Sensing and Imaging (2022) 23:9
https://doi.org/10.1007/s11220-022-00379-1

ORIGINAL PAPER

Hand Detection by Two-Level Segmentation with Double-Tracking and Gesture Recognition Using Deep-Features

Debajit Sarma[1] · M. K. Bhuyan[1]

Received: 6 October 2021 / Revised: 23 January 2022 / Accepted: 2 February 2022 /
Published online: 22 February 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Vision-based hand gesture recognition involves a visual analysis of handshape, position and/or movement. Most of the previous approaches require complex gesture representation as well as the selection of robust features for proper gesture recognition. To eliminate the problem of illumination variation and occlusion in gesture videos, a simple model-based framework has been presented here using a deep network for hand gesture recognition. The model is fed with 'hand-trajectory-based-contour-images'. These images represent the motion trajectory of the hand for isolated trajectory gestures obtained via pre-processing steps—a two-level segmentation process and a double-tracking system. Deep features extracted from these images are used for estimating the hand gestures. Conventional machine learning methods involve tedious feature engineering schemes, while deep learning approaches can learn image features hierarchically from local to global with multiple layers of abstraction from a vast number of raw sample images. The feature learning capability of CNN architecture has been used here and it has shown outstanding results on three different datasets.

Keywords Hand gesture recognition · Human-computer interaction · Convolutional neural network · Hand-trajectory-based-contour-images · Double-tracking system · Two-level segmentation process

✉ Debajit Sarma
s.debajit@iitg.ac.in

M. K. Bhuyan
mkb@iitg.ac.in

[1] Department of EEE, IIT Guwahati, Guwahati, Assam 781039, India

1 Introduction

Generally nonverbal communication occurs through hand gestures, body postures, and facial expressions that makes almost two-thirds of all communication among human. While rest of the body indicates a more general emotional state, hand gestures can have specific linguistic content in it. One contemporary goal in *human-computer interaction* (HCI) design is to enable effective and engaging interaction. For example, vision-based hand gesture recognition (VGR) systems can enable contactless interaction in sterile environments such as hospital surgery rooms, or simply provide engaging controls for entertainment and gaming applications. However VGR is not as robust as standard keyboard and mouse based interaction. Issues such as sensitivity to size and speed variations of the gesturing body part, varying luminance conditions in the scene, a poor performance against complex backgrounds and the reliable detection of the gesturing phase due to various reasons like blurring or occlusion, etc, have limited the use of hand gestures as a reliable tool in interface design [1].

Numerous attempts have been taken by various researchers to deal with these inherent problems in a vision-based hand gesture recognition system. In this process, it is seen that a classical vision-based hand gesture recognition system usually consists of three main stages: first, acquisition, detection and pre-processing; second, gesture representation and feature extraction; and finally recognition [1, 2]. Among them, proper segmentation of the hand and robust and effective feature representation are two major challenges. Accurate segmentation of the hand from the captured images/videos remains a challenge for many preoccupied constraints like illumination variation, background complexity, and occlusion due to the articulated shape of the hand. Here in this work, the major issues that have been tried to address are -

Firstly, to get proper segmentation, the effect of illumination variations has been tried to minimize as much as possible. Various researchers have shown significant attention towards skin color information owing to its computational efficiency yet, robustness against rotations, scaling and partial occlusions. Generally, the chrominance cue information of skin color is less sensitive to illumination changes as compared to the luminance counterpart. Human skin color does not scatter randomly in a given color space, but clusters in a minimal region of a given color space. However, the color representation is not segregated into luminance and chrominance components in RGB color space, and all the three color components R, G, B are highly correlated to each other. Due to this reason, HSV and YCbCr color spaces are used in the skin segmentation process where the luminance component has been dropped, and only the chrominance component is used. Along with skin-color segmentation, motion information has also been used to segment the moving hand through the three-frame differencing method. These two methods are applied separately on the video frames and finally, the intersected portion is retained after a double check on the region of interest and the next process is carried out on this segmented hand region.

Secondly, tracking is done on the segmented region through a particle filter tracker which has been modified to handle the problem of blurring or occlusion to a great extent. Moreover, another CAMShift tracker tracks the centroid of each tracked hand region in each frame. In this way, a double-tracking system gives a single image called 'hand-trajectory-based-contour-image' that describes the trajectory of the gesture video. These images are used in recognition after feature extraction.

Lastly, the manual handcrafted features usually demand the user to have some prior knowledge and some preprocessing such as image transformation, etc, for proper recognition by a classifier. This has motivated the development of learning robust and effective representations directly from raw data and deep learning provides a plausible way of automatically learning multiple level features, by using multiple processing layers to learn image representations with multiple levels of feature abstraction. The recent popular deep learning methods like *convolutional neural network* (CNN) have demonstrated competitive performance in both image representation as well as classification. This work utilizes CNN to extract robust hand gesture features that can be used to recognize the hand gestures more precisely. The features are invariant to a certain extent to the shift and shape mutilations of the input characters. This invariance happens on the grounds that CNN embraces the weight-sharing procedure on the feature map. Unexpectedly, the hand-crafted feature extractor needs intricately planned provisions or even applies various sorts of features to accomplish the contortion invariance of the individual characters. Moreover, the topology of transcribed characters is vital on the grounds that pixels situated close to one another in space have solid associations. The elementary features like the corners, endpoints, and so on are made out of these close-by pixels. CNN utilizes the receptive field idea effectively to get such nearby visual provisions. Nonetheless, the hand-crafted feature extraction techniques disregard and lose such chronology of the contribution to most cases. Subsequently, the trainable features of CNN can be utilized rather than the hand-crafted feature to gather more delegate and significant data, particularly for the manually written digits.

The main contribution of this work is to present a hand gesture recognition model for isolated trajectory-based gestures using CNN architecture which is more suitable in spatial and semantic representation. This work basically is an extension of our previously published conference paper [3]. Especially, in order to enhance the hand gesture representation, a technique for converting a gesture video into a single image representing the trajectory of the gesture is introduced in the pre-processing step and we call these images 'hand-trajectory-based-contour-images'. Here the preprocessing scheme removes the constraint of variable illuminations inherently present in the original gesture videos. Through a proper tracking system, the problem of blurring or occlusion has also been tried to minimize. The experimental results demonstrate the effectiveness of the model achieved through these steps: preprocessing that removes the illumination variation and occlusion problem cooperating in raw data and the salient features learned through CNN for hand gesture recognition.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed hand gesture recognition model via CNN. The experimental results and analysis based on the proposed method are shown in Sect. 4. The last section summarizes this study and proposes some future direction of work.

2 Related Work

Vision-based hand gesture recognition for natural human-computer interaction is still an active research field [4–9]. A lot of early works prevailed in state-of-the-art hand gesture recognition that requires prior knowledge for designing hand-crafted features [4–6]. Different spatio-temporal descriptors such as Fourier descriptor (FD) [4], scale-invariant feature transform (SIFT) [5], histogram of oriented gradients (HOG) [6] are used to recognize gestures. Different template matching approaches like dynamic time warping (DTW) [7] and various state-space models like hidden Markov models (HMM) [8] and conditional random fields (CRF) [9] have been widely used as gesture classifiers. SVM is another famous gesture recognition classifier [5, 6]. A problem with many of these approaches is that a large variety of gestures executed by different people is very difficult to match. Hence, robust classification of gestures under widely varying lighting conditions with complex backgrounds, and from different subjects is still a challenging problem.

Indifference to hand-crafted features, there is a growing trend towards feature representations learned by deep neural networks. The main reason for deep learning having an upper hand over the traditional machine learning methods is the fact that the issue of “handcrafted features" can be addressed by deep learning. The feature learning mechanism of deep learning at different levels of representation is fully automatic, thereby allowing the computational model to implicitly capture intricate structures embedded in the data. In deep learning, higher-level features are defined in terms of lower-level features. 2D-CNNs are such a type of deep network that can act directly on raw images *i.e.* it can handle 2D images. Whereas 3D-CNN models, also called C3D, act on videos for action/gesture recognition. Recently, deep convolutional neural networks have been successful for both feature extraction and classification in various recognition challenges [10–15].

The unique color characteristic of human skin makes color-based skin segmentation very useful. However, the compactness of the skin colored clusters is not the same for all the color spaces. RGB is the most primitive color space in computer vision. However, in RGB the color representation is not segregated into luminance and chrominance components, and all the three color components R, G, B are highly correlated to each other. So, researchers have adopted various other types of color cues. One of the earliest methods of skin detection is proposed by Sobottka and Pitas [16]. They proposed a skin detection boundary along S and H channels in HSV color space. Later, Tsekeridou and Pitas proposed a modification [17] to this method for face region segmentation in an image watermarking system [18]. Hsu et al. [19] proposed a boundary rule based on YCbCr color space. The authors observed that the shape of the skin tone cluster in Cb-Cr space can be approximated as an elliptical structure where the cluster location depends on luminance Y. In [20], Shaik et al. compared HSV and YCbCr spaces for skin detection using a boundary-based method. Kukharev and Nowosielski proposed another set of skin detection rules [21] using RGB and YCbCr color spaces.

However, skin segmentation becomes very challenging when the color information is not available or when there are multiple skin-colored scenes in the background. Basically, when prior knowledge of the color of the moving object is not available, pixel-level change can provide powerful motion-based cues for detecting and localizing objects. Various approaches for moving object detection using pixel-level change can be background subtraction, inter-frame difference or three-frame difference. The background subtraction method is extremely sensitive to the changes in illumination. Moreover, stabilized background detection is always a costly affair making it vulnerable for long and varied video sequences [22]. Apart from this, the choice of temporal distance between frames is a tricky question. It depends on the size and speed of the moving object. Conversely, inter-frame difference methods can easily detect motion but show poor performance in localizing the object. The three-frame difference approach uses previous, current and future frames to localize the object in the current frame. Using future frames introduces a lag in the tracking system, but this lag is acceptable if the object is far away from the camera or moves slowly relative to the high capture rate of the camera. But three-frame difference method has some other advantages for which we adopted this method and it will be discussed in the implementation section.

