Innovations in Systems and Software Engineering
https://doi.org/10.1007/s11334-022-00477-z

S.I. : LOW RESOURCE MACHINE LEARNING ALGORITHMS (LR-MLA)

Two-stream fusion model using 3D-CNN and 2D-CNN via video-frames and optical flow motion templates for hand gesture recognition

Debajit Sarma[1] · V. Kavyasree[1] · M.K. Bhuyan[1]

Received: 29 December 2021 / Accepted: 16 August 2022

Abstract
Hand gestures are useful tools for many applications in the human-computer interaction community. Here, the objective is to track the movement of the hand irrespective of the shape, size and color of the hand. And, for this, a motion template guided by optical flow (OFMT) is proposed. OFMT is a compact representation of the motion information of a gesture encoded into a single image. Recently, deep networks have shown impressive improvements as compared to conventional hand-crafted feature-based techniques. Moreover, it is seen that the use of different streams with informative input data helps to increase the recognition performance. This work basically proposes a two-stream fusion model for hand gesture recognition. The two-stream network consists of two layers—a 3D convolutional neural network (C3D) that takes gesture videos as input and a 2D-CNN that takes OFMT images as input. C3D has shown its efficiency in capturing spatiotemporal information of a video, whereas OFMT helps to eliminate irrelevant gestures providing additional motion information. Though each stream can work independently, they are combined with a fusion scheme to boost the recognition results. We have shown the efficiency of the proposed two-stream network on two databases.

Keywords Hand gesture recognition · Two-stream fusion model · Optical flow-guided motion template (OFMT) · 2D-CNN · 3D-CNN

1 Introduction

In the present time, a critical exertion has been given to body/body-part movement analysis in real life. Also, gesture recognition is a significant field of research in computer vision with numerous applications. Since gestures constitute a common and natural means for non-verbal communication, hand gesture recognition from visual images constitutes a vital role of this research [1]. The utilizations of hand gesture recognition systems cover different spaces like sign language, medical assistance, virtual reality-augmented reality (VR-AR) [2]. Moreover, as the world adapts to the new changes after the COVID-19 pandemic, touch-less technology can be the 'new normal' in such situations to minimize the risk of a global health crisis. For instance, in airports, if cameras and hardware are already embedded, passengers can take benefit from hand tracking and gesture recognition to control menus without physically touching a platform. Though there are some other touch-less technologies such as voice recognition, language and pronunciation become a barrier in many instances. Moreover, people are focusing on using smartphones to minimize contact when it comes to aspects such as check-in. However, with smartphones, passengers still often have to touch a screen, which gives a chance of risk. Additionally, at airport border control, it is often forbidden to use a smartphone. So, there are further limits to these existing features. In addition, on roads, drivers can control auto-navigation through simple in-air movements. In such cases, hand-tracking and gesture recognition technology can provide a hardware-agnostic solution to these problems. Gestures can be made universal and users can apply user-friendly gestures in place of multi-step interactions for communication. With a worldwide focus on reducing the risk of spreading bacteria and viruses, this sort of solution would undoubtedly be welcomed by all.

✉ Debajit Sarma
s.debajit@iitg.ac.in

V. Kavyasree
kavyasre@alumni.iitg.ac.in

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

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

For achieving good performance, the hand gesture recognition system should be independent of different textures like shape, size and color of the hand for tracking purpose. Out of various methods, the estimation of the motion field is invariant to shape and appearance (at least in theory) and can be used directly to describe human gestures/actions [3]. Optical flow and motion-templates are the two main motion-based representation methods used for this purpose [4]. Generally, both these methods are used separately for motion estimation since both have their own advantages. Motion templates like motion-energy-image (MEI) and motion-history-image (MHI) [5] give a global aspect of motion without the requirement of segmentation of the moving object. It is computationally very efficient making it suitable for real-time applications [6]. In [7], authors applied an approach combining MHI with statistical measures and frequency domain transformation on depth images for one-shot-learning hand gesture recognition. Due to the availability of the depth information, the background-subtracted silhouette images were obtained using a simple mask threshold, whereas in [8], authors used pseudo-color-based MHI images as input to convolutional networks. On the other hand, optical flow is obtained from the movement of the target object in a video scene. Though it is computationally a little expensive, still it has the advantage that it can produce good results even in the presence of a bit of camera movement. In [9], the optical flow was used to detect the direction of motion along with the RANSAC algorithm which in turn helped to further localize the motion points. There are only a few examples like [10] where the optical flow is combined with the motion template. In [10], the authors claimed that the combined technique can give a better discrimination power to describe local motions in a global time-space representation. In this work, we have also proposed a motion template driven by optical flow. Our method is different from [10] in reducing background noises through an update rule. In our work also, better discrimination can be seen for optical flow-guided motion template (OFMT) over conventional motion templates. This combined method can accurately detect the location and thus provide the contour of the moving object just like a tracker. The effectiveness of this method is quite impressive for long and varied video sequences.

On the other hand, feature extraction is one of the most important steps for proper recognition of the action/gesture. Most hand-crafted features usually demand the user to have some prior knowledge and some pre-processing steps. Generally, the feature extraction process needs to segment the body/body-part from the background in the video sequences. For a complex and changing background environment, segmentation may be very difficult due to the variation in shape and appearance of body/body-part depending on many factors like clothing, illumination variation, image resolution, etc. In [11,12], the authors used the skin segmentation method to segment the hand portion from the background. But this method had issues when there were some skin-color like objects present in the background. So, one major objective in this work is to skip the segmentation part or to adopt some other method for this purpose. 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. Indifference to hand-crafted features, there is a growing trend toward feature representations learned by deep neural networks [8,11,13–15]. But, in deep learning techniques, the main requirement is lots of database samples. Several authors have emphasized the importance of using many diverse training examples for CNN’s [16]. For datasets with limited diversity, they have proposed data augmentation strategies to prevent CNN from overfitting. In order to avoid overfitting, Molchanov et al. [14] introduced several space-time video augmentation techniques and applied the whole hand gesture video sequences to a 3D-CNN that can extract features from both spatial and temporal dimensions by performing 3D convolutions [17].

Ciregan et al. [18] has shown that the use of multi-column deep CNNs with multiple parallel networks improves recognition rates of single networks by 30–80% for various image classification tasks. Similarly, for large-scale video classification, Karpathy et al. [19] have shown the best results on combining CNNs trained with two separate streams of the original and spatially cropped video frames. Simonyan and Zisserman [20] proposed separate CNNs for the spatial and temporal streams that are late-fused and that explicitly use optical flow in the context of action recognition. To recognize sign language gestures, Neverova et al. [21] employed CNNs to combine color and depth data from hand regions and upper-body skeletons. A two-stream model with two C3D layers that takes RGB and optical flow computed from the RGB stream as inputs was used by [13] for action recognition. [22] used a hidden two-stream CNN model which takes only raw video frames as input and directly predicts action classes without explicitly computing optical flow. Here, the network predicts the motion information from consecutive frames through a temporal stream CNN that makes the network 10x faster [22], without computing optical flow which is time-consuming. But still, two hidden layers in one stream are computationally not so efficient. Moreover, the state-of-the-art performance is achieved through traditional optical flow precomputed for the convolution layer in a two-stream network for action and hand gesture recognition [8,13,20]. But this approach of precomputation of optical flow motion vectors through CNN is computationally expensive and storage inefficient [22].

