kabir, A. A. Lima, M. R. Islam, and Y. Watanobe. “Brain-computer interface: Advancement and challenges,” Sensors, vol. 21, no. 17, p. 5746, Aug. 2021, doi: 10.3390/s21175746.

JUBAYER AL MAHMUD received the B.Sc. and M.Sc. degrees in computer science and engineering from the University of Dhaka. He has been active in teaching and research in the fields of computer science and engineering for five years. He has worked at several universities as a Lecturer and later as an Assistant Professor in computer science and engineering. He is currently working on several research projects, such as COVID-19 vaccine efficacy and aftereffects in Bangladesh, automated gesture adaptation for human–robot interaction, and heart-block identification before angiogram. His research interests include computer science, especially machine learning, computer vision, data mining, natural language processing, human–robot interaction, and human–computer interaction.

BANDHAN CHANDRA DAS received the bachelor’s and master’s degrees in computer science and engineering (CSE) from Jahangirnagar University, Bangladesh. His research works have been published in widely recognized international conferences and journals. His research interests include data mining, web mining, analysis, machine learning, natural language processing, and data science. He received the Developing Country Researcher Award at 7th IEEE Asia–Pacific Conference on Computer Science and Data Engineering (CSDE-2020), Gold Coast, Australia. Moreover, he serves as a reviewer in international conferences and journals of his research areas.

JUNGIL SHIN (Senior Member, IEEE) received the B.Sc. degree in computer science and statistics and the M.Sc. degree in computer science from Pusan National University, South Korea, in 1990 and 1994, respectively, and the Ph.D. degree in computer science and communication engineering from Kyushu University, Japan, in 1999, under a Scholarship from the Japanese Government (MEXT). He was an Associate Professor, a Senior Associate Professor, and a Full Professor with the School of Computer Science and Engineering, The University of Aizu, Japan, in 1999, 2004, and 2019, respectively. He has coauthored more than 270 published papers for widely cited journals and conferences. His research interests include pattern recognition, image processing, computer vision, machine learning, human–computer interaction, non-touch interfaces, human gesture recognition, automatic control, Parkinson’s disease diagnosis, ADHD diagnosis, user authentication, machine intelligence, as well as handwriting analysis, recognition, and synthesis. He is a member of ACM, IEICE, IPSJ, KISS, and KIPS. He served as the program chair and as a program committee member for numerous international conferences. He serves as an Editor for IEEE journals and Sensors (MDPI). He serves as a reviewer for several major IEEE and SCI journals.

Khan Md. Hasib (Member, IEEE) received the B.Sc. degree from the Computer Science and Engineering Department, Ahsanullah University of Science and Technology (AUST), and the M.Sc. degree from the Computer Science and Engineering Department, BRAC University. He has more than three years of research and teaching experiences in computer science. He has been heavily involved in collaborative research activities especially in the fields of machine learning, deep learning, data mining, health informatics, computer vision, and natural language processing. He has published over 20 research articles in highly recognized journals and conference proceedings. He is working on several projects, such as airline industry customer review prediction, early Parkinson’s disease prediction, COVID-19 detection from medical image data, and diabetes prediction using deep learning and machine learning algorithms.

Rifat Sadik received the B.Sc. degree from the Department of Computer Science and Engineering, Jahangirnagar University, Dhaka, Bangladesh, where he is currently pursuing the M.Sc. degree. He has authored several journals and conference papers. Currently, he is conducting some research works based on machine learning and deep learning techniques in computer vision and natural language processing domains. His research interests include deep learning, artificial intelligence, natural language processing, and developing machine learning models for intelligent systems. He has received several scholarships and some significant ones are National Science and Technology (NST) Fellowship, and University Merit Scholarship.

M. F. Mridha (Senior Member, IEEE) received the Ph.D. degree in AI/ML from Jahangirnagar University, in 2017. He is currently working as an Associate Professor with the Department of Computer Science, American International University-Bangladesh (AIUB). Before that, he worked as an Associate Professor and the Chairperson at the Department of CSE, Bangladesh University of Business and Technology. He also worked as the CSE Department Faculty Member at the University of Asia Pacific and as a Graduate Head, from 2012 to 2019. His research experience, within both academia and industry, results in over 120 journal and conference publications. His research work contributed to the reputed journal of Scientific Reports (Nature), Knowledge-Based Systems, Artificial Intelligence Review, IEEE Access, Sensors, Cancers, and Applied Sciences. For more than ten years, he has been with the master’s and undergraduate students as a supervisor of their thesis work. His research interests include artificial intelligence (AI), machine learning, deep learning, natural language processing (NLP), and big data analysis. He has served as a program committee member in several international conferences/workshops. He served as an Associate Editor for several journals including PLOS ONE journal. He has served as a Reviewer for reputed journals and IEEE conferences like HONET, ICIIEV, ICCIT, IJCI, ICAE, ICSIPA, SCORED, ISIEA, APACE, ICOS, ISAIE, BIALC, ISWTA, IC3e, ISWTA, CoAST, icIVPR, ICSCT, JICT, and DATA21.