Tracking of the hand in a gesture video can be very difficult as the movement of the hand can be very fast and its appearance can change vastly within a few frames. In such cases, model-based algorithms like mean-shift [23], Kalman filter [24], particle filter [25] are some of the methods used for tracking. The mean-shift is a purely non-parametric mode-seeking algorithm that iteratively shifts a data point to the average of data points in its neighbourhood (similar to clustering). However, tracking often converges to an incorrect object when the object changes its position very quickly in the two neighbouring frames. Because of this problem, a conventional mean-shift tracker fails to position a fast-moving object. In [26], a modified version of mean-shift algorithm has been used to continuously adapt mean-shift (CAMShift) where the window size is adjusted so as to fit the gesture area reflected by any variation in the distance between the camera and the hand. Though CAMShift performs well with objects that have a simple and constant appearance, it is not robust in more complex scenes. The motion model for the Kalman filter is based on the assumption that the velocity is relatively small when objects are moving, and therefore, it is modeled by a zero mean and low variance white noise. One limitation of the Kalman filter is the assumption that the state variables are based on Gaussian distribution, and thus the Kalman filter will give incorrect estimations of state variables that do not follow a linear Gaussian environment. The particle filter is generally a better method than the Kalman filter because it can consider non-linearity and non-Gaussianity. The main idea of the particle filter is to apply a weighted sample particle set to approximate the probability distribution, i.e., the required posterior density function is represented by a set of random samples with associated weights and estimation is done on the basis of these samples and weights. Both Kalman filter and particle filter have the disadvantage of the requirement of previous knowledge in modeling the system. Kalman filter or particle filter can be combined with the mean shift tracker for precise tracking.

Comaniciu et at. [27] proposed a model based on the color histogram for tracking hands. The color histogram of the hand detected from the video was used initially as the information for shifting the hand region and track it in the consecutive frames of the video sequences. The limitation of the system was that it can track the object if the background color has a different color other than the tracked object. To remove this difficulty, a method was proposed by Guo et at. which combined two additional information i.e. AdaBoosting and background removal along with the color information [28]. The limitation of this approach was that the background model has to be known beforehand and the tracking object should not be included in the background model. In [29], the CAMShift algorithm was proposed for tracking objects, but the constraint faced by this algorithm was that it was not able to handle occlusion. Later Shi and Tomasi [30] used minimum Eigen feature points to track the target object. The advantage of this model was that it does not depend on the color information. But this tracker faced difficulty in tracking objects with the long video sequences. This was because the feature points went on decreasing with consecutive frames of the video may be due to the change in illumination or shape of the hand or occlusion. KLT tracker was used by Kolsch and Turk [31] to track the hand. The major problem was that it was not able to track if the target object suffers from any shape transformations. Asaari et at. [32] incorporated eigenhand with the adaptive Kalman filter for tracking hand. They proved that their algorithm could be suitable for challenging environments, but it fails when there are large-scale variations and changes in poses of the gesturing body part. In [33], authors have detected hand movement using Adaboost with the histogram of oriented gradient (HOG) method. But this tracking algorithm is very sensitive to illumination variation.

Here in this work, we have segmented the moving hand through skin segmentation and motion-based segmentation through the frame-differencing scheme. The luminance counterpart in skin segmentation is also suppressed for segmentation under varying illumination conditions. To minimize some of the prevailing tracking problems like changes in the hand shape or blurring or occlusion, a double-tracking system consisting of CAMShift and particle filter has been adopted. For the measurement model of particle filter, a histogram of oriented gradients (HOG) has been applied. The details of the implementation are given in the methodology section.

3 Proposed Hand Gesture Recognition Methodology

This work presents a model for the recognition of isolated hand gestures under constrained environments like varying illumination, occlusion, or blurring conditions (shown in Fig. 1). This has been one of the substantially challenging tasks in gesture recognition. The main aim of this work is to build a model that can correctly classify an instance of the test gesture that actually belongs to one of the pre-defined classes. For this model, the first step is to segment the hand region effectively followed by tracking the moving hand. From these steps, some images are generated which basically represent the trajectory of the hand for the whole gesture. Thus the preprocessing part yields some 2D images called hand-trajectory-based-contour-images from gesture videos. In the next stage, CNN is chosen as feature extractor due to its inherent ability to learn features directly from raw images. It generally gives improved recognition performance if there is no over-fitting. In case of over-fitting various approaches like regularization, increase in training samples through

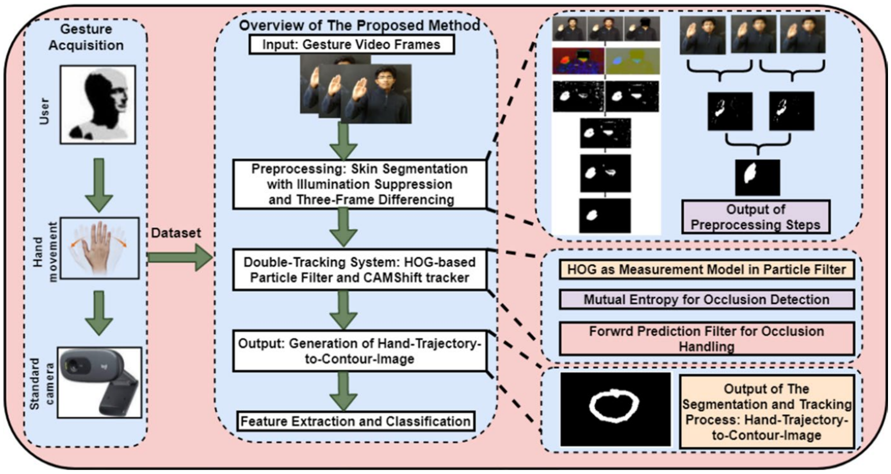

Gesture Acquisition
User
Hand movement
Standard camera
Dataset
Overview of The Proposed Method
Input: Gesture Video Frames
Preprocessing: Skin Segmentation with Illumination Suppression and Three-Frame Differencing
Double-Tracking System: HOG-based Particle Filter and CAMShift tracker
Output: Generation of Hand-Trajectory-to-Contour-Image
Feature Extraction and Classification
Output of Preprocessing Steps
HOG as Measurement Model in Particle Filter
Mutual Entropy for Occlusion Detection
Forwrd Prediction Filter for Occlusion Handling
Output of The Segmentation and Tracking Process: Hand-Trajectory-to-Contour-Image

Fig. 1 Block diagram of our proposed hand gesture recognition framework

data augmentation, etc, are adopted. But we have adopted another method which will be explained in the experiment section pertaining to a specific dataset.

3.1 Pre-processing

In the pre-processing stage, detection, segmentation and tracking all-together play a crucial role, because the accuracy of the VGR system depends on it. A gesture in our case is the motion of the human hand particularly the palm region along with fingers either in folded or open mode. So basically the palm region is the region of interest (ROI) in our case. A process for segmenting the hand region using skin color detection and motion-based segmentation followed by a tracking algorithm is discussed here. Segmentation refers to picking only the palm region leaving the rest part in each frame and thus creating a binary image. A flowchart for hand segmentation is shown in Fig. 2. Tracking here refers to locating the palm in each frame. The task at hand is to effectively build an approach to track the segmented palm and to learn the temporal evolution of its motion. Temporal evolution is about the trajectory of the ROI for the entire gesture. Thus, the entire trajectory of a gesture is converted into a single image consisting of the contour of the gesture trajectory through the simultaneous process of segmentation and tracking.

3.1.1 Skin Segmentation

The objective of this stage is to extract the hand region from the image frame. The hand region is obtained in this stage by the following steps: face detection and removal, change of color-space for illumination compensation, hand region detection, morphological filtering and smoothening followed by retrieval of the largest connected area *i.e.* the hand (shown in Fig. 3). Identifying the range of skin color of the person in the image frame is a problem as the pixel intensities are subjected

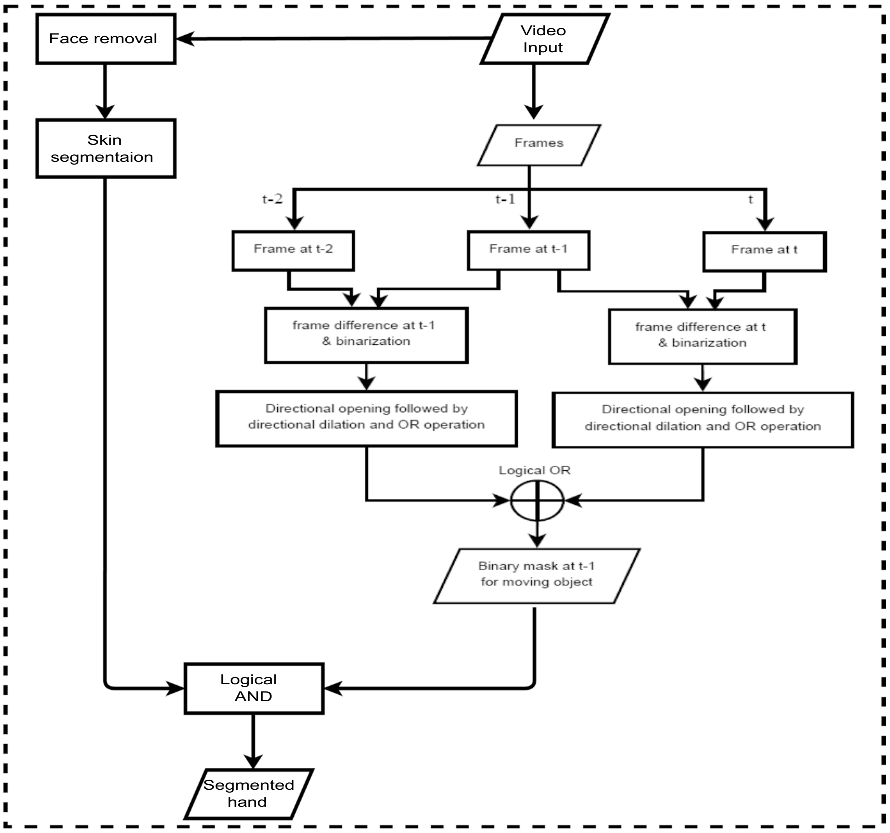

Face removal
Video Input
Skin segmentaion
Frames
t-2
t-1
t
Frame at t-2
Frame at t-1
Frame at t
frame difference at t-1 & binarization
frame difference at t & binarization
Directional opening followed by directional dilation and OR operation
Directional opening followed by directional dilation and OR operation
Logical OR
Binary mask at t-1 for moving object
Logical AND
Segmented hand

Fig. 2 Flowchart for hand segmentation using skin segmentation and three-frame differencing

to change in illumination. Variation in size and color among people also poses a problem. Moreover, human skin color does not fall randomly in a given color space but clustered at a small area in the color space [19]. So, the color needs to be represented in a color space where the skin class is most compact in order to be able to tightly model the skin class. The RGB color-space is not perceptually uniform, which means distances in the space do not linearly correspond to human perception. In addition, RGB color space does not separate luminance and chrominance, and the R, G, and B components are highly correlated. The luminance of a given RGB pixel is a linear combination of the R, G, and B values. Therefore, changing the luminance of a given skin patch affects all the R, G, and B components.