So in this paper, our main intention is to propose a resource-efficient network in terms of data and processing power as much as possible without compromising much in

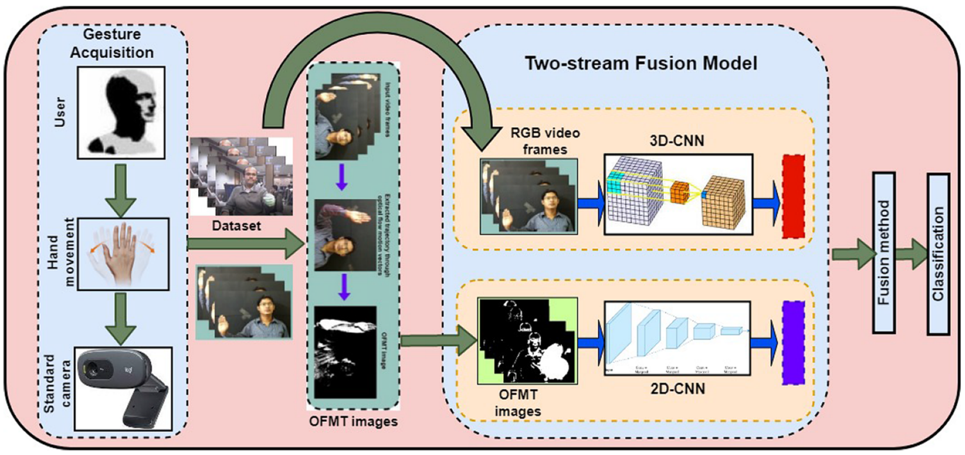

Fig. 1 Proposed framework for hand gesture recognition

its performance. The complete framework is shown in Fig. 1. Here, we propose a deep learning-based two-stream network that is zoomed in Fig. 2. The first layer/stream is a 3D-CNN (C3D) network in the two-stream architecture that captures the spatial as well as temporal information from gesture videos. The second layer is a 2D-CNN model where the input is an optical flow-guided motion template (OFMT) image. OFMT is a hybrid representation, proposed in this work that is obtained by combining optical flow with the motion template to get the advantage of both the methods for temporal evaluation analysis. OFMT is used to provide additional motion pattern information which in turn helps to eliminate irrelevant gestures. The output score of both the layers is fused using an ensemble method to boost the final output. The main contributions of our proposed model are as follows:

1. Ground truth flow is required in supervised training for optical flow estimation. But, generally, the ground truth flow is not available except for limited synthetic data [22]. Moreover, computation of optical flow and then learning the mapping from optical flow to action labels are time-consuming as well as storage demanding. So, we have proposed optical flow-guided motion template (OFMT) images as input to the 2D-CNN stream which provides additional temporal information in a resource constraint environment.
2. Our method is efficient in terms of computation and storage point of view as we do not need to store the precomputed optical flow. Moreover, the requirement of segmenting the hand portion from the body is not needed, and also, the size, shape, and color of the hand have no effect on the OFMT images.
3. A late-fusion scheme is proposed to leverage the information containing in both RGB gesture videos and motion template modalities. The advantage of the proposed method is that different deep models can provide complementary motion information. The first layer can capture the spatiotemporal information through the 3D deep network, while the motion-patterns are obtained using 2D-CNN through OFMT images.

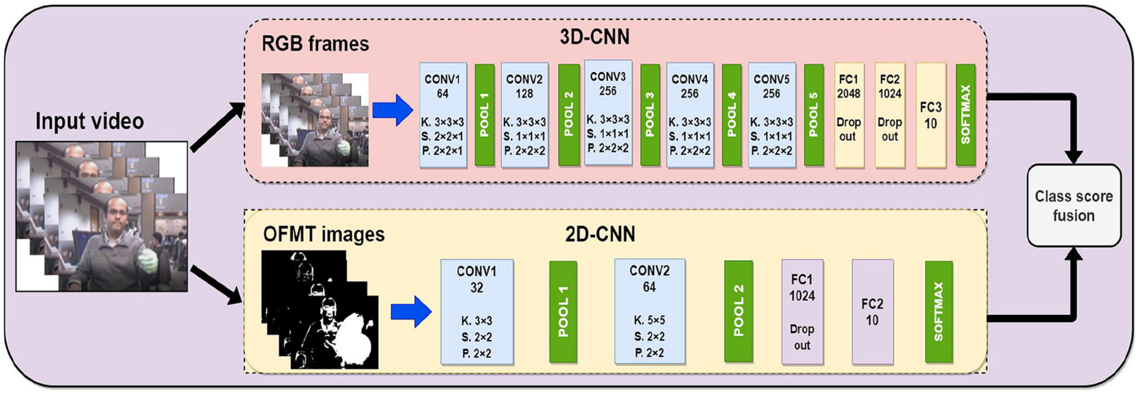
Input video
RGB frames
3D-CNN
CONV1 64 K. 3×3×3 S. 2×2×1 P. 2×2×1
POOL 1
CONV2 128 K. 3×3×3 S. 1×1×1 P. 2×2×2
POOL 2
CONV3 256 K. 3×3×3 S. 1×1×1 P. 2×2×2
POOL 3
CONV4 256 K. 3×3×3 S. 1×1×1 P. 2×2×2
POOL 4
CONV5 256 K. 3×3×3 S. 1×1×1 P. 2×2×2
POOL 5
FC1 2048 Drop out
FC2 1024 Drop out
FC3 10
SOFTMAX
OFMT images
2D-CNN
CONV1 32 K. 3×3 S. 2×2 P. 2×2
POOL 1
CONV2 64 K. 5×5 S. 2×2 P. 2×2
POOL 2
FC1 1024 Drop out
FC2 10
SOFTMAX
Class score fusion

Fig. 2 Proposed two-stream network for hand gesture recognition (K = kernel size, S = stride size, P = pooling size, max-pooling is used here)

2 The Proposed Methodology

As shown in Fig. 2, the proposed gesture model is composed of two main streams/layers—the first layer is a 3D-CNN (C3D) network in a two-stream architecture to capture spatial as well as temporal information of a gesture. The second layer is a 2D-CNN network, where the input is an optical flow-guided motion template (OFMT) image. OFMT is a hybrid, compact and robust motion representation, proposed in this work. The OFMT template is obtained by combining optical flow information with the motion template. In this way, we get the advantage of both the methods for temporal evaluation analysis. The OFMT is used to provide motion pattern information, which in turn eliminates irrelevant gestures. The proposed OFMT can nominally reduce computational complexity and memory requirement. The output of a CNN classifier is a class-membership probability for each of the gestures under consideration, and thus, the prediction results of 3D-CNN and 2D-CNN networks are fused through a simple probability-based ensemble method at the decision level to boost the final output by taking advantage of both the models. Here, we will first talk about proposed OFMT images then about the two-stream network.

2.1 Proposed Optical Flow-guided Motion Templates (OFMT):

The input to the 2D-CNN is a compact motion template. For this, a hybrid representation is proposed for encoding temporal information of a gesture by combining optical flow motion information with motion templates. This representation takes advantage of both optical flow and motion-energy-image (MEI) and motion-history-image (MHI) templates.

MEI represents where motion has occurred in an image sequence; whereas MHI represents how an object is moving [6]. MEI describes the motion-shape and spatial distribution of motion, and MHI is the function of the intensity of motion of each pixel at that location. MEI-MHI can be implemented by the following algorithm.
MEI-MHI Algorithm [5]:

- **Image sequences**

$$I(x, y, t) = (I_1, I_2, ..., I_n). \tag{1}$$

- **Image binarization**

$$B(x, y, t) = |I(x, y, t) - I(x, y, t-1)|. \tag{2}$$

$$\text{where } B(x, y, t) = \begin{cases} 1 & \text{if } B(x, y, t) > \xi \\ 0 & \text{otherwise} \end{cases}$$

- **MEI**

$$E_\tau(x, y, t) = \bigcup_{i=0}^{\tau-1} B(x, y, t-i). \tag{3}$$

- **MHI**

$$H_\tau(x, y, t) = \begin{cases} \tau & \text{if } B(x, y, t) = 1 \\ max(0, H_\tau(x, y, t-1) - \delta & \text{otherwise} \end{cases}$$

where τ decides the temporal extent of the motion (in terms of frames) and δ is the decay parameter (Figs. 3, 4).

