IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 12, 2021, accepted May 6, 2021, date of publication May 12, 2021, date of current version May 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3079397

An Automated Method for Identification of *Key frames* in *Bharatanatyam* Dance Videos

HIMADRI BHUYAN[1], PARTHA PRATIM DAS[1], JATINDRA KUMAR DASH[2], AND JAGADEESH KILLI[1]

[1]Indian Institute of Technology Kharagpur, Kharagpur 721302, India
[2]Department of Computer Science and Engineering, SRM University AP, Amaravati 522503, India

Corresponding author: Himadri Bhuyan (himadribhuyan@gmail.com)

ABSTRACT Identifying *k*ey frames is the first and necessary step before solving the variety of other *B*haratanatyam problems. The paper aims to partition the momentarily stationary frames (*key frame*s) from this dance video's motion frames. The proposed *key frame*s (KFs) localization is novel, simple, and effective compared to the existing dance video analysis methods. It is distinctive from standard KFs detection algorithms as used in other human motion videos. In the dance's basic structure, the occurrence of KFs during performances is often not completely stationary and varies with the dance form and the performer. Hence, it is not easy to decide a global threshold (on the quantum of motion) to work across dancers and performances. The earlier approaches try to compute the threshold iteratively. However, the novelty of the paper is: (a) formulating an adaptive threshold, (b) adopting Machine Learning (ML) approach and, (c) generating the effective feature by combining three frame differencing and bit-plane technique for the KF detection. In ML, we use Support Vector Machine (SVM) and Convolutional Neural Network (CNN) as the classifiers. The proposed approaches are also compared and analyzed with the earlier approaches. Finally, the proposed ML techniques emerge as a winner with around 90% accuracy.

INDEX TERMS *K*ey frame, *A*davu, three frame difference, bit-plane extraction, adaptive threshold, machine learning.

I. INTRODUCTION

Bharatanatyam[1] is mostly practiced oldest Indian Classical Dance (ICD) form. Using this dance form, the dancer illustrates the *Hindu* religion themes and spiritual ideas with elegant footwork, impressive body postures, emotional facial expression, and hand gestures. All these well-defined gestures, postures (*Key postures*), movements (motions), and transitions are the units of an *Adavu*. It is used to train the dancers. Like the other dance forms, it is also audio driven. The dancer follows the rhythmic beats (*Tal*) in audio to perform the *Adavu*. The Fig.1 shows an example of *Adavu*. The occurrence of the sequence of *Key postures* (*KP01-KP02-KP03-KP02*) and beats are shown. The transformation of one posture to another involves a motion. As stated earlier, each posture/motion is driven by an audio beat shown in the figure. The Fig.1 also shows the sequence of beats (*tei-yum-tat-tat-tei-yum-ta*) associated with the postures and movements. For example, *KP01, KP02, KP03*, and *KP02* synch with the beats *tei-yum, tat-tat, tei-yum,* and *ta* respectively during the performance. The sample video is available in [1].

The complex postures, gestures, and the attired [1] of the dancers are vital in the perspective of image processing and computer vision aspects while a computer analysis or/and interpretation of the dance form is the intention.

During the performance of *Adavu* (video stream), the dancer takes a momentarily stationary pose (*Key postures*) followed by some simple/complex motions. Our objective is to detect the occurrence of KFs during the *Key posture*s (KPs). This detection is a variant of the video segmentation problem [2]. Here, we distinguish the momentarily stationary frames (KFs) from the *motion frames* (MFs) in a given *Adavu* video. It may be noted that *key frame* segmentation is a necessary first step for a variety of other problems in *Bharatanatyam* dance analysis like: (a) Distinguishing KP and transitions in-between, (b) Recognition of *Adavu*s based on the occurrence of KP sequences, (c) Distinguishing KF to MF for automated motion and KP annotation, (d) Identifying various limb trajectories in motion, and (e) Dance transcription, etc.

[1] An Indian Classical Dance form approved by *Sangeet Natak Akademi* and the Ministry of Culture, Govt. of India.

The associate editor coordinating the review of this manuscript and approving it for publication was Hao Luo.