To tackle the illumination variation problem, RGB images are transformed into HSV and YCbCr color spaces. HSV and YCbCr are two color spaces that separate the chrominance ([H S] or [Cb Cr]) and luminance (V or Y) components [34]. CIE-Lab is another color-space with separable luminance and chrominance part with lightness variable L^* and chromaticity indices a^* and b^*. But the main problem with CIE-Lab ($L^*a^*b^*$) color space is that some color differences

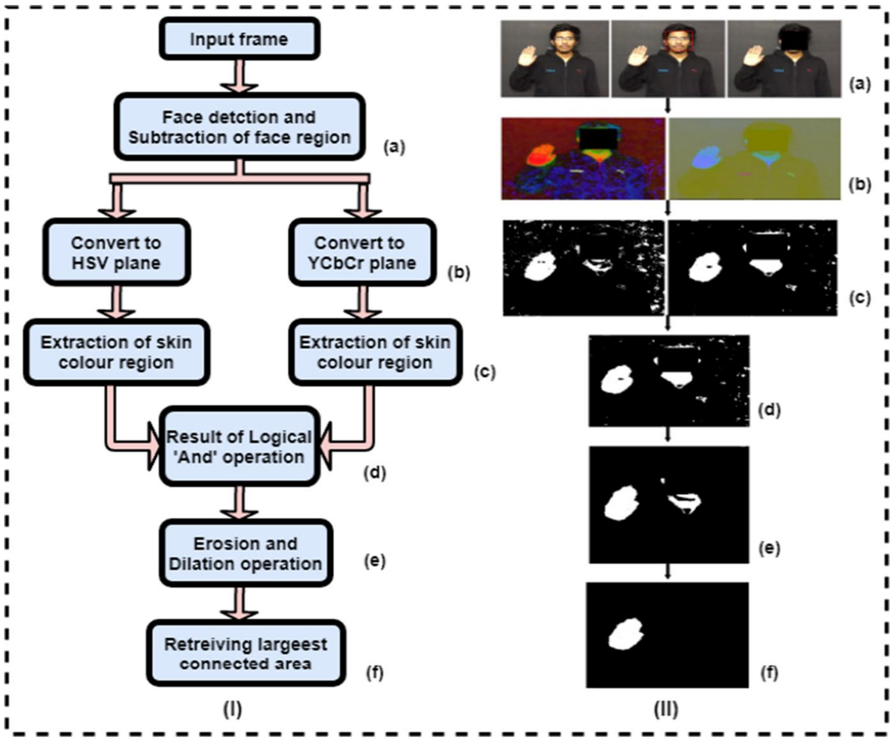

Input frame
Face detction and Subtraction of face region
(a)
Convert to HSV plane
Convert to YCbCr plane
(b)
Extraction of skin colour region
Extraction of skin colour region
(c)
Result of Logical 'And' operation
(d)
Erosion and Dilation operation
(e)
Retreiving largeest connected area
(f)
(I)
(a)
(b)
(c)
(d)
(e)
(f)
(II)

Fig. 3 Hand segmentation steps: (I) Processing steps and (II) Corresponding output where **a** Face detection and removal, **b** RGB to HSV and YCbCr, **c** Extraction of skin, **d** Logical 'AND', **e** Erosion and dilation, and **f** Smoothening and retrieval of the largest connected area

are evaluated differently between the color difference and the human eye. This is because the color discrimination threshold of the human eye greatly differs from the range of color differences defined by CIE Lab [35]. So, in our approach, we will stick to HSV and YCbCr color spaces. One of the major advantages of using these color spaces in skin detection is that a skin color boundary can be specified intuitively by a user. The HSV based detection is well suited for images with uniform background. In the case of YCbCr color space, transformation and efficient separation of color and intensity information is easy as compared to HSV. The skin color cluster shows more compactness in YCbCr space as compared to other spaces [19]. Moreover in YCbCr space, skin and non-skin colors show minimum overlap under different illumination conditions. So YCbCr color space works for the complex color images with uneven illumination. Here, both the color-spaces are used with an adjustable range to suit the color space of the person for segmentation in effective and efficient way. The output images in each steps of segmentation process is shown in Fig. 3. First, the face region in the frame is removed by using an inbuilt Viola-Jones [36] face detection algorithm (using haar cascade filter). After removing the face region, the RGB image is converted to HSV and

YCbCr images, and sub-sequently thresholding is done to the chrominance components of both HSV and YCbCr color spaces. The threshold is determined by computing histogram of all the components. By ignoring the luminance channel, the variation in background illumination factor can be reduced, and thus making it suitable to be used for skin color segmentation. A logical AND operation is performed between the images to obtain the skin likely region. After getting the skin color segmented regions, two approaches are tried to get the ROI: in first approach, morphological operations like erosions and dilations of convex hull technique are performed on the segmented region. Erosion first deletes the small white spots in the image followed by dilation which fills the gaps. The largest connected segment obtained corresponds to the palm region of the hand; and in second approach a median filter is used to remove the *salt and pepper* noise. The median filter replaces each pixel in the image with the median value from the square neighbourhood. To remove Gaussian noise from the input image, a two dimensional Gaussian filter is used. Then the resulting image is binary thresholded using Otsu thresholding [37] where ROI is assigned white color. After the contours are obtained from the previous step, the region with maximum connected area is chosen as the palm region.

3.1.2 Motion-based Segmentation

Assuming the background is static, the frame difference method is one of the easiest ways to detect moving objects in a frame. However, the frame difference algorithm suffers from some limitations like—the occurrence of ghost foreground regions and foreground apertures. Ghost foreground regions occur due to the motion of the objects. During frame differencing, an ambiguity may occur between real foreground regions and ghost foreground regions. The other drawback of frame difference is the occurrence of foreground object aperture (FOA). FOA occurs if the object is textureless and/or the intensity gradient of the image is significantly less. The probability of occurrence of FOA is significantly high in the case of moving skin regions as they have less texture and intensity gradient. In order to avoid the occurrence of ghost foreground regions, Kameda and Michihiko [38] proposed a 'double-difference of frame' (DDF) or three-frame difference method. In this method, three frames at time $t-2$, $t-1$, and t are selected. The DDF method performs a logical AND operation over thresholded difference frames between frames at $t-2$ and $t-1$, and frames at $t-1$ and t. The DDF algorithm produces a narrow region for a moving object if the object has less texture and/or intensity gradient. The use of morphological operations can reduce FOA in a difference frame. A dilation along the direction of motion of an object can reduce FOA with the inclusion of ghost foreground regions. Motivated by this fact, a morphological enhancement-based three-frame difference method is proposed to detect moving hand regions (shown in Fig. 4). In our proposed method, morphological dilation is applied to each of the thresholded difference frames. In order to reduce inclusion of background regions in foreground regions of a thresholded difference frame, dilation should be performed in the direction of motion of the foreground objects. However, in case of articulated objects like hands, foreground motion could be complex. We approximate complex movements

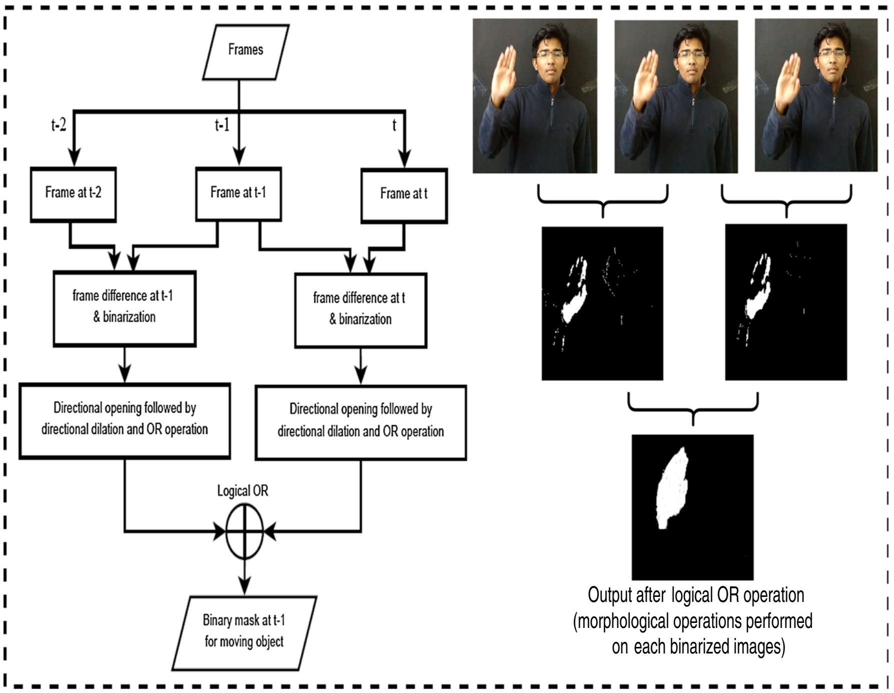

Frames
t-2
t-1
t
Frame at t-2
Frame at t-1
Frame at t
frame difference at t-1 & binarization
frame difference at t & binarization
Directional opening followed by directional dilation and OR operation
Directional opening followed by directional dilation and OR operation
Logical OR
Binary mask at t-1 for moving object
Output after logical OR operation (morphological operations performed on each binarized images)

Fig. 4 Flowchart of three-frame difference method with corresponding output for motion-based segmentation of the hand

as a combination of motions in four directions—$0°, 45°, 90°, 135°$ with respect to the horizontal direction. Directional opening can be used to select a region in a particular direction. After directional opening, a dilation in the perpendicular direction of the opening process grows a region in the direction of its motion. Finally, a logical OR operation is performed on the two morphologically enhanced thresholded difference frames to obtain a moving hand mask. The OR operation helps in including more moving skin regions between the consecutive frames.