Optical flow indicates the change in image velocity of a point moving in the scene, also called a motion field. Here, the goal is to estimate the motion field (velocity vector) which can be computed from horizontal and vertical flow fields. Here, optical flow is used to track the movement of the hand irrespective of shape, size and color. One major problem with optical flow estimation is that it is very sensitive to noise and outlier due to background motion. And, to get rid of this problem of noise and outlier, Gaussian smoothing operation is done to image frame $k(x, y, t)$, where (x, y) denotes the location of the pixel and t denotes time. Smoothing is done by convolving each frame with a Gaussian kernel $G_\sigma(x, y)$ of standard deviation σ:

$$I(x, y, t) = (G_\sigma * k)(x, y, t), \tag{4}$$

The low-pass effect of Gaussian convolution removes noise and other destabilizing high-frequency outliers. In a subsequent procedure, σ also called the 'noise sale,' can be chosen of different values. While some moderate pre-smoothing improves the results, great care should be taken not to apply too much pre-smoothing, since this would severely destroy important image structure.

Another problem is tracking points that are moving long distances with a higher speed of motion. This can be mitigated by a course-to-fine optical flow estimation by forming an image pyramid. All these steps are shown in Fig. 5.

For tracking the motion of the hand, Lukas-Kanade Algorithm (LKA) [23] is applied by extracting the desired features of the gesture. For that purpose, a mouse call back function is adopted to select the desired feature on the first frame that will be tracked successively in the next frames of the video. The optical flow motion vectors track these features during the trajectory of the dynamic gesture. These vectors are tracked on a per-frame basis. By joining these flow vectors on the mask, the trajectory of the gesture is recovered from the continuous motion of the hand.
Optical Flow Tracking Algorithm:

- The whole video is converted into gray images.
- A customized black window mask is created with the same dimensions as that of the original frame of the video.

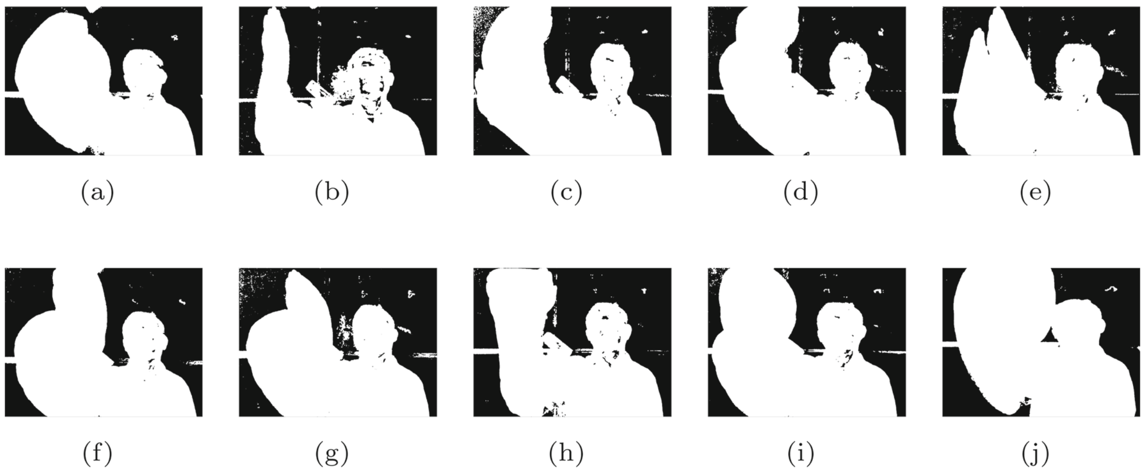

Fig. 3 MEI images: **a**–**j** representing gestures 0–9

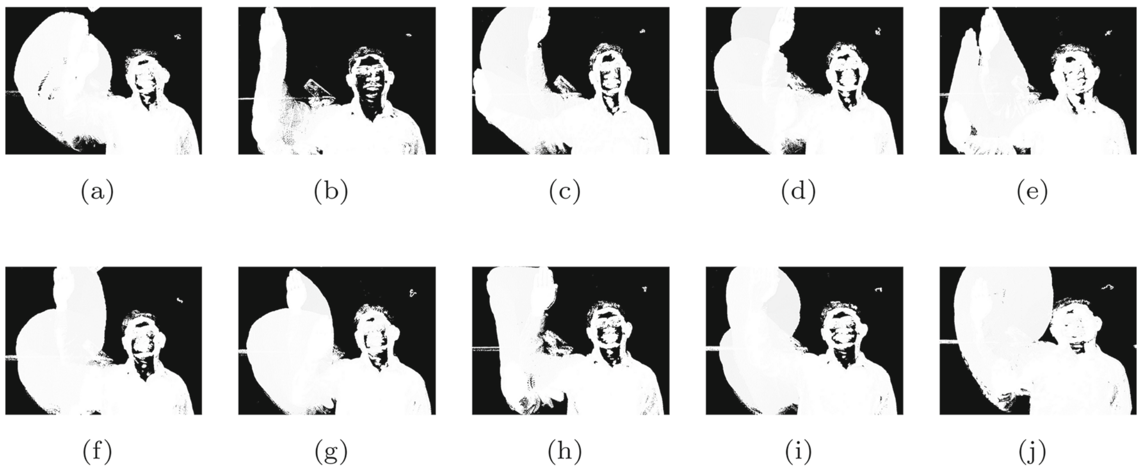

Fig. 4 MHI images: **a**–**j** representing gestures 0–9

- The first frame of the video is read and it is called $frame_{old}$.
- The initial points are selected on the $frame_{old}$ using a mouse call back function (mouse cursor). These points are named $points_{old}$.
- Initialize a new list to store optical flow vectors.
- Initialize Lucas-Kanade parameters with the window size as of 15×15 and pyramid level as 4. Large motion is ignored by the pyramidal model.
- while for the next number of frames:
  $frame_{new}$ = Next frame of the video converted into a gray scale image.
  $points_{new}$ = Generated by optical flow with $frame_{old}$, $frame_{new}$, $points_{old}$ and LK parameters
- copy $frame_{new}$ into $frame_{old}$.
  copy $points_{new}$ into $points_{old}$.
  Append $points_{new}$ to list.
- if length(list) ≥ 2:
  Draw line with the last two points in the list on the mask.
- Thus, the optical flow vector between two frames is obtained.

Now, here, we present a motion template driven by the optical flow method. This combined method can accurately detect the location of the hand and also provide the contour of the moving hand just like an object tracker. MEI and MHI are generated using a binarized image, obtained from frame subtraction, using a threshold ξ as shown in Eq. (2). In the motion template representation, all the foreground or moving pixels (i.e., $B(x, y, t) = 1$) are considered for creating the templates irrespective of the duration and speed of the individual moving pixel. Motion templates basically describe the global motion of a scene and cannot fully describe the local motions of the target object, whereas optical flow is generally used for foreground segmentation or to extract moving objects. It can better describe the local motions of the target object. In our method, optical flow is judiciously combined with the motion templates to exploit the advantages of both

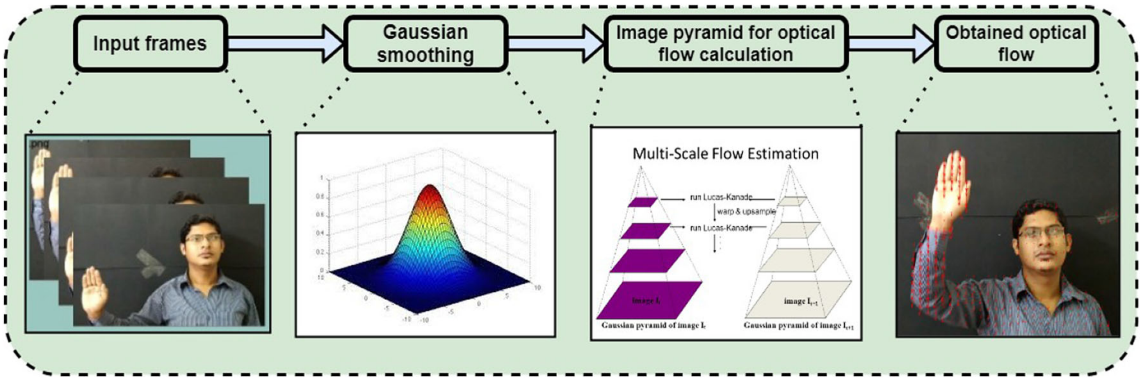
Input frames
Gaussian smoothing
Image pyramid for optical flow calculation
Obtained optical flow
Multi-Scale Flow Estimation
run Lucas-Kanade
warp & upsample
run Lucas-Kanade
Gaussian pyramid of image I
Gaussian pyramid of image I