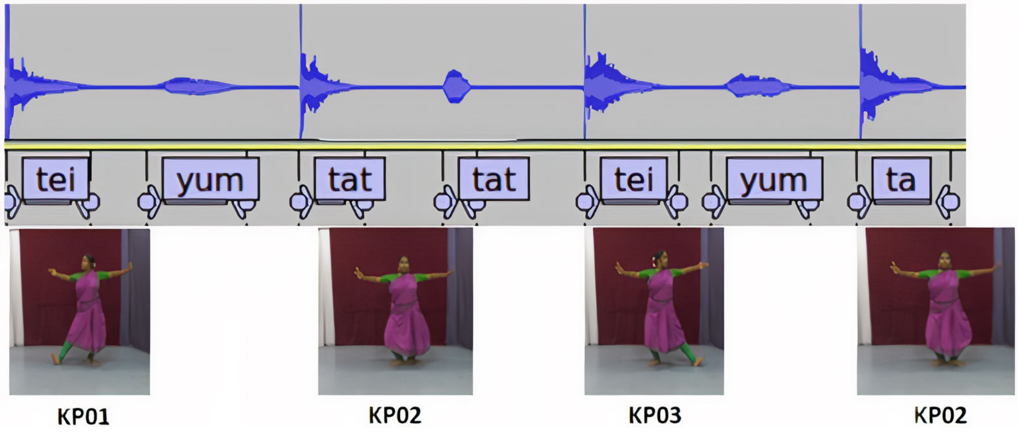


FIGURE 1. Example of an *Adavu* performance.

While interests in automated dance analysis are on the rise, this segmentation problem is non-trivial with several challenges: (a) During the transition from one KP to another, we may encounter a very slow motion at the starting. These slow motioned frames may be falsely classified as KFs. (b) With complex movements and postures, the distinction between KFs and MFs may not be easy. (c) The intricate dress style (mainly below the waist) contributes to the occlusion, and sometimes, it drives the non-visibility. This attire may lead to misinterpretation of MFs as KP.

II. MOTIVATION & RELATED WORK

The two frame differencing [3]–[7] are the apparent approaches to detect the motion and non-motion frames. However, it requires strict, static, and manually adjustable thresholds, which vary with each video. Similarly, the optical flow technique [8] and the recently deep learning approach also serve the same purpose but incurs heavy computation load.

The literatures; [9]–[16] on posture & gesture recognition or motion classification overlooks the segmentation issue. They assume that a pre-segmented (pre-annotated) sequence of frames is available for analysis. The authors in [9] develop a system to recognize 3D dance postures. A method is proposed in [10] to classify the different ICDs using the pose descriptor of each frame of an ICD video. In [11], authors propose a gesture recognition algorithm to recognize the gestures; happiness, fear, anger, relaxation, and sadness in ICD using the joint coordinates captured by Kinect. Reference [12] proposes, deep learning-based *Convolution Neural Network* (CNN) to recognize body postures and hand gestures that convey the ICD performances' semantics. Reference [13] recognize the posture from 2D images of *Ballet* dance and claim their approach can be implemented in posture recognition from video sequences. Using CNN, [14] identify the dance poses and classify the ICD form with the help of SVM. Reference [15] classify the motions involved in the *Natta Adavu* of *Bharatanatyam* dance. The authors in [16] recognizes the action in ICD using the dance postures from the YouTube videos. Similarly, in [17], feet postures are classified in South Indian Classical Dance. The authors in [18] create several possible coveted dance sequences for a piece of particular dance music using learning examples. They use deep learning to learn the examples. It is nothing but dance choreography. A discriminator is used to identify the best feat among the several possible combinations. The combination is the set of postures and motions involved during a performance. In [19], authors classify the images of ICD forms (Kathak, Bharatanatyam, Yakshagana, Odissi, Kathakali) using deep learning (CNN). Hand gestures (24 classes) classification of *Kathakali* dance is done by [20] using SVM and CNN and achieve an accuracy 74%.

In all these above works, whether it is a problem of recognition or classification, firstly, the segmenting or distinguishing KF and MF in the videos were necessary and supposed to be addressed.