3.1.3 Tracking of the Hand using a Double-Tracking System

The next step is to track the hand after successful hand detection through the segmentation process. Existing tracking algorithms suffer from various flaws and therefore there is a need to develop an algorithm that overcomes some of the existing constraints like changing the shape of the hand, a fixed searching window for the target hand, occlusion or blurring of the hand while tracking, etc.

In traditional tracking algorithms like CAMShift or particle filter, the initial tracking region needs to be selected manually. But to make a robust system, automatic selection of tracking regions is necessary. In this proposed system, initialization of the first tracking window has been made automatic by considering the detected hand as the initial tracking window. Detection should be proper to set the tracking window properly as the initial tracker will decide the accuracy of the complete trajectory of the gesture video. After the initialization of the tracking region, a proper selection of features is essential. Here we are using a particle filter for tracking purposes along with a histogram of oriented gradients (HOG) features as a measurement model. As the video progresses, detected particles may start decreasing and a time comes when tracking is lost due to the loss of all particles. This is due to the change in hand shape or occlusions or blurring of the target hand. To reduce these challenges, a modified system is defined using the re-detection of the hand according to the newly defined area. So, when the particles reduce in number, detection is performed again to eliminate the issue of tracking at reduced particles as soon as the number of observable particles decreases. Until detection, the searching area is doubled so that the resulting area is greater to reach the visible particles and then skin filtering and three-frame differencing are done. Logical AND operation is performed between the skin-segmented and three-frame differenced images to get the target hand. Then new particles are generated according to the HOG features of the newly identified region-of-interest and particles are inserted again. The particle filter tracker is run again, and visible particles are counted and then the complete process of particle filter is again performed till the end of the video-frames. To form the gesture trajectory, centroids of the detected region are calculated and marked each time through CAMShift tracker equations and then the trajectory is generated by joining the smoothed centroids.

Algorithm 1 Proposed double-tracking algorithm

```
Data: Read the initial video frame along with the segmented hand region
Output: Hand-trajectory-based-contour-image
Initialize: Initialize the particle filter tracker according to the HOG features of the segmented region and
Set: Set optimum number of particles ($Thr$) of the particle filter to represent the hand region
Initialize: Initialize $N$ particles of the particle filter with associated weight $1/N$ in the segmented area
Loop:
If($N \geq Thr$)
    • Change the detected area according to the segmented hand position
    • Apply CAMShift algorithm to find and save the centroid of the detected area
elseif()
    • Double the searching window
    • Do segmentation through skin and motion segmentation after the hand detection
    • Detect the new region-of-interest and find HOG features
    • Initialize the particle filter according to the new HOG features and insert $N$ particles
    • Count the number of particles
    • Go to Loop till the end of the video frames
```

- **Modified particle filter framework:** During the gesture trajectory, the captured video sequences may suffer from blurring and/or occlusion. In such a scenario, tracking an object becomes very difficult. Detection of such events during trajectory is very crucial in gesture recognition. Whenever the particles lose the target, they gradually increase their search area until they find the target again. To control such types of scenes of occlusions or blurring, an uncertainty factor is introduced to increase the search space of the particles of the particle filter tracker. This part of the work is motivated by works like [39] and [40]. Cumulative measure by all the particles in the particle filter is taken for each frame of the gesture video sequence and mutual entropy is proposed as a measure. Mutual particle entropy, H is given by:

$$H = \sum_{i=1}^{N} w_k^i log(w_k^i) \tag{1}$$

where w_k is the weight of particle at k^{th} instant. When the particles converge to the target, mutual entropy tends to decrease, and it increases when the particles lose the target either because of occlusion or blurring. Thus, an uncertainty factor is introduced, which is related to mutual entropy given by:

$$E = 1 - e^{-H^{\gamma}} \tag{2}$$

where γ is a constant laying in between 0 and 1, H is the mutual particle entropy. The purpose of this function is to keep the value of the uncertainty factor close to 0 when entropy is low and close to 1 when entropy is high thus scaling the measurement noise accordingly.

- *Particle update:* For a particle filter framework the state update equation is given by:

  $$\vec{X}_k = \vec{X}_{k-1} + k\vec{\mathcal{N}} \tag{3}$$

  where $\vec{\mathcal{N}}$ represents the measurement of noise and k is a scaling constant. In this work, the modified framework has the state update equation given by:

  $$\vec{X}_k = \vec{X}_{k-1} + E\vec{\mathcal{N}} \tag{4}$$

  where $E = 1 - e^{-H^{\gamma}}$. Thus, for a very confident measurement, the noise gets scaled down to a low value, and the search space becomes small and vice versa.
- *Measurement model:* The weight assigned to each particle depends on the closeness to the neighborhood target features. If the measurement is closer to the target, then higher weights will be given to the particles. These weights give the probability of the presence of the target in the neighborhood of each particle. Different measurement models have been proposed in the literature. In this work, histograms of oriented gradients (HOG) features are generated from the segmented hand portion for each frame and used for measurement. HOG basically returns the counts of occurrences of the gradient orientation in localized portions of an image. This portion is called a cell. The cells representing the HOG features are shown in Fig. 5. This represents the image-oriented gradients and it is different for the hands of different persons due to varied shapes and sizes. Even the HOG features change along the trajectory of the gesture due to the variation in shape and size of the moving hand. By

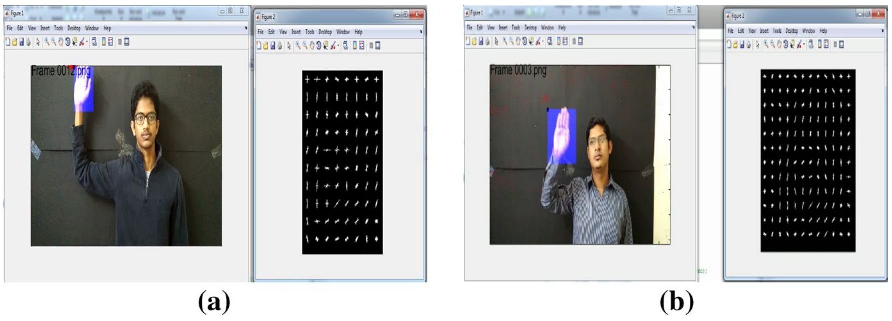


Fig. 5 Showing hand region detection through HOG-based particle filter and sample of HOG features for different subjects

this measurement model, we try to represent the hand model along the gesture trajectory. The weights assigned to the particles in each frame in the neighborhood of the hand regions are higher than the weights of the particles associated with the non-region-of-interest.

– *Occlusion handling:* When occlusion is detected with the use of uncertainty measure, a forward prediction filter is activated. It predicts the next state as a combination of a few previous states. The details of the method can be found in the paper [39]. Gang Yu et al. [41] suggested object tracking in case of occlusion by using an incremental PCA-based method. But Fig. 6 shows the effectiveness of the proposed method compared to other methods like the incremental PCA-based approach.
– *Re-sampling of particles:* This is the final step in every iteration of the particle filter algorithm. Here, N particles are randomly picked from the existing particle set according to the updated weights. Thus the particles with lower weights will be less picked and will finally die out. Whereas the particles with higher weights will be picked more than once and another set of N particles will be chosen from the existing particles.

- **CAMShift algorithm:** The CAMShift algorithm uses a color histogram of the moving target as target mode and is thus known as the target tracking algorithm.

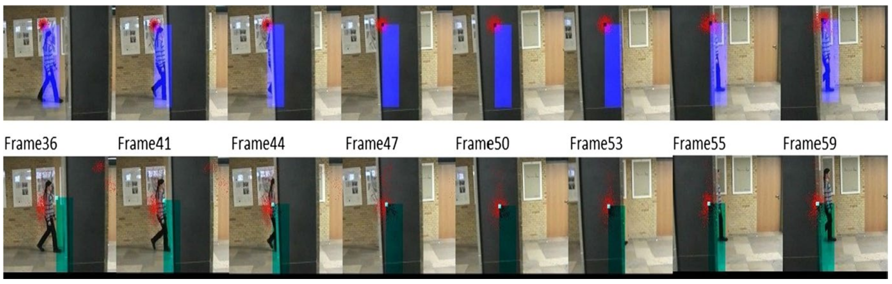


Fig. 6 Tracking results using the proposed method (first row) [39] and PCA-based method (second row) [41]

CAMShift algorithm when used as a mean shift algorithm has advantages like simplicity and fast speed to process and converge. The window size of this algorithm is constant so that the object location cannot be exactly detected when the size of the object changes and hence tracking loses the path sometimes. Moreover, it also loses its path when there is some skin-colored object that causes occlusion. To mitigate these problems, we have to go for a double-tracker system. As explained above, the particle filter tracker is used to keep track of the segmented hand region frame after frame according to the hand shape defined by HOG features. And, CAMShift is used to capture the trajectory of the centroid of the palm in the sequence of frames in the input gesture video to generate the gesture trajectory. The centroid is calculated from the first-order moments of the pixels (white) of the maximum connected area obtained from the previous steps. The end of the gesture is detected by the absence of a connected area greater than the threshold. The CAMShift algorithm with related equations [9] are given below. • Computation of 0^{th} and 1^{st} moment: The 0^{th} and the 1^{st} order moments are defined as:

$$M_{00} = \Sigma\Sigma I(x,y),\ M_{10} = \Sigma\Sigma xI(x,y),\ M_{01} = \Sigma\Sigma yI(x,y) \tag{5}$$