Fig. 5 Steps to obtain optical flow from input video frames

methods. In the proposed method, the optical flow sequence $O(x, y, t)$ representing the moving regions of the previously smoothed image is accumulated and fused together to form an optical flow-guided motion template as per the following equation:

$$E_\tau(x, y, t) = \bigcup_{i=0}^{\tau} O(x, y, t - i - 1) + \lambda . O(x, y, t) \quad (5)$$

where τ indicates the duration of the gesture, and λ is an update parameter. If the optical flow length $O(x, y, t)$ is small compared to a pre-defined threshold ϵ_s, then it labels the pixel (x, y) as a background point, and hence, λ value is taken as zero to reduce the effect of background noises. If the optical flow length value is greater than the threshold, then it labels a pixel as a foreground moving point, then λ is empirically set to 5 to consider foreground pixels. In this way, the background noise is reduced in our proposed method. The saved moving points from the video frames generate a single image providing the trajectory of the gesture as shown in Fig. 6. These steps are performed for all the hand gesture videos and the corresponding OFMT images are pre-obtained as shown in Fig. 7. The entire experiment is done in python environment, with an OpenCV tool and 30 frames/s is the frame rate used for generating the OFMT images that are fed to the 2D-CNN network. Another advantage obtained here is that the requirement of segmenting the hand portion from the body is not needed and also, the size, shape, and color of the hand have no effect on the OFMT images. Through the naked eye, it can be easily noticed that our proposed OFMT images give much better results than the conventional motion templates. Still, for quantitative analysis, we calculate entropy and structural similarity index measurement (SSIM) for each set of images to get a clear idea.

2.1.1 Entropy

The entropy of a discrete random variable X with possible values $\{x_0, x_1, x_2, ..., x_N\}$ can be defined as [24]

$$H = -\sum_{k=0}^{N} p_k log_2(p_k) \quad (6)$$

where H indicates entropy and p_k is the probability associated with input k.

For a grayscale image, the intensity value of each pixel varies from 0 to 255, and the possibility of a particular value occurring is random and varies with the pixel intensity values of the images. Considering an image with dimension $M \times N$ having a total of $W = M \times N$ pixels, the probability of a particular intensity value x_k occurring in the image is $p(x_k) = n_k / W_k$, where n_k is the number of occurrences of x_k among the W pixels. In this case, considering $\sum_k n_k = M$, the entropy of the image can be expressed as

$$H = -\frac{1}{W} \sum_{k=0}^{255} n_k log_2(n_k) \quad (7)$$

Table 1 shows that most OFMT images have low entropy values compared to MHI or MEI images. This is due to the fact that, from the image compression algorithm viewpoint, entropy tries to discover predictability and each aspect of predictability requires some storage to represent. The more storage that an image requires to represent its predictability, the higher is the entropy the image posses. An image that is all the same is entirely predictable and has low entropy. An image that changes from pixel to pixel might at first thought to be unpredictable, but the change might follow a pattern too. From this point, we can say that if our motion template can give some idea about its gesture i.e., predictable in nature, it should contain low entropy value.

2.1.2 Structural Similarity Index Measurement (SSIM)

There are many algorithms developed to provide an index for image quality analysis. SSIM is a reference image quality indexing algorithm which is why it requires a reference image to estimate the quality of the test image. The param-

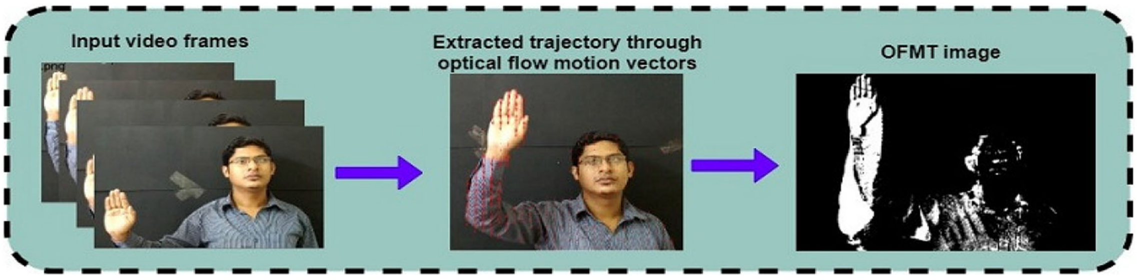

Fig. 6 Steps to obtain OFMT images from input video frames

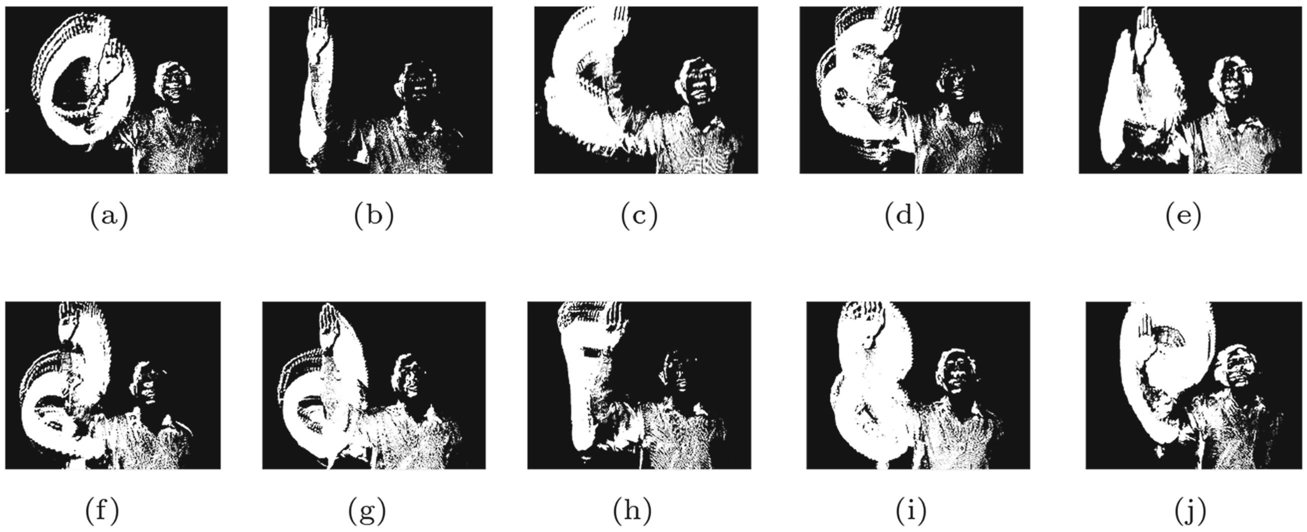

Fig. 7 Optical flow-guided motion template (OFMT) images applied on our in-house dataset: **a**–**j** for gesture 0–9

eters that are considered for comparison are the luminance, contrast and structure of the images [25]. These three factors are estimated from the images and a relative score is being provided to the test image. The factors mentioned are some of the important factors used by the human eye to provide a subjective analysis of the images. These physical factors are simulated with the use of the basic statistical parameters like mean, variance and covariance.