In contrast to the above, [21] propose a method to automatically segment the gestures from dance sequences. References [22] and [23] work on musical information for motion structure analysis and gesture segmentation. It is done by detecting the onset from the music signal and tracking the beats. Authors in [3], do a similar work, where they can mark the start point of non-motion frames using the onset beat detection method. In this, image differencing and instantaneous acceleration use to segment the KFs. Here they tried manual thresholding for this segmentation and achieved an average accuracy of 74%. Nevertheless, using onset beat detection, they score 84%.

In [24], based on the motion captured data, Labanotation is generated automatically. This article segments the motions considering the kinematics of limb joints (skeletal joints) and the rhythm. The authors in [16] recognize the actions in the given dance videos using a learning-based embedded system and the music information to segment the actions.

Though there is no much literature found in the segmentation of dance videos, we explore non-dance videos that help us achieve our target. In [25], authors use background subtraction and three frame differencing to detect the object movement. Similar work is also done in [26]. They adopt two frame differences and the 4-bit plane extraction technique. In these works, they also use the manual threshold to distinguish motion and non-motion objects.

The advancement in deep learning techniques helps us to solve many computer vision problems proficiently. So, we try to explore some of the deep learning works [27]–[33] as well. These are identifying the KFs in the videos to recognize/classify human action or behavior. The approach in [27] extracts the frame's feature with the help of a CNN. Then it uses a discriminative score to pool the desired frames from the given video. These pooled frames are used to classify an action. Reference [28] identifies the KFs using CNN and reported precision of 92% in the human action videos. The authors in [29] identify 4 KFs in the sports videos with the help of fully convolutional networks. The author in [30] feeds both the RGB stream and the optical flow to CNN. Finally, it applies a smoothing fitting function to distinguish KFs from the entire video sequence using regression. To reduce the complexity of real time face recognition application on IoT edge devices [31], a CNN is applied to extract KFs based on the face image quality. Similar works also report in [32], [33], associated with video surveillance and security system. In [32] the frames containing similar faces are grouped using a clustering algorithm where deep CNN does feature extraction.

From the above discussions, it becomes clear that the KFs contribute substantially to a particular action. However, the definition of KF is not absolute. So, its extraction algorithm may not be same for all the problems. Hence, we propose a novel approach to solve KF detection problem where KF is a momentarily stationary frame. However, the approach can also be applied to non-dance videos though it is tested on dance data that is more complex than the usual human action.

We explore the following shortcomings concerning the discussed literature leading to five significant contributions of this paper.

- **Shortcomings:** (a) No adaptive threshold is developed yet. (b) For each video, a manual threshold is defined for classification. (c) No automation to compute the threshold. (d) ML techniques are yet to be explored in dance videos
- **Major contributions**:
  - Combining three frames difference and bit-plane (3-MSBs) techniques for segmentation.
  - Segmentation of Dance video.
  - Adaptive threshold is devised successfully for Non-ML approach.
  - Explores ML technique (SVM and CNN).
  - The proposed approach is compared with the contemporary methods to analyze and evaluate the performance.

The list of tried techniques/approaches are shown in the Table- 1

TABLE 1. Comparative study (NA: Not Attempted).

Technique	Input Image			
	BGS Image		Non-BGS Image	
	ML	Non-ML	ML	Non-ML
Image diff	NA	✓	NA	✓
Image diff + Bit-plane	✓	✓	✓	✓

III. DATA SETS

Bharatanatyam Adavu videos are captured at 30 fps by Microsoft Kinect 1.0 [34] using Nuicapture software, [35]. Each recorded video comprises depth, skeleton, RGB, and audio streams. There exist 15 *Adavu*s consisting of 58 variations in total. Three different dancers perform each variant. However, we used only 166 videos by ignoring the erroneous recordings. The average number of frames in each video is 700-1000 frames. The Table- 2 shows the data set. A part of the data set is also made available on the web by [1].