In the above equations, *I(x,y)* is the pixel value at the position *(x,y)* in the image. • Compute the new center of feature: As the image is binary, the centroid of the hand in a frame is also the centroid of the total frame which is given by:

$$x_c = \frac{M_{10}}{M_{00}} \text{ and } y_c = \frac{M_{01}}{M_{00}} \tag{6}$$

- **Smoothening of the obtained gesture trajectory:** The gesture trajectory is traced out by joining all the calculated centroids in a sequential manner. The trajectory obtained by joining the centroid points could be noisy due to various reasons. Some common reasons are - the points being too close, varying the hand shape/orientation could lead to isolated points far from the trajectory, some unintentional movements and trembling of hand *etc.* The final gesture trajectory is obtained by a technique of considering an average of the mean values of three successive video frames which reduces various noises to a great extent.

$$(\hat{x}_t, \hat{y}_t) = \left(\frac{x_{t-1} + x_t + x_{t+1}}{3}, \frac{y_{t-1} + y_t + y_{t+1}}{3} \right) \tag{7}$$

Thus, a dynamic hand gesture (*DG*) can be interpreted as a set of points joined together in a spatio-temporal space (Fig. 7):

$$DG = \{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \dots, (\hat{x}_t, \hat{y}_t)\} \tag{8}$$

3.2 CNN Network Architecture and Training

A convolutional neural network (CNN) architecture is used for feature extraction and gesture recognition on the segmented and tracked images obtained in the image

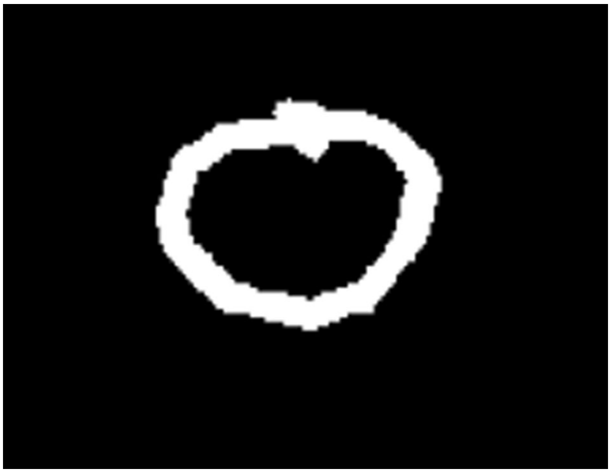


Fig. 7 Tracking gesture for English alphabet ‘O’

processing section. A simple custom structure with two convolution layers and two fully-connected layers is chosen here since more complex networks with more layers can become overfitting and they try to memorize a particular work making it less generic. For training purposes, we have used EMNIST (letters) dataset [42] as our training dataset. And, for testing, we have used three databases. The details of network architecture and training are given below.

3.2.1 Network Architecture

The CNN architecture used for this work is shown in Fig. 8. The first layer of the network is a convolutional layer. The design of this layer comprises 32 filters, with a kernel size of 5×5 each. The activation function used is *Rectified Linear Units*

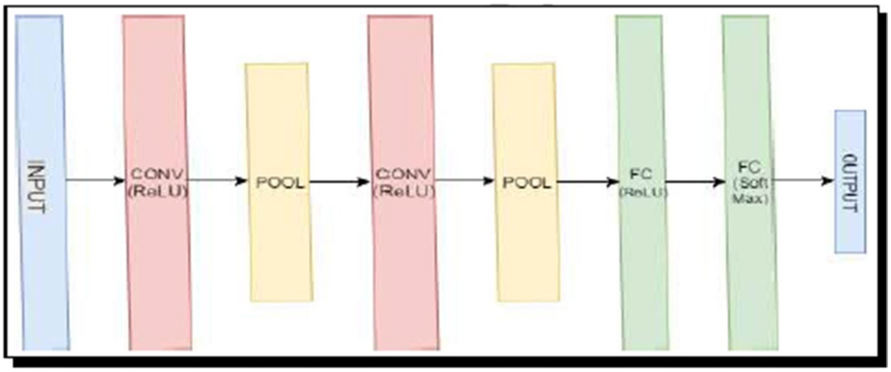


Fig. 8 CNN architecture for hand gesture recognition (inspired by LeNet [10]) CONV:Convolutional, FC:Fully connected

(ReLU) which generally outperforms sigmoid and tangent in speed for training without any tradeoff for accuracy. ReLU activation introduces non-linearities and offers less saturation. Efficient gradient back-propagation can be achieved by ReLU along with momentum [12]. After that, we use a max-pooling layer. A 2×2 box non-overlapping max pooling with stride 2 in both horizontal and vertical directions is applied. The next layer is again a convolution layer which consists of 64 filters, each with the size of 7×7 and ReLU activation function. The output of this step undergoes 2×2 box max-pooling. The weight of all the filters of these convolution layers gets updated during the training phase. The output feature maps can be seen to contain a few evident features from input as training proceeds. To avoid overfitting while learning, there is a dropout layer. The parameter for this layer is set to 0.4 which means the layer randomly excludes 40% of neurons (and weights associated with it) in the layer. It is configured to speed up the training process and reduce over-fitting. The 2D matrix data is converted to a column vector (one dimensional) in order to be processed by fully connected dense layers. A fully connected layer with 1024 neurons and ReLU activation function process the result from the previous step. A softmax activation function is used on the output fully connected layer to turn the outputs into probability-like values. These values are used for prediction.

3.2.2 Training

As already mentioned, the EMNIST dataset (letters) has been used as our training dataset. Here categorical cross-entropy (logarithmic) loss function is used as a cost function and *stochastic gradient descent* (SGD) with momentum is used as an optimization technique for convergence of the model. Momentum in gradient descent is a method in which some fraction from the previous update is added to the current update to compound the effect of repeated updates in a particular direction. As a result, the descent in the desired direction is faster. The learning rate of the model is slowed down with an increase in the number of epochs in order to allow the model to converge more accurately. With a deeper network, there is a problem of loss in training and testing. This is not because of overfitting, but due to the slow learning of a deep network. So, generally, the learning rate is varied in different steps to get rid of this. The choice of learning rate is 0.01 if the epoch count is less than 25 and is reduced to 0.001 for epoch count up to 50. To further promote slow learning, values of 1e-4 and 1e-5 are used, till 75 and 100 epochs respectively.

4 Experimental Results

4.1 Databases and Experimental Set-up

The performance of the model has been tested on two publicly available datasets and our own in-house dataset. The publicly available datasets are: EMNIST (letters) dataset [42] and NITS hand gesture database (with no variation in gesticulation speed or pattern) [43]. Our in-house dataset is a limited dataset of English upper case letters and numerals 0-9 mimicking the structure of the EMNIST dataset. The dataset is created with the help of three subjects and the background is kept simple. In this experimentation, datasets that define the same classes which are upper case letters of the English alphabet are used. So, the same model that has been trained using EMNIST (letters) training dataset, is used for testing which reduces the burden of training requirements again and again. For testing purposes, EMNIST (letters) test images and the images obtained (through the process explained in the image processing section) from NITS hand gesture database and our own video dataset are used and the results are discussed in the next subsections. So, in a nutshell, our goal is to recognize the images of the English upper case alphabets that have been obtained from the three different databases. All experiments are performed in a workstation with Intel® Core$^{\mathrm{TM}}$i5-4570 CPU at 3.2 GHz and 8 GB in RAM without any GPU usage.

4.2 Results using EMNIST (Letters) Dataset

The EMNIST (Extended MNIST) [42] dataset is a set of handwritten character/digits derived from the NIST (National Institute of Standards and Technology) [10] database converted to a 28×28-pixel image format and dataset structure that directly matches the MNIST (Modified NIST) [10] dataset. The MNIST database is a large database of handwritten digits that is commonly used for training and testing various image processing and machine learning systems. The MNIST database of handwritten digits has a training set of 60,000 examples and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized to 28×28 and centered in a fixed-size image. The MNIST dataset was extended including handwritten letters by the name EMNIST which was published in 2017. It contains 2,40,000 training images and 40,000 testing images of handwritten digits and 1,24,800 and 20,800 handwritten images for training and testing respectively for letters. There are six different splits provided in this dataset and each is provided in two formats: binary and CSV (combined labels and images). The description of six categories of the EMNIST dataset is given in Table 1.

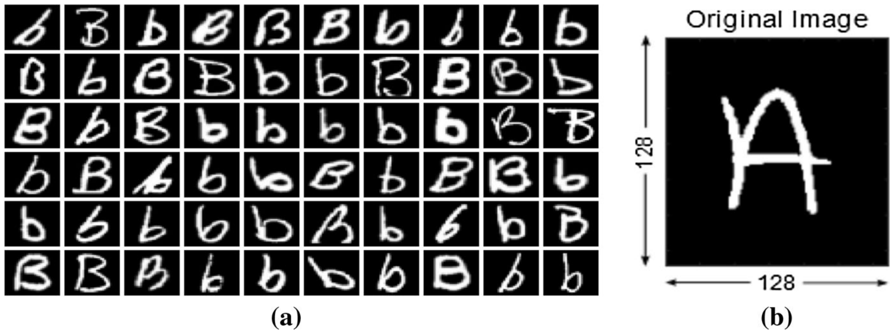


Fig. 9 **a** Different variants of 'B' and 'b' from EMNIST (letters) dataset, **b** One particular sample of 'A' from EMNIST (letters) dataset

Table 1 Categories of EMNIST dataset [44]

Categories	Classes	Training	Testing	Valid.	Total
By class	62 unbal[1]	697,932	116,323	No	814,255
By merge	47 unbal	697,932	116,323	No	814,255
Balanced	47 bal[2]	112,800	18,800	Yes	131,600
Digits	10 bal	240,000	40,000	Yes	280,000
Letters	26 bal	124,800	20,800	Yes	145,600
MNIST	10 bal	60,000	10,000	Yes	70,000

[1]unbal = *unbalanced*; [2]bal = *balanced*

EMNIST (letters) contains 1,45,600 characters of 26 balanced classes (shown in Fig. 9a and b). Our model is trained on 1,24,800 samples and tested on 20,800 samples (800 test samples per class).

The training and testing categorical cross-entropy losses and accuracies as a function of a number of epochs are shown in Figs. 10 and 11 respectively.