The SSIM indexing algorithm for image quality assessment uses a sliding window approach. The window moves pixel-by-pixel across the whole image space. At each step, the SSIM index is calculated within the local window. If one of the images being compared is considered to have a perfect quality, then the resulting SSIM index map can be viewed as the quality map of the other (distorted) image. The distribution of the window can be rectangular or Gaussian distribution. The Gaussian distribution is preferred to avoid the blocking effect which is predominant in a rectangular window. Here, we consider Gaussian distribution with parameters $\mu = 0$ and $\sigma = 1.5$. Finally, a mean SSIM index (mSSIM) of the quality-map is used to evaluate the overall image quality. Here, we have compared normal motion templates with our OFMT images (Refer Table 1). Generally, the zero value of SSIM indicates the worst case and one the best-case scenario. Visually, an SSIM index greater than 0.94 can be considered a good image in comparison with the original image.

2.2 Spatiotemporal feature learning through a 3D convolutional (C3D) model

The original C3D [17] was designed for RGB videos. The number of parameters of the networks depends on the resolution of input frames. The original C3D was trained on the large-scale dataset Sport1M [19], which consists of 1.1M videos downloaded from YouTube consisting of 487 sports classes. 2D-CNN is extended to a 3D-CNN by incorporating the temporal dimension of a video sequence. In 2D-CNNs, the dimension of each feature map is $c \times h \times w$, where c represents the number of filters in the convolutional (conv) layer, h and w represent the height and width of the feature map. In 3D-CNNs, the dimension of each feature map is $c \times l \times h \times w$, where additional parameter l represents the number of frames. This network extracts the features which are compact and generic while being discriminative. As we worked on two smaller databases, a slightly different

Table 1 Image entropy and mSSIM values for different motion templates

→	0	1	2	3	4	5	6	7	8	9
MEI	1.0128	1.0358	1.0217	1.0119	1.0215	1.0090	1.0300	1.0237	0.9801	0.9227
MHI	2.4577	1.9533	2.5930	2.6417	2.4296	2.4893	2.4033	2.3524	2.6540	2.7437
OF-MT	0.9258	0.6670	1.0636	0.9568	1.0130	0.9902	1.0339	0.9235	1.0811	1.0351
mSSIM	0.8348	0.8509	0.830	0.8532	0.8821	0.8810	0.8890	0.8627	0.9102	0.8642

architecture with five conv layers is employed which has a smaller number of parameters compared to the original C3D [17] with eight conv layers. The proposed network has five space-time conv layers with 64, 128, 256, 256, 256 kernels. Each conv layer is followed by a rectified linear unit (ReLU) and a space-time max-pooling layer. All 3D convolution kernels are of size $3 \times 3 \times 3$, that gives the best performance [17] with stride $1 \times 1 \times 1$. Max pooling kernels are of size $2 \times 2 \times 2$ except for the first, where it is $2 \times 2 \times 1$ and stride is $2 \times 2 \times 1$. The conv layers are followed by two dense layers with 2048 and 1024 neurons and ReLU as the activation function. To avoid over-fitting while learning, there is a dropout in each dense layers. The parameter of dropout is set to 0.4, which means the layer randomly excludes 40% of neurons. The final dense layer of the classifier has 10 neurons giving us the respective class labels where softmax function is used for activation.

2.3 2D motion template CNN model

As illustrated in Fig. 2, the proposed 2D motion template CNN model consists of two major parts—motion templates and a 2D-CNN model. The generation of the motion templates is explained in the next section. The 2D-CNN architecture used in our method is a simple structure based on LeNet [26] (shown in Fig. 2). The network has two conv layers with 32 and 64 kernels followed by two fully connected layers of size 1024 and 10. The final dense layer of the classifier has 10 neurons giving us the respective class labels. The size of the kernels is 3×3 and 5×5, respectively, for the two conv layers. Each conv layer is followed by ReLU and 2×2 box non-overlapping max-pooling layers with stride 2 in both horizontal and vertical directions. A dropout of 40% is used in the dense layer with 1024 neurons to avoid over-fitting.

2.4 Fusion Rules

In [20], the authors used two decision level fusion methods of averaging and SVM fusion, on two identical C3D networks. In the averaging method, two softmax prediction scores are averaged to represent the output class scores. In the SVM fusion method, the features from fully-connected layers of both the streams are stacked and, after L2 normalization, are input to an SVM classifier, whereas in our two-stream model, we have used two non-identical networks applied on inputs with different dimensions. Hence, decision level fusion is preferred here in place of feature-level fusion due to computational overhead. But, in place of just simple averaging fusion, we have adopted an empirical formula given by Eq. (5) for output prediction score fusion.

$$p_i = \gamma . p_i^{3D} + (1-\gamma) . p_i^{2D} \quad where \;\; i = 1, 2, ..., N \tag{8}$$

Here, p_i^{3D} and p_i^{2D} are the prediction class scores of 3D-CNN and 2D-CNN, respectively, N is the number of gesture classes, and γ is an empirical parameter. For fusion at the decision level, experiment with different values of γ such as 0.2, 0.3, 0.4, 0.5, 0.6, 0.7 and 0.8 is carried out. We investigated that γ as 0.6 achieves the best performance. This is quite justified since the score given by 3D-CNN has a greater impact than the score given by 2D-CNN. The final prediction class score S is the one whose value is maximum and it is calculated as given below:

$$S = \underset{1 \le i \le N}{\operatorname{argmax}} \, p_i \quad where \;\; i = 1, 2, ..., N \tag{9}$$

3 Experimentation and Results

To evaluate the performance of the proposed method, we have carried out experiments on two databases: 1) Palm's Graffiti Digits [27] and 2) Our in-house database [11]. The following sections elaborate on all the details during the implementation and evaluation processes.

3.1 Databases

In our work, two datasets are employed, one is Palm's Graffiti Digits [27] and another is the self-collected in-house dataset [11]. The details are described below.

1. **Palm's Graffiti Digits**: The Palm's Graffiti digits database [27] contains standard RGB 2D videos of ten subjects writing "in the air" the ten Hindu-Western Arabic numerals, 0-9, in a continuous streaming mode with video size 320×240, 30 frames/s as shown in Fig. 8. This database

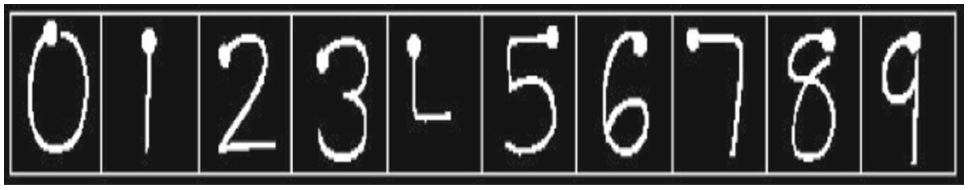

Fig. 8 Palm's Graffiti digits [27]. The dot point indicates the starting position

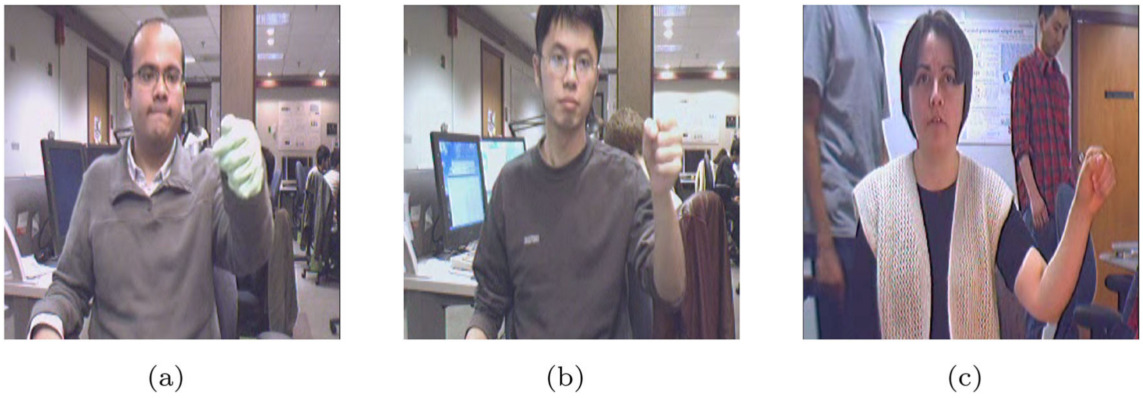

Fig. 9 Different scenes of Palm's Graffiti digits dataset [27]: **a** Green glove in GreenDigits, **b** Bare hand in EasyDigits, **c** With moving persons in background