We annotate all these videos manually. As a sample, the annotation of *Mettu Adavu* is shown in Table- 3. A particular annotation file provides information about the occurrence of KFs and *Non-KFs* (motion frames–MFs). Any frame that belongs to KF is called a *Key posture* (KP). Since *Adavu* follows a rhythmic pattern, there are motion frames between two KPs. In other words, KPs follow the motion frames. In annotation, ID comprises the variation of *Adavu*, Dancer#, Cycle #, music beat #, and KP #. In the 1^{st} row, the information M4D1C1B01P21 implies that $Adavu = Mettu4$ (M4), Dancer# = 1 (D1), KP# = 21 (P21), Cycle# = 1 (C1: The repeat of the rhythmic pattern), and Beat# = 01 (B01: The music beat at which the KP occurs). In this paper, beat and cycle information are not necessary.

IV. PROPOSED METHOD

Flow chart in Fig.2 illustrates the proposed method. It takes an *Adavu* video as an input and segment the entire video into motion and nonmotion frames, but our objective is the KF extraction. It mainly comprises three parts: (a) Preprocessing, (b) Feature Extraction and Average Filtering, and (c) Classification. Further, for classification, two techniques are followed:

- Non-ML technique: It follows (a) Adaptive threshold computation, (b) Classification of KFs (True Positive) and MFs (True Negative) using an adaptive threshold, (c) Reduction of False Negative and False Positive through majority voting.
- ML technique: SVM and CNN

A. PRE-PROCESSING

To meet the requirement and reduce the algorithm's complexity, the transformation of visual data is necessary. Therefore the RGB frames are converted to grayscale frames.

1) RGB TO GRAY IMAGE

The gray level is a critical factor in detecting the motion. Moreover, it helps in reducing the complexity of the used

TABLE 2. **Data set.**

Adavu Name	Variations	# of Dancers	# of Recordings	*Adavu* Name	Variations	# of Dancers	# of Recordings
Joining	3	3	9	Pakka	2	3	6
Kartari	1	3	3	Sarika	4	3	12
Nattal	8	2	16	Sarikkal	3	3	9
Tattal	5	3	15	Tatta	8	3	24
Mandi	2	3	6	Tei-TeiDhatta	3	3	9
Mettu	4	3	12	Tirmana	3	3	9
Natta	8	3	24	Utsanga	1	3	3
Paikal	3	3	9	Total	58		166

TABLE 3. **A sample annotation file.**

ID	Start Frame #	End Frame #
M4D1C1B01P21	49	61
M4D1C1B02P22	65	75
M4D1C1B03P12	103	120
M4D1C1B04P09	124	143
....	...	...
M4D1C2B14P26	972	986
M4D1C2B15P27	1006	1027
M4D1C2B16P28	1034	1065

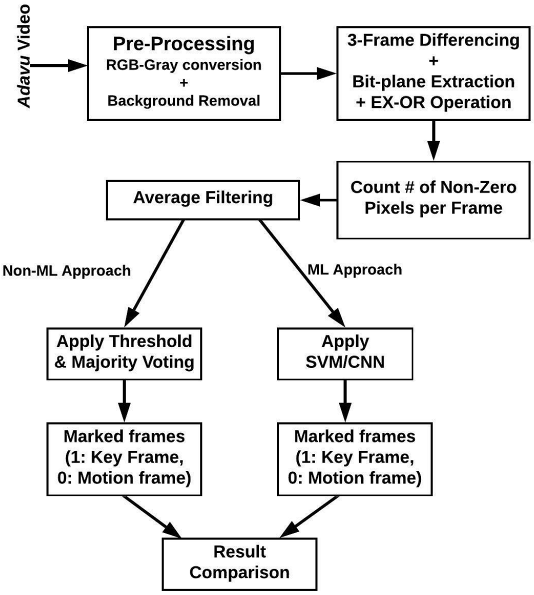


FIGURE 2. **Flow diagram of proposed method.**

algorithms. The color image of size 480 × 640 is converted to gray image, refer (1).

$$\begin{aligned} GrayFrame(i,j) &= 0.299 * Frame(i,j)_R \\ &\quad + 0.587 * Frame(i,j)_G \\ &\quad + 0.144 * Frame(i,j)_B \end{aligned} \tag{1}$$

where $1 \leq i \leq 480$ and $1 \leq j \leq 640$. Suffixes denote the components.