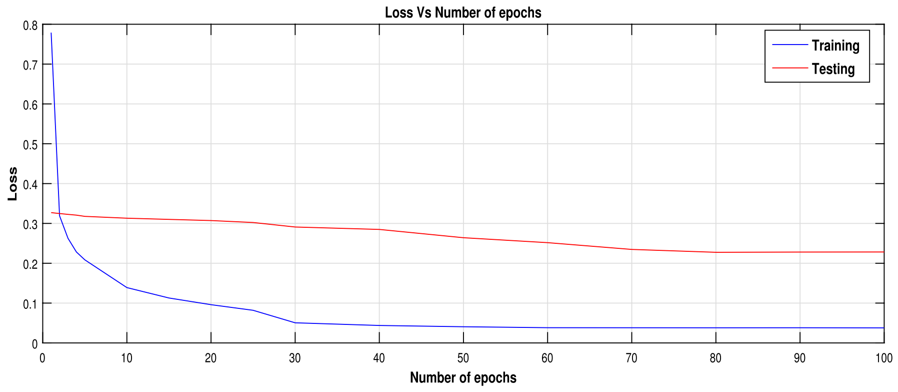


Fig. 10 Training and testing loss as a function of number of epochs for EMNIST (letters) dataset

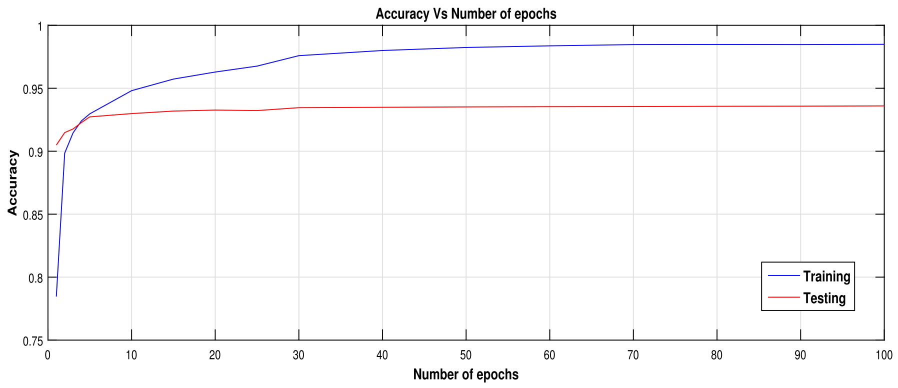


Fig. 11 Training and testing accuracy as a function of number of epochs for EMNIST (letters) dataset

Table 2 Comparison of error rate (%) with state-of-the-art on EMNIST dataset

EMNIST dataset	Linear classifier [42]	OPIUM classifier [42]	MRF-CNN [44]	Our method
Balanced	49.17	21.98	9.71	–
By class	48.19	30.29	12.33	–
By merge	49.49	27.43	9.06	–
Letters	44.22	14.85	4.56	6.40
Digits	15.30	4.10	0.25	–
MNIST	14.89	3.78	0.33	–

The comparison among state-of-the-art results for all variants of the EMNIST dataset with our result on the portion of the letters of the EMNIST dataset is given in Table 2. Peng [44] has obtained a better result compared to our method. There are two reasons for this improvement: First, the incorporation of Markov random field (MRF) based filter banks along with DCT and Gabor filters as prefixed convolutional operators to derive the feature maps. By combining Markov random field models, to extract salient information from the raw data, along with the convolutional neural networks, to implicitly learn high-level features, it has combined the respective strengths of MRFs and CNNs to bring the expressive power of a deep architecture. Another major difference is that [44] has used a deeper convolutional layer structure compared to our two-convolutional layer model. Our model has been kept simple so that it can also be applied in resource constraint environments without the use of GPUs. This comparison is shown here to make it clear that though our model is not state-of-the-art, still it is good enough to carry out our recognition procedure on the hand-trajectory-based-contour-images derived from real gesture databases. Due to the limited number of data samples in the gesture databases, the model is trained with the EMNIST image dataset and it is saved in the h5.py format to be used directly in the recognition system. Here data augmentation and transfer learning type approaches are not adopted due to limited sample size in the real gesture databases and are used only for testing.

4.3 Results using NITS Hand Gesture Database

NITS hand gesture database [43] is developed by the Speech and Image Processing Lab of NIT, Silchar, India in seven different categories with different variations in pattern and speed. Here, the database I have been used as it resembles our training dataset. The database consists of 40 gestures, which are alphabets (A-Z), numbers (0-9) and mathematical operators (+, -, /, *) (Fig. 14). Out of all these, we have used the alphabet gestures for testing purposes. The database is recorded in different sessions with multiple participants. Gestures are captured in an uncontrolled environment with both simple and complex backgrounds, and under varying illumination conditions (shown in Fig. 12). However, color cues show variations in the skin color in different lighting conditions, and also skin color changes with the change in human tribes, leading to many segmentation

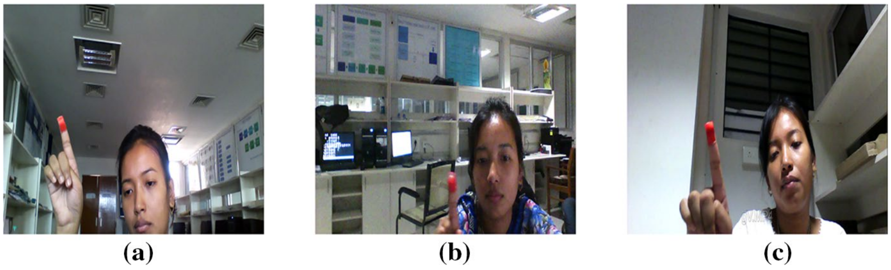
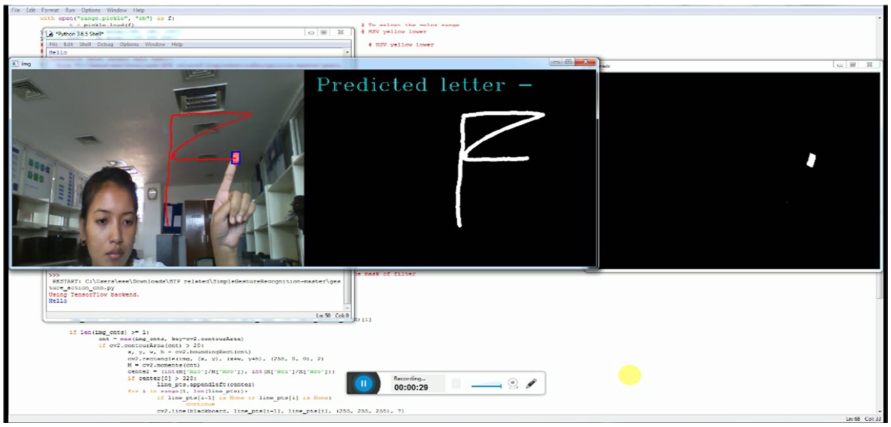


Fig. 12 Screenshots from NITS hand gesture database showing varying illumination conditions

issues and also create restrictions due to the presence of skin-colored objects in the background. To overcome the disadvantage of skin color detection, participants of the dataset have used markers on hand or fingers in the dataset which enhances segmentation accuracy leading to better performance.



```
with open("range.pickle", "rb") as f:
# To select the color range
# HSV yellow lower
# HSV yellow lower
```

```
Hello
```

```
>>>
 RESTART: C:\Users\eee\Downloads\STP related\SimpleGestureRecognition-master\ges
ture_action_cnn.py
Using TensorFlow backend.
Hello
```

```
if len(img_cnts) >= 1:
        cnt = max(img_cnts, key=cv2.contourArea)
        if cv2.contourArea(cnt) > 20:
                x, y, w, h = cv2.boundingRect(cnt)
                cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
                M = cv2.moments(cnt)
                center = (int(M['m10']/M['m00']), int(M['m01']/M['m00']))
                if center[0] > 320:
                        line_pts.appendleft(center)
                for i in range(1, len(line_pts)):
                        if line_pts[i-1] is None or line_pts[i] is None:
                                continue
                        cv2.line(blackboard, line_pts[i-1], line_pts[i], (255, 255, 255), 7)
```

Fig. 13 A snapshot of the gesture recognition system that has been used for user interface

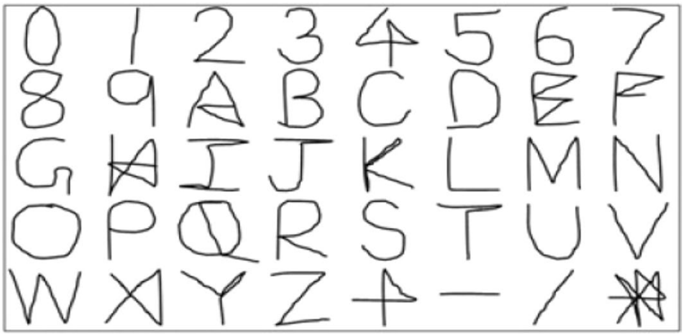


Fig. 14 Gesture set of NITS hand gesture database [43]

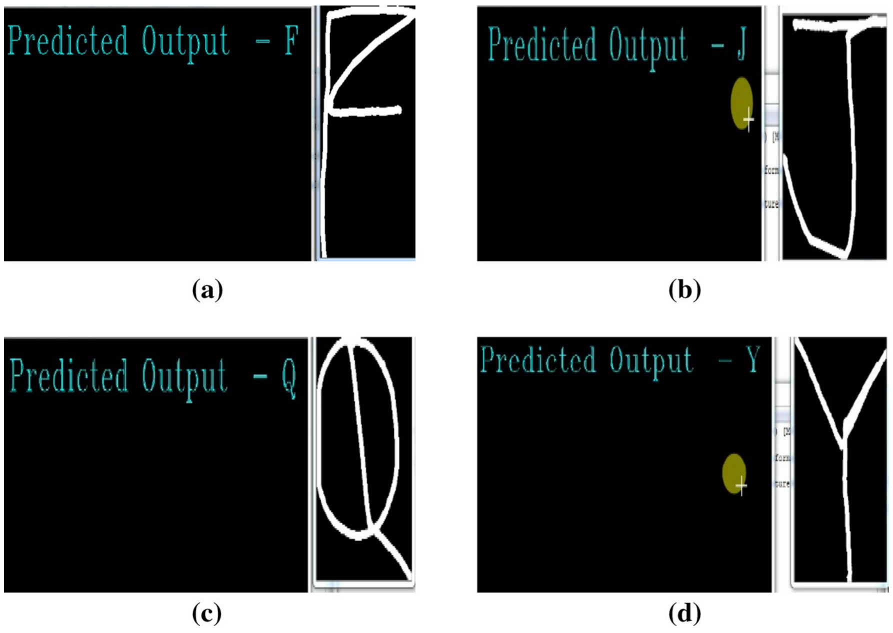


Fig. 15 Correct prediction of different test samples **a** 'F', **b** 'J', **c** 'Q', **d** 'Y'