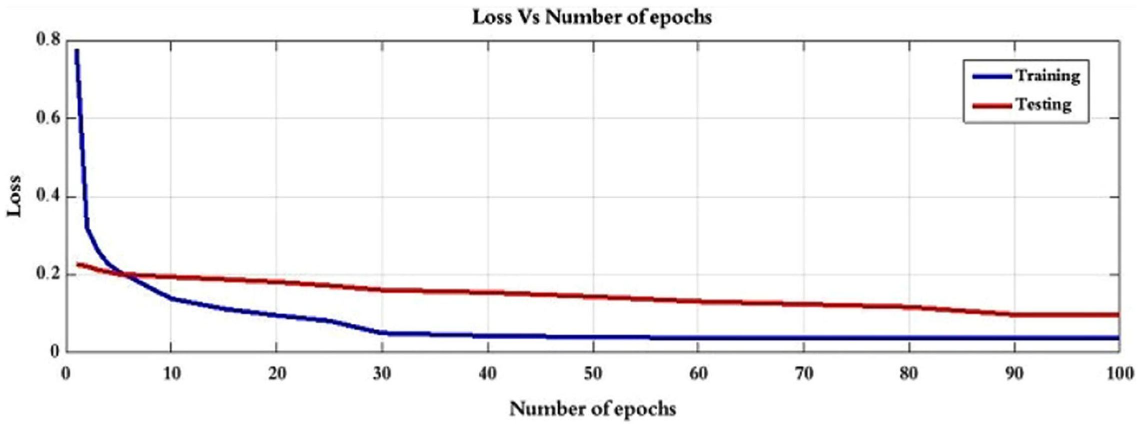

Fig. 10 Training and testing loss as a function of the number of epochs for 3D-CNN

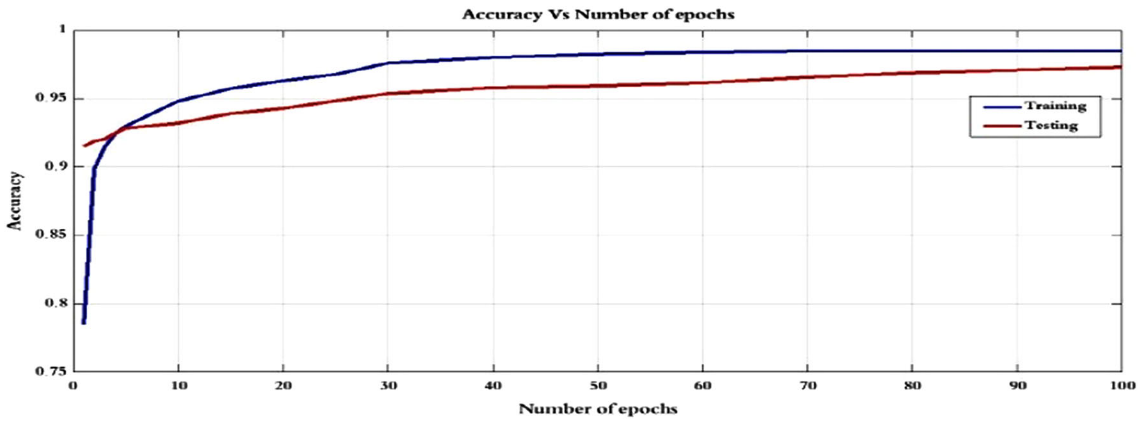

Fig. 11 Training and testing accuracy as a function of the number of epochs for 3D-CNN

is split into three subsets, namely ”GreenDigits,” ”EasyDigits,” and ”HardDigits” sets. Each one of the first two datasets contains 300 gestures (ten subjects × ten digits × three examples/digit/subject). In both datasets, the subjects used tightly folded palms while performing the gestures. GreenDigits dataset after data augmentation is used for the training phase and EasyDigits and HardDigits sets are used for the testing phase. In the GreenDigits set, subjects wear a green glove, while short-sleeves in EasyDigits. In this dataset, the subjects used tightly folded palms while performing the gestures. GreenDigits dataset after data augmentation is used for the training phase and EasyDigits and HardDigits sets are used for the testing phase. In the GreenDigits set, subjects wear a green glove, while short-sleeves in EasyDigits. HardDigits is intended for the evaluation of hand detection methods under very challenging and uncontrolled scenarios like with movement of people in the background (shown in Fig. 9). It has 140 videos (seven subjects × ten digits × two examples/digit/subject). The proposed OFMT images are obtained for these gesture videos and are fed into the 2D-CNN network for classification purposes in our experiments. While obtaining the OFMT images of the gestures, whether the subject is wearing any glove or using a bare hand, there is no effect in detecting and tracking the hand. This indicates the robustness of our method.

2. **In-house dataset**: One limited in-house dataset [11] has been created with the help of three subjects. Here, also, the gestures are ten Hindu-Western Arabic numerals, 0–9, but in an isolated mode (320 × 240, 30 frames/s). Dataset consists of 90 videos with 10 classes (three subjects × ten digits × three examples/digit/subject). For simplicity, a very simple black background is used with full-sleeve attire worn by the subjects and they used open palm while performing the gestures.

3.2 Data Augmentation

Data augmentation plays a vital role in the deep learning approach due to the huge amount of data required in these techniques. It generates more data from a small database using some simple methods like affine transformations. With this, we can increase the diversity of data available for the training model, without actually collecting new data. The generation of new data provides robustness as well as scale, translation and rotation invariance to the system. In this work, data augmentation methods are used on the GreenDigits dataset. Initially, 300 videos from the GreenDigits dataset are preprocessed into 300 OFMT images. These images are then increased to 1500 images after data augmentation. For data augmentation, several transforms are used like rotation up to 20 degrees, width shift (up to 0.2 range), height shift (up to 0.2 range), sheer (up to 0.2 range), zoom mode (up to 0.2), fill mode on nearest data, etc. Data augmentation techniques like horizontal and vertical flipping are not used on the images as it may lead to confusion between a few pairs of digits like (2, 5), (4, 7), (6, 9), etc. EasyDigits and HardDigits sets are used for the testing phase on which data augmentation is not implemented.

3.3 Experimental setup

This section gives an idea of the experimental setup, work performed and the analysis done on the databases to obtain the results. Also, it throws light on the importance of data augmentation process for small databases. The experimentation part is done taking the help of the Google Colab GPU, especially, for the training phase. Other parts of the experiment are performed in a workstation with Intel® Core™i5-4570 CPU at 3.2 GHz and 8GB in RAM without any GPU usage.

For training the 3D-CNN model, the segmented video clips of isolated gestures from GreenDigits sets are used from the Graffiti dataset. Here, stochastic gradient descent (SGD) algorithm is used with the cross-entropy loss function given by Eq. (10).

$$Loss(y, \hat{y}) = -\sum_{j=1}^{M}\sum_{i=1}^{N} y_{ij} log \hat{y}_{ij} \tag{10}$$

where M is the number of samples, N is the number of classes and $\hat{y}$ is the predicted value for a true value y. The batch size is set to 10 videos and the model is trained with 100 epochs on the training dataset. 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. Figures 10 and 11 give the training-testing loss and accuracy curves for the Graffiti dataset. From the training loss and accuracy curves, it can be concluded that the system is not suffering from over-fitting after some tuning of the hyper-parameters. Since the gestures of our limited in-house dataset are the same as the training dataset, our in-house dataset is used only for testing purpose which reduces the burden of training requirements again and again.

With regard to the training of the 2D-CNN model, the GreenDigits set from Graffiti Digits is used. SGD algorithm is carried out with a cross-entropy loss function in the training process. The initial learning rate is set to 0.01 if the epoch count is less than 25 and is reduced to 0.001 for epoch count up to 50 with batch size as 32. The training process is stopped after 50 epochs. Figures 12 and 13 give the training-testing loss and accuracy curves for the Graffiti dataset.