2) BACKGROUND SUBTRACTED IMAGE

The paper removes the gray image background by retaining only the dancers' information using the Kinect recorded data set's depth stream information. Kinect depth information is stored in 16 bits, where the first three bits denote player index, and the next 13 bits hold the depth data in mm. Kinect makes use of the player index to detect the whole user. To indicate a user, it marks the pixels with an index of 1 to 7. This index value determines whether a given pixel is part of user 1, 2, and/or…,7. It assigns index zero if a pixel is not part of the image of a user. RGB camera and the depth camera are not calibrated, so each camera's pixels are not correspondent. Hence, Kinect provides a mapping technology called depth frame to color frame map, [36], which uses depth data. The mapping is used in the Algorithm-1 during background subtraction from a given gray image.

Algorithm 1: Background Removal

```
Input: GrayImage, DepthData
Output: BGSImage // Background subtracted Image
Initialization: Set Mask(i, j) = 0          // i:1-M, j:1-N
for $i = 1$ to $M$ do
                        // Frame size = $M \times N$
    for $j = 1$ to $N$ do
        [x, y]=MapDepthToColorFrame(DepthData(i,j))
        if $PlayerIndex(i, j) > 0$ then
            Mask(x, y) = 1
BGSImage = GrayImage * Mask
return BGSImage
```

B. FEATURE EXTRACTION

This section describes the feature extraction, which is to be used as an input to classify the KF and non-KF in a given *Adavu* video. Initially, three frame differencing is computed, and then bit-plane is extracted. Finally, average filtering is performed to minimize the noise.

1) THREE FRAME DIFFERENCING AND BIT-PLANE EXTRACTION

The relative change can be detected by temporal differencing in the successive frames. As per [7], three frame difference works better than the two-frame difference to detect a real moving target. Here we consider 3 consecutive frames (F_k, F_{k+1} and F_{k+2}) in a sliding window fashion, resulting in two temporal differentiated images, $d_\alpha(i,j) = |F_{k+1}(i,j) - F_k(i,j)|$ and $d_{\alpha+1}(i,j) = |F_{k+2}(i,j) - F_{k+1}(i,j)|$. d_α and $d_{\alpha+1}$

generate pixel wise absolute intensity differences between two successive frames. Since MSBs contribute most to the intensity values during motion, 3-MSB values are extracted and merged as given in (2) to compute $B_\alpha(i,j)$ and $B_{\alpha+1}(i,j)$. On the resultant gray frames $B_\alpha(i,j)$ & $B_{\alpha+1}(i,j)$, we consider 3 MSBs and do the Ex-OR operation to compare and compute the pixel difference between bit-planes. It is saved as the resultant value of F_{k+1} frame, that is $F_{k+1} = B_\alpha(i,j) \oplus B_{\alpha+1}(i,j)$. It is the relative change of F_{k+1} with respect to F_k and F_{k+2} where $b =$ Bit position, $0 \leq b \leq 7$, $d(i,j)_b =$ Intensity at b:

$$B_\alpha(i,j) = \sum_{b=5}^{7} 2^b * d_\alpha(i,j)_b \tag{2}$$

The steps followed for three frame difference and the bit-plane extraction is given in the Algorithm 2.

Algorithm 2: Three Frame Differencing and Bit-Plane Extraction, Procedure Name: *ThreeFrameBitplane*()

```
Input: $F_k$ // Where $k = 1, 2, \ldots n-2$. $n =$ Number of frames/video
Output: $ResultF_k$
for $k = 1$ to $n-2$ do
    for $i = 1$ to $M$ do
                        // Frame size = $M \times N$
        for $j = 1$ to $N$ do
            $d_k(i,j) = |F_{k+1}(i,j) - F_k(i,j)|$
             // pixel wise differencing is Done
            $d_{k+1}(i,j) = |F_{k+2}(i,j) - F_{k+1}(i,j)|$
            $B_k(i,j) = Extract3MSB(d_k(i,j))$
            $B_{k+1}(i,j) = Extract3MSB(d_{k+1}(i,j))$
            $ResultF_{k+1}(i,j) = B_k(i,j) \oplus B_{k+1}(i,j)$
return $ResultF_k$
```

2) AVERAGE FILTERING

The output of Algorithm-2 provides the frames where we count non-zero pixels. Average filtering is applied using a particular mask size and weight to reduce the error. The number of non-zero pixel count per frame is considered as the feature for classification. Section IV-C2 discusses this in detail.