In Figs. 13, 14, a snapshot of the gesture recognition system has been shown that is used as a handy user interface in the experimentation. Our model has been able to recognize most of the English alphabet gestures (shown in Fig. 15). However, there are a few gesture examples where the CNN classifier has misclassified as shown in Fig. 16. The model predicts the testing samples 'A' as 'K' shown in Fig. 16a. This is due to the structural difference of the testing sample 'A' with the training one as shown in Figs. 9b and 14. The same is the case with gestures 'I' and 'T' where the system has not been able to predict correctly mainly due to the similarity of the gesturing alphabets with some other training examples (Refer Fig. 16b and c). It is also seen that there is some minor structural difference in the test samples of 'E', 'F', 'H' and 'X' compared to training samples, still, our model has been able to recognize

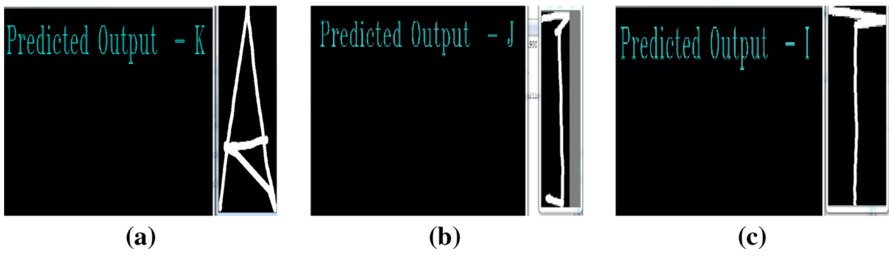


Fig. 16 **a**Prediction of test sample 'A' as 'K' due to its resemblance in training samples of EMNIST (letters) dataset, **b** Prediction of test sample 'I' as 'J', **c** Prediction of test sample 'T' as 'I'

them correctly (shown in Fig. 15a). So, in this way, our prediction accuracy of the whole system stands at 93.02% due to the above-mentioned misclassifications.

One easy way of increasing the accuracy performance by removing such type of ambiguity is the addition of different variants of the misclassified test samples in the training set similar to that of testing one. In the case of a limited database, an increase in training samples through data augmentation can be a very good option that basically diversifies the training process. However, in our case, we have trained the model on the EMNIST dataset, and the saved trained model has been used in the recognition system to test some gesture databases like the NITS gesture dataset and our in-house dataset. Though the training-testing accuracy curves on the EMNIST dataset suggest that the model is not overfitting, but through the experimentation, it is quite clear that it has tried to memorize only the particular training samples making it less generic. To overcome this problem, we have used the CNN structure only for feature extraction purposes and used a support vector machine (SVM) classifier for the recognition task. It is expected that this hybrid CNN–SVM model will show better performance compared to each individual classifier based on the fact that the hybrid system compensates the limits of individual classifiers by incorporating the merits of both classifiers. Since the hypothetical learning technique for CNN is equivalent to that for the MLP, it is an extension model of the MLP. The learning calculation of MLP depends on the Empirical Risk Minimization, which endeavors to limit the errors in the training set. At the point when the first isolating hyperplane is found by the back-propagation calculation, regardless of whether it is the local or the global minima, the training process stops and the calculation doesn't keep on further developing the isolating hyperplane arrangement. Thus, the speculation capacity of MLP is lower than that of SVM. Then again, the SVM classifier tries to minimize the generalization error on the concealed information with a fixed distribution on the training set, by utilizing the Structural Risk Minimization standard. The isolating hyperplane is a global ideal arrangement. It is determined by tackling the quadratic programming problem, and the margin between the two classes of training tests attains its maximum. Subsequently, the generalization ability of SVM is maximized to improve the performance. Through this process, we have been able to increase the recognition accuracy to a great extent and achieved satisfactory performances compared to the state-of-art method. This is explained in the next section.

4.3.1 Classification using SVM

An SVM is a supervised classifier for both linearly separable and linearly nonseparable data [45]. In a linearly nonseparable case, the input data is mapped to some higher dimensional space, where the data can be separated linearly. To do so, a nonlinear mapping is done through kernels. This mapping from lower to higher dimensional spaces makes the classification of the input data simpler and more accurate. There are many kernels used for this operation and we have chosen different SVM kernel functions such as linear, polynomial (quadratic), Gaussian and radial basis function (RBF) to compare the performance measure. For linear and polynomial auto kernel scale and for Gaussian and RBF, 0.79 and 1 are used as kernel scale and their accuracy performance is shown in Table 3. The same is shown in graph

Table 3 Results of accuracy performance with different SVM kernels

SVM kernel function	Kernel scale	Classification accuracy (%)
Linear KF	Auto	95.93
Quadratic KF	Auto	97.67
Gaussian KF	0.79	97.13
RBF KF	1	99.09

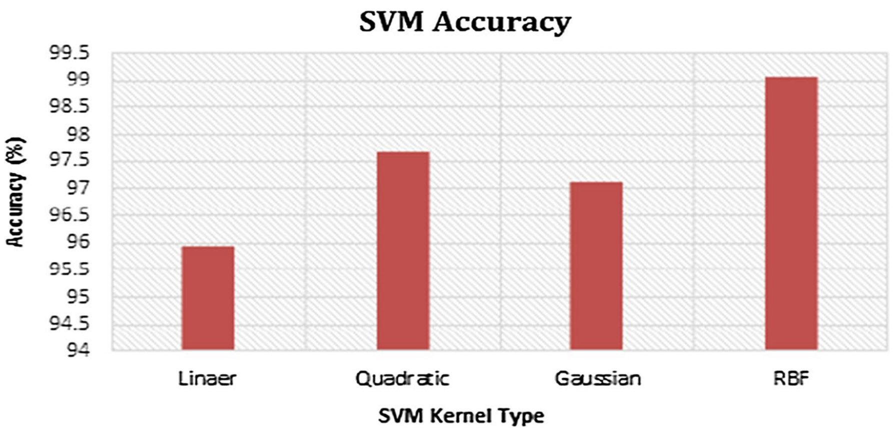


Fig. 17 Graph showing accuracy of SVM classifier with different kernels

form in Fig. 17. In a CNN architecture, the convolutional layers extract the features and the fully connected layers are responsible for recognition. The feature vector of dimension 2048 extracted from the output of the second fully-connected layer of the CNN structure is given as input to the SVM recognition classifier after L_2 normalization. It can be observed that SVM with RBF kernel provides the highest accuracy of 99.09% for the alphabet gestures and hence, we have chosen RBF kernel in the final model. The RBF kernel on two samples x_i and x_j, represented as a feature vector in some input space, is given by -

$$K(x_i, x_j) = \exp\left(-\frac{(\left\|x_i - x_j\right\|)^2}{2\sigma^2}\right), \quad \sigma > 0 \tag{9}$$

which can be written as -

$$\boldsymbol{K}(x_i, x_j) = \exp(-\beta\left\|x_i - x_j\right\|^2), \quad \beta > 0 \tag{10}$$

β is a parameter of a Gaussian RBF kernel to handle non-linear classification. In the experiments, we have used $\sigma = 0.7$ or $\beta = 1$ as the basis for trial and error. C is another parameter for the soft margin cost function, which controls the influence of each individual support-vectors. A large C gives low bias and high variance, whereas a small C gives higher bias and lower variance. So, this process involves trading error penalties for stability. The RBF function is straightforward that can achieve nonlinear mapping, and uses relatively few parameters. In our experiment, we have used a non-linear support vector machine (SVM) with a Gaussian radial basis function kernel with C as a parameter with values 1 and 10. The kernel trick is a strength of SVM. The risk of overfitting is less in SVM and has good generalization. There are basically two methods to use SVM for a multiclass problem [46]. The first approach is called "one-versus-one" (OVO) that construct one classifier per pair of classes and combine binary classifiers in a way to form a multi-class classifier by selecting the most voted class. So, $\frac{N(N-1)}{2}$ binary SVM classifiers are needed, each of them is trained on the samples of the two corresponding classes. The second method is called "one-versus-all" (OVA) and it considers all the classes of data in one optimization problem. In fact, for each classifier, the considered class is fitted against all the other classes, so for N number of classes, N SVM classifiers are required. In this work, we have used support vector machines (SVMs) that implement a one-versus-all (OVA) multiclass approach.

4.3.2 Comparison with State-of-the-Art

A comparison among state-of-the-art results for NITS hand gesture database with our result is given in Table 4. In [47], bare hand English alphabet gestures are classified with the help of handcrafted features using different classifiers like kNN, SVM and ANN. [43] has used handcrafted features in SVM and ANN classifiers for classification. Ours is the first work on this dataset where deep network features are used along with an SVM classifier and it has been able to achieve state-of-the-art performance.

4.4 Results using Our In-house Dataset

One limited in-house dataset has been created with the help of three participants and is mainly used for testing purposes. It consists of 78 videos with 26 classes of English upper case letters. For simplicity, a very simple black background is kept with full-sleeve attire worn by the subjects as shown in Fig. 18. Due to the simple background and similarity with the training set, the segmentation of the hand is very accurate which consequently gives us a recognition accuracy of 100% for our small dataset. So, this is a clear indication that a simple background results in good segmentation which subsequently gives better recognition performance.