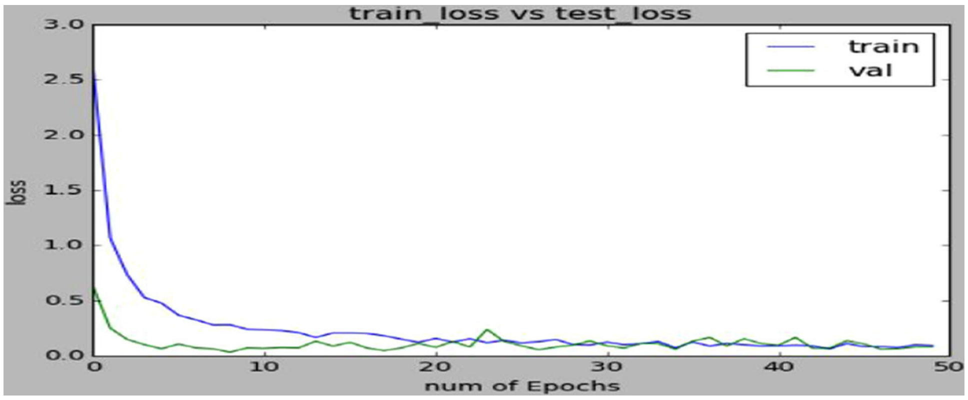

Fig. 12 Training and testing loss as a function of the number of epochs for 2D-CNN

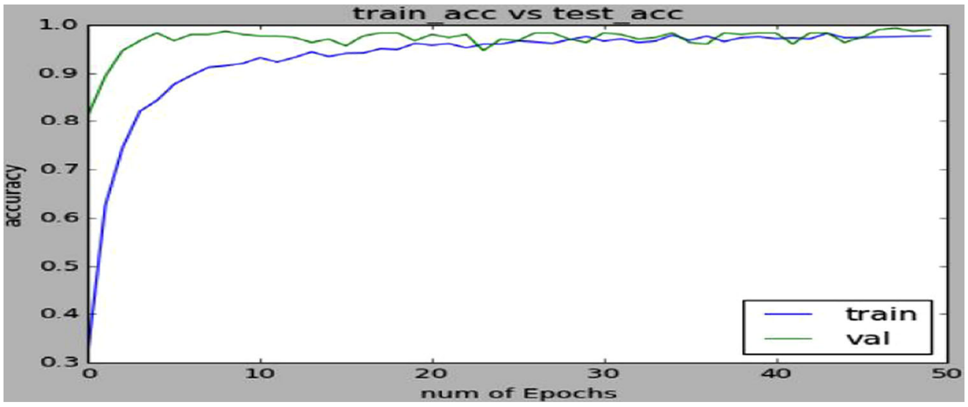

Fig. 13 Training and testing accuracy as a function of the number of epochs for 2D-CNN

Table 2 Performance accuracy (%) of 2D-CNN motion template network alone

Dataset	Without data augmentation (%)	With data augmentation (%)
Graffiti	86.24	92.60
In-house	81.20	89.70

3.4 Results

In this subsection, the performance of the proposed two-stream network is evaluated in three aspects on Graffiti as well as in-house datasets: 2D-CNN model alone with OFMT motion templates as input, 3D-CNN model alone with RGB gesture videos as input, and the combined fusion model. Basically, here, accuracy, which indicates the proportion of correctly classified samples with respect to the total number of samples, is used as the evaluation index. Since data imbalance is not there in our databases, i.e., each class is constituted by an equal number of samples so other performance matrices like precision, sensitivity, or f-score are not considered here.

In the proposed model, used 3D-CNN and the 2D motion template CNN architecture can be regarded as two heterogeneous networks that can be used as independent models. So, first, we have evaluated the performance of the independent streams and then the fusion performance as a combined network. In the case of 2D-CNN, confusion occurs mainly in discriminating class ‘3’ with class ‘5’ and class ‘1’ with class ‘7’ due to the similarities in their shapes in the OFMT images which can be seen in Fig. 7. Here, 3.4% of ‘5’ are misclassified as ‘3’ or vice versa and 2.8% of ‘7’ are misclassified as ‘1’ or vice versa and rest 2.2% are various misclassification, whereas 3D-CNN has performed quite better in this regard since it also considers the temporal evaluation of the gestures in the video clips. Table 2 gives the results of the proposed 2D motion template CNN model, where two cases, without data augmentation and with data augmentation, are considered. From Table 2, we can conclude that the data augmentation has a great impact on accuracy and can improve the network performance up to a great extent.

To analyze the performance of the 3D-CNN model, stratified 10-fold cross-validation is carried out on combined GreenDigits and EasyDigits sets of Graffiti dataset. The mean accuracy for this dataset is obtained at 97.30%, whereas our in-house dataset has provided an accuracy of 98.67% when tested on the pre-trained network. Lastly, the prediction class/label scores from both the streams are fused for each class. Since both the streams acquire complementary motion information regarding the gesture, so such fusion

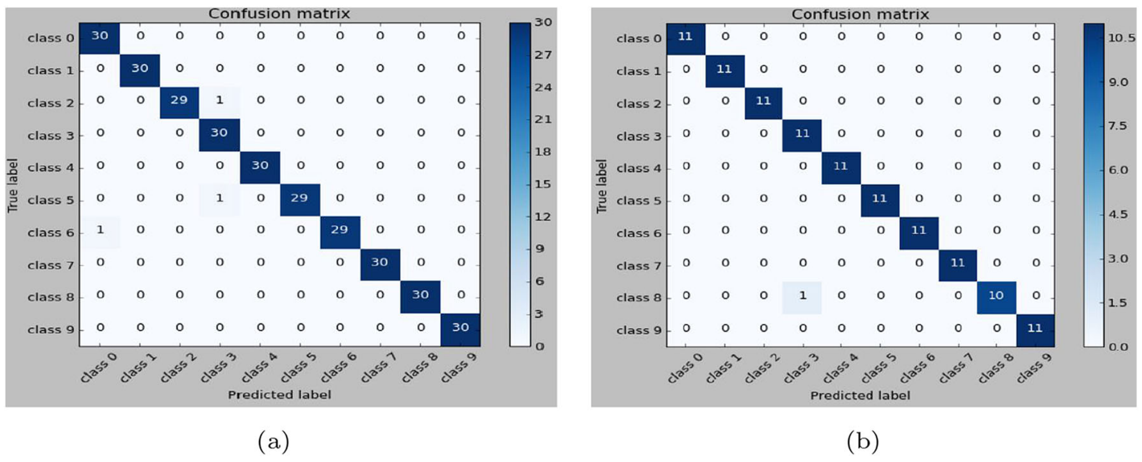

Fig. 14 Confusion matrix for: **a** EasyDigits set, **b** HardDigits set

generally boosts the recognition performance. That is, the streams complement each-other in acquiring the spatiotemporal information from their respective inputs, and thus, certain good output score w.r.t. the target can be achieved, at least by one or the other or by both. In our case also, the same scenario is noticed when both streams are fused at the decision level. Here, decision level fusion is chosen since we have used two non-identical networks applied on inputs with different dimensions. Moreover, rather than simple averaging of the prediction class scores, we have formulated a probabilistic ensemble formula given by Eq. (8) with γ as fusion parameter. Different values of γ have been tried out and γ as 0.6 has provided us the best fusion accuracy of 99.20%. So, here, more weight is given to the score provided by the 3D-CNN since it can capture more subtle spatiotemporal features compared to the other one. This is quite justified from the performance achieved by the individual networks performed on the two databases. Our in-house dataset has achieved a recognition rate of 99% when tested on the fused model. The confusion matrix for EasyDigits and HardDigits sets is also shown in Fig. 14.