C. CLASSIFICATION

Based on the non-zero pixel counts per frame, a frame can be marked as KF or Non-KF (MF). The challenge here is to determine the threshold for a frame being KF or MF. Most of the algorithms compute this threshold heuristically through observation over series of frames. In this work, we present an adaptive method to calculate this threshold. The non-zero pixels per frame are considered the feature set for the Non-ML algorithm or the ML Algorithm to segment a given video. In the Non-ML approach, an adaptive threshold is devised for the segmentation, whereas in the ML method, trained SVM and CNN are used.

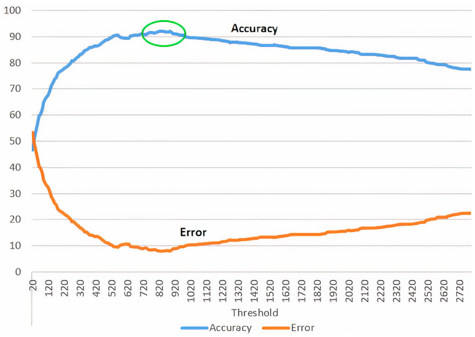


FIGURE 3. **Accuracy and error against varying threshold of *Mettu-1*.**

1) NON-ML APPROACH

The first step in this approach is to automate the computation of each video threshold for the segmentation. To start with, the paper extensively tries out with several thresholds (step size 10) iteratively and computes the accuracy for each. Fig.3 shows the variation in the segmentation's accuracy and error by varying the thresholds for Dancer-3, *Mettu-1*. From this extensive study, initially, we detect the threshold that attains maximum accuracy. Next, when we analyze the entire range of thresholds and their corresponding accuracy, it is found that there exist a set of consecutive thresholds where no much variation is identified in the accuracy. These are close to the maximum accuracy attended threshold. Moreover, this maximum accuracy attended threshold is very close to the devised threshold, as shown in (3), Where $ResultF_{k+1}$ is generated by Algorithm 2 and $\{k = n-2 | n =$ number of frame in an video$\}$

$$Threshold = \frac{\sum_{i=1}^{k} CountNonZeroPixels(ResultF_{i+1})}{k} \tag{3}$$

The computed *Threshold* become the decider on the basis of the number of non-zero pixels in the $ResultF_k$ to mark a frame as KF(1) or MF (0). The marking is stored in a 1-D array (*MarkedFrames*). For the Computed Threshold, refer (3), we analyze the thresholds across all the *Adavu* videos. We identify that the computed and maximum attended accuracy differ in the range 0.55%–1.79%. For illustration, we use the *Mettu Adavu* as an example. In Fig.3, the green circle shows the steady accuracy close to maximum attended accuracy 92.12% in the range of thresholds 820-900 for the Dancer-3, *Mettu-1* video. The computed adaptive threshold (refer (3)) comes out to be 906. For the threshold=906, the $ComputedAccuracy = 91.57\%$, close to the maximum attended accuracy of 92.12% Here, $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$, where $TP =$ True positive (KF), $TN =$ True Negative (MF), $FP =$ False Positive and $FN =$ False Negative samples.

2) MAJORITY VOTING: REDUCTION OF FP AND FN IN NON-ML

As we discussed earlier in Section-III, a sequence of KFs follows a sequence of MFs and vice versa. So in the automatic detection of KF and MF, at times, there may be a false KF detection in between MFs or a false MF detection in between KFs. To deal with such kind of situation, we adopted the two majority voting techniques.