5 Conclusion

In this work, a model has been presented to recognize trajectory-based isolated hand gestures where the isolated dynamic gesture is converted into a single image consisting of the contour of the gesture trajectory. This is done by combining the

Table 4 Comparison with other results for NITS database

Work	Dataset	Features	Classifier	Accuracy (%)
[47]	Bare hand gestures, English alphabets	Handcrafted features	kNN, SVM, ANN	87.82%, 88.31%, 90.58%
[43]	English alphabets	Handcrafted features	ANN, SVM	96.41%, 96.95%
Our method	Red marker, Bare hand, English alphabets	Deep network features	SVM classifier	97.76% (Bare hand), 99.09% (Red marker)

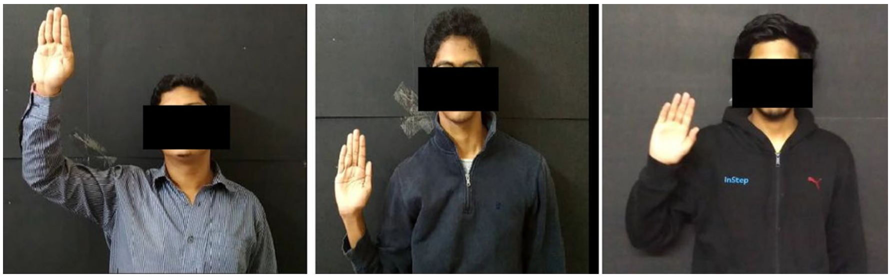


Fig. 18 Sample frames from our in-house dataset

classical preprocessing step of segmentation and tracking. These preprocessing steps are adopted in such a way to eliminate the constraints of illumination variation and occlusion in gesture videos. The output image of the image processing step is fed to a pre-trained deep network that is robust at learning shape features. Thus, the extracted deep features has been applied to a SVM classifier to test different datasets. Due to the built-in feature learning capability of deep networks, considerable good results are obtained in the results.

The major disadvantage of this model may occur in the segmentation process especially with complex background scenes. Accurate segmentation of hand or body parts from the captured images/videos still remains a challenge in computer vision for many preoccupied constraints such as illumination variation, background complexity, skin-color variation, occlusion and the articulated shape of the hand. So, the performance of the model mainly depends on the segmentation process of the hand which is basically dependent on the selection of a proper threshold for the color cues. As a future direction, some modifications may be carried out in the pre-processing step to eliminate these types of problems.

References

1. Chakraborty, B. K., Sarma, D., Bhuyan, M. K., & MacDorman, K. F. (2017). Review of constraints on vision-based gesture recognition for human-computer interaction. *IET Computer Vision, 12*(1), 3–15.
2. Sarma, D., & Bhuyan, M. K. (2021). Methods, databases and recent advancement of vision-based hand gesture recognition for hci systems: A review. *SN Computer Science, 2*(6), 1–40.
3. Sarma, D., & Bhuyan, M.K. (2018). Hand gesture recognition using deep network through trajectory-to-contour based images. In: *15th IEEE India Council International Conference (INDICON)*, pp. 1–6.
4. Harding, P.R., & Ellis, T. (2004). Recognizing hand gesture using fourier descriptors. In: Pattern Recognition, 2004. ICPR 2004. *Proceedings of the 17th International Conference On*, vol. 3, pp. 286–289 . IEEE.
5. Dardas, N., Chen, Q., Georganas, N.D., & Petriu, E.M. (2010). Hand gesture recognition using bag-of-features and multi-class support vector machine. In: *Haptic Audio-Visual Environments and Games (HAVE)*, 2010 IEEE International Symposium On, pp. 1–5 . IEEE.
6. Feng, K.-P., & Yuan, F. (2013). Static hand gesture recognition based on hog characters and support vector machines. In: *Instrumentation and Measurement, Sensor Network and Automation (IMSNA), 2013 2nd International Symposium On*, pp. 936–938 IEEE.

7. Alon, J., Athitsos, V., Yuan, Q., & Sclaroff, S. (2009). A unified framework for gesture recognition and spatiotemporal gesture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(9), 1685–1699.
8. Lee, H.-K., & Kim, J.-H. (1999). An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(10), 961–973.
9. Bhuyan, M. K., Kumar, D. A., MacDorman, K. F., & Iwahori, Y. (2014). A novel set of features for continuous hand gesture recognition. *Journal on Multimodal User Interfaces, 8*(4), 333–343.
10. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.
11. Ciregan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference On,* pp. 3642–3649 IEEE.
12. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1725–1732.
13. Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems,* pp. 1097–1105.
14. Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In: *Advances in Neural Information Processing Systems,* pp. 568–576.
15. Pigou, L., Dieleman, S., Kindermans, P.-J., & Schrauwen, B. (2014). . Sign language recognition using convolutional neural networks. In: *Workshop at the European Conference on Computer Vision,* pp. 572–578 Springer.
16. Sobottka, K., & Pitas, I. (1998). A novel method for automatic face segmentation, facial feature extraction and tracking. *Signal Processing: Image Communication, 12*(3), 263–281.
17. Tsekeridou, S., & Pitas, I. (1998) Facial feature extraction in frontal views using biometric analogies. In: *Proc. 9th European Signal Process. Conf (EUSIPCO 1998),* pp. 1–4.
18. Nikolaidis, A., & Pitas, I. (2000). Robust watermarking of facial images based on salient geometric pattern matching. *IEEE Transactions on Multimedia, 2*(3), 172–184.
19. Hsu, R.-L., Abdel-Mottaleb, M., & Jain, A. K. (2002). Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 696–706.
20. Shaik, K. B., Ganesan, P., Kalist, V., Sathish, B. S., & Jenitha, J. M. M. (2015). Comparative study of skin color detection and segmentation in HSV and YCbCr color space. *Procedia Computer Science, 57*(C), 41–48.
21. Kukharev, G., & Nowosielski, A. (2004). Fast and efficient algorithm for face detection in colour images. *Machine Graphics and Vision International Journal, 13*(4), 377–399.
22. Yin, Z., & Collins, R. (2006). Moving object localization in thermal imagery by forward-backward mhi. In: *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06), pp. 133–133*. IEEE.
23. Fukunaga, K., & Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory, 21*(1), 32–40.
24. Dondi, P., Lombardi, L., & Porta, M. (2014). Development of gesture-based human-computer interaction applications by fusion of depth and colour video streams. *IET Computer Vision, 8*(6), 568–578.
25. Chai, Y., Shin, S., Chang, K., & Kim, T. (2010). Real-time user interface using particle filter with integral histogram. *IEEE Transactions on Consumer Electronics, 56*(2), 510–515.
26. Nadgeri, S.M., Sawarkar, S., & Gawande, A.D. (2010). Hand gesture recognition using camshift algorithm. In: *Emerging Trends in Engineering and Technology (ICETET), 2010 3rd International Conference On,* pp. 37–41 . IEEE.
27. Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*(5), 564–577.
28. Guo, J.-M., Liu, Y.-F., Chang, C.-H., & Nguyen, H.-S. (2011). Improved hand tracking system. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(5), 693–701.
29. Bradsky, G. (1998). Computer vision face tracking as a component of a perceptual user interface. In: *Workshop on Applications of Computer Vision*, pp. 214–219.
30. Shi, J., & Tomasi, C. (1994). Good features to track. In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 593–600 . IEEE.

31. Kolsch, M., & Turk, M. (2004). Fast 2d hand tracking with flocks of features and multi-cue integration. In: *2004 Conference on Computer Vision and Pattern Recognition Workshop,* pp. 158–158. IEEE.
32. Asaari, M. S. M., Rosdi, B. A., & Suandi, S. A. (2015). Adaptive kalman filter incorporated eigenhand (akfie) for real-time hand tracking system. *Multimedia Tools and Applications, 74*(21), 9231–9257.
33. Wang, X., Xia, M., Cai, H., Gao, Y., & Cattani, C. (2012). Hidden-markov-models-based dynamic hand gesture recognition. *Mathematical Problems in Engineering*.
34. Chai, D., & Ngan, K. N. (1999). Face segmentation using skin-color map in videophone applications. *IEEE Transactions on Circuits and Systems for Video Technology, 9*(4), 551–564.
35. Sharma, G., & Rodríguez-Pardo, C.E. (2012). The dark side of cielab. In: *Color Imaging XVII: Displaying, Processing, Hardcopy, and Applications,* vol. 8292, p. 82920 . International Society for Optics and Photonics.
36. Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision, 57*(2), 137–154.
37. Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics, 9*(1), 62–66.
38. Kameda, Y., & Minoh, M. (1996). A human motion estimation method using 3-successive video frames. In: *International Conference on Virtual Systems and Multimedia,* pp. 135–140.
39. Gupta, S., Bhuyan, M.K., & Sasmal, P. (2020). Occlusion robust object tracking with modified particle filter framework. In: *2020 IEEE Applied Signal Processing Conference (ASPCON),* pp. 257–261. IEEE.
40. Wu, B.-F., Kao, C.-C., Jen, C.-L., Li, Y.-F., Chen, Y.-H., & Juang, J.-H. (2013). A relative-discriminative-histogram-of-oriented-gradients-based particle filter approach to vehicle occlusion handling and tracking. *IEEE Transactions on Industrial Electronics, 61*(8), 4228–4237.
41. Yu, G., Hu, Z., Lu, H., & Li, W. (2011). Robust object tracking with occlusion handle. *Neural Computing and Applications, 20*(7), 1027–1034.
42. Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). Emnist: an extension of mnist to handwritten letters. arXiv preprint arXiv:1702.05373
43. Misra, S., Singha, J., & Laskar, R. (2018). Vision-based hand gesture recognition of alphabets, numbers, arithmetic operators and ascii characters in order to develop a virtual text-entry interface system. *Neural Computing and Applications, 29*(8), 117–135.
44. Peng, Y., & Yin, H. (2017). Markov random field based convolutional neural networks for image classification. In: *International Conference on Intelligent Data Engineering and Automated Learning,* pp. 387–396 Springer.
45. Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*(2), 121–167.
46. Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks, 13*(2), 415–425.
47. Singha, J., Roy, A., & Laskar, R. H. (2018). Dynamic hand gesture recognition using vision-based approach for human-computer interaction. *Neural Computing and Applications, 29*(4), 1129–1141.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.