3.5 Comparison with state-of-the-art

Our proposed model is compared with three existing methods performed on the same Graffiti dataset. Table 3 represents a comparison of performance for the different methods. The first two methods [28,29] rely on hand-crafted feature representations for gesture classification, while [30] uses CNN to extract gesture features. In [28], the most probable longest common subsequence (MPLCS) is proposed to measure the similarity between the probabilistic template and hand gesture sample. The final decision is based on the probability and length of the extracted subsequences. The method is also compared with HMM and CRF classifiers for performance analysis, whereas maximum cosine similarity and fastNN are used as a trajectory mapping scheme for digit hand gesture recognition in [29]. A combined fusion-based method with CNN as trajectory shape recognition and CRF as temporal feature recognition is proposed by [30]. From Table 3, it can be noticed that our 2D-CNN model is as efficient as the classic HMM model, whereas 3D-CNN has achieved even better results. In [30], fusion-based model with CNN and CRF as components has achieved 98.4% accuracy which is similar to the sequential state-space MPLCS [28] method. On the other hand, our 2D-CNN- and 3D-CNN-based fusion model has achieved state-of-the-art results with 99.20% of accuracy. The fusion result at the decision level has outperformed all other methods, which shows the effectiveness of the fusion scheme. The only work done on our in-house dataset is [11], which has a similar accuracy of 99% as this work for alphabet gesture recognition.

4 Conclusion

In this work, we propose a two-stream network with 3D-CNN and 2D-CNN as its two streams/layers for hand gesture and in general action recognition. So, the main objective of the model is to detect and recognize isolated dynamic hand gestures with varying shape, size, and color of the hand. This is possible because of the fact that the system doesn't require the pre-segmentation of the hand portion through various methods like skin-segmentation, etc. The first stream of the system is a 3D-CNN applied for capturing the spatiotemporal information directly from the RGB gesture videos. The second layer is a 2D-CNN model employed to extract motion-patterns for gesture classification. For this stream, an optical flow-guided motion template (OFMT) is used as input where the temporal motion information of a gesture is encoded into a single image which helps to remove irrelevant gesture patterns. Moreover, the proposed OFMT can nominally reduce computational complexity and memory requirement as compared to more complex networks like double 3D-

Table 3 Comparison with other methods for pre-segmented Graffiti database

Paper	Feature-type	Features	Classifier	Accuracy
[28]	Hand-crafted	Longest common subsequence (LCS)	HMM, CRF, Most probable LCS (MPLCS)	89.50%, 96.40%, 98.30%
[29]	Hand-crafted	Trajectory matching	Max cosine similarity, fastNN	97.60%
[30]	Both hand-crafted and deep features	CRF-based temporal features	CNN and CRF combined	98.40%
Our method	Deep features	Deep network, motion template	Only 2D-CNN, only 3D-CNN, Late fusion	92.60%, 97.30%, 99.20%

CNN/RNN/LSTM models. So, our proposed model can be used in a resource constraint environment without affecting much to its performance. For improving results, the prediction scores of the 3D-CNN model and the 2D-CNN model are fused. Since both the streams acquire complementary motion information regarding the gesture, so such fusion generally boosts the recognition performance. Though our model is simple, experimental results have demonstrated that it is able to achieve the state-of-the-art results. However, the adopted motion template has the limitation that the moving body has to be in a plane perpendicular to the camera. In the future, we will investigate more on robust feature learning methods for distinguishing the subtle differences among some gesture classes for the viewpoint-invariant gesture recognition method.

Declarations

Conflict of interest The authors declare that they have no conflict of interest. No funding was received for this work. In this research work, there is no involvement of human participants and/or animals in any part of the experimentation. The databases used in the experiments are publicly available.

References

1. Karam M (2006) Ph.D. thesis: a framework for research and design of gesture-based human-computer interactions. Ph.D. thesis, University of Southampton
2. Chakraborty BK, Sarma D, Bhuyan MK, MacDorman KF (2018) Review of constraints on vision-based gesture recognition for human–computer interaction. IET Comput Vis 12(1):3–15
3. Sarma D, Bhuyan M (2021) Methods, databases and recent advancement of vision-based hand gesture recognition for HCI systems: a review. SN Comput Sci 2(6):1–40
4. Sarma D, Bhuyan M (2020) Optical flow guided motion template for hand gesture recognition. In: 2020 IEEE applied signal processing conference (ASPCON), pp 262–266. IEEE
5. Bobick AF, Davis JW (2001) The recognition of human movement using temporal templates. IEEE Trans Pattern Anal Mach Intell 23(3):257–267
6. Ahad MAR, Tan JK, Kim H, Ishikawa S (2012) Motion history image: its variants and applications. Mach Vis Appl 23(2):255–281
7. Mahbub U, Imtiaz H, Roy T, Rahman MS, Ahad MAR (2013) A template matching approach of one-shot-learning gesture recognition. Pattern Recognit Lett 34(15):1780–1788
8. Zhang E, Xue B, Cao F, Duan J, Lin G, Lei Y (2019) Fusion of 2d CNN and 3D densenet for dynamic gesture recognition. Electronics 8(12):1511
9. Mahbub U, Imtiaz H, Ahad MAR (2011) An optical flow based approach for action recognition. In: 14th International conference on computer and information technology (ICCIT 2011), pp 646–651. IEEE
10. Xu H, Li L, Fang M, Zhang F (2018) Movement human actions recognition based on machine learning. Int J Online Biomed Eng (iJOE) 14(04):193–210

11. Sarma D, Bhuyan MK (2018) Hand gesture recognition using deep network through trajectory-to-contour based images. In: 15th IEEE India council international conference (INDICON), pp 1–6
12. Sarma D, Bhuyan M (2022) Hand detection by two-level segmentation with double-tracking and gesture recognition using deep-features. Sens Imaging 23(1):1–29
13. Khong V-M, Tran T-H (2018) Improving human action recognition with two-stream 3D convolutional neural network. In: 2018 1st International conference on multimedia analysis and pattern recognition (MAPR), pp 1–6. IEEE
14. Molchanov P, Gupta S, Kim K, Kautz J (2015) Hand gesture recognition with 3D convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 1–7
15. Kavyasree V, Sarma D, Gupta P, Bhuyan M (2020) Deep network-based hand gesture recognition using optical flow guided trajectory images. In: 2020 IEEE applied signal processing conference (ASPCON), pp 252–256. IEEE
16. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
17. Tran D, Bourdev L, Fergus R, Torresani L, Paluri M (2015) Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE international conference on computer vision, pp 4489–4497
18. Ciregan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR), pp 3642–3649. IEEE
19. 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
20. Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. In: Advances in neural information processing systems, pp 568–576
21. Neverova N, Wolf C, Taylor G, Nebout F (2015) Moddrop: adaptive multi-modal gesture recognition. IEEE Trans Pattern Anal Mach Intell 38(8):1692–1706
22. Zhu Y, Lan Z, Newsam S, Hauptmann A (2018) Hidden two-stream convolutional networks for action recognition. In: Asian conference on computer vision, pp 363–378. Springer
23. Lucas BD, Kanade T et al (1981) An iterative image registration technique with an application to stereo vision
24. Fan X, Tjahjadi T (2017) A dynamic framework based on local zernike moment and motion history image for facial expression recognition. Pattern Recognit 64:399–406
25. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
26. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
27. Alon J, Athitsos V, Yuan Q, Sclaroff S (2009) A unified framework for gesture recognition and spatiotemporal gesture segmentation. IEEE Trans Pattern Anal Mach Intell 31(9):1685–1699
28. Frolova D, Stern H, Berman S (2013) Most probable longest common subsequence for recognition of gesture character input. IEEE Trans Cybern 43(3):871–880
29. Poularakis S, Katsavounidis I (2015) Low-complexity hand gesture recognition system for continuous streams of digits and letters. IEEE Trans Cybern 46(9):2094–2108
30. Yang C, Han DK, Ko H (2017) Continuous hand gesture recognition based on trajectory shape information. Pattern Recognit Lett 99:39–47