- **Approach-1**:
  In this approach, we consider a window acquiring three frames and checking if there is a key frame in between two motion frames and vice versa. The Algorithm-3 achieves this using three frame sliding window.

```
Algorithm 3: Majority Voting, Approach-2
  Input: $MarkedFrames[1, 2 \ldots k]$
  Output: $UpdatedMarkedFrame[1, 2 \ldots k]$
  $UpdatedMarkedFrame = MarkedFrames$
  for $i = 1$ to $k\text{-}2$ do
      $W_i = MarkedFrame[i]$ $W_{i+1} = MarkedFrame[i+1]$
      $W_{i+2} = MarkedFrame[i+2]$
      if $W_i == 0$ && $W_{i+2} == 0$ && $W_{i+1} == 1$ then
          $UpdatedMarkedFrames[i+1] = 0$
      if $W_i == 1$ && $W_{i+2} == 1$ && $W_{i+1} == 0$ then
          $UpdatedMarkedFrames[i+1] = 1$
  return $UpdatedMarkedFrame$
```

- **Approach-2**:
  In this type, we consider a five-window frame and count the number of zeros and ones. If the number of zeros is more than the ones, then the center frame in the window is marked as KF. If the reverse is the case, then the middle frame in the window is marked as a motion frame. The Algorithm-4 describes this approach.

We analyze across all the performances and the dancers and conclude that the Approach-2 majority voting technique performs slightly better than the Approach-1. Table-4(a) shows the average accuracy of each of the *Adavu* variant by applying Approach-2. Here, accuracy refers to the precision of KF (*PrecisionKF*); correctness in KF prediction in each video. Along with the Precision, Recall, and F1 score metrics are also necessary for evaluating any binary classification algorithm's performance. Reference [37] explain the equations associated with these measures. When an algorithm gives good precision and recall results, the F1 score is also yield to be good. The good F1 Score value implies the robustness of the given algorithm.

```
Algorithm 4: Majority Voting, Approach-2
  Input: $MarkedFrames[1, 2 \ldots k]$
  Output: $UpdatedMarkedFrame[1, 2 \ldots k]$
  $UpdatedMarkedFrame = MarkedFrames$
  for $i = 1$ to $k\text{-}4$ do
      $W[1, 2..5] = MarkedFrame[i, i+1, \ldots i+4]$
      $count0 = countZeros(W)$
      $count1 = 5 - count0$
      $MF = MarkedFrames[i+2]$
      if $count0 > count1$ && $MF == 1$ then
          $UpdatedMarkedFrames[i+2] = 0$
      if $count1 > count0$ && $MF == 0$ then
          $UpdatedMarkedFrames[i+2] = 1$
  return $UpdatedMarkedFrame$
```

3) MACHINE LEARNING APPROACH

The best part of the supervised machine learning approach is learning the threshold for classification from the given training data. This paper uses SVM and CNN as classifiers to classify the incoming video sequence into KFs and MFs. Fig.4 shows the training and testing process.

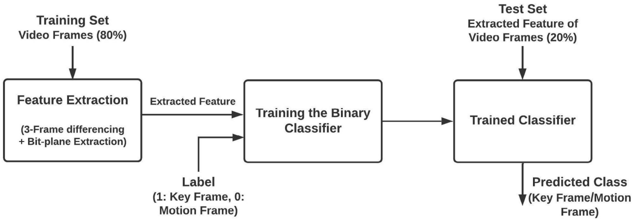

Training Set
Video Frames (80%)
Feature Extraction
(3-Frame differencing + Bit-plane Extraction)
Extracted Feature
Training the Binary Classifier
Trained Classifier
Test Set
Extracted Feature of Video Frames (20%)
Label
(1: Key Frame, 0: Motion Frame)
Predicted Class
(Key Frame/Motion Frame)

FIGURE 4. Train and test flow to classify *Key frame* & motion frame.

SVM As classifier: Like any ML model, the SVM also requires each frame's features to train the model and test as well. The features are the resultant pixel values generate for each frame by Algorithm-2, as discussed in Section IV-B. The dimension of the feature is $1 \times 3, 07, 200$ since the size of the each frame = 480×640. It says, for each frame, we provide 3,07,200 values to train the SVM and test as well. We supply the labelled (1: KF, 0: MF) feature array to SVM during training. While testing, we provide the unlabeled feature sets to our trained SVM model, and the model predicts whether a given feature set belongs to KF or MF. We use MATLAB for SVM implementation. The functions *fitcecoc* [38] (Classifier: SVM–OneVsOne, ClassNames: {'setosa' 'versicolor'} where setosa = 1 and versicolor = −1, CodingName: 'onevsone') and *predict* [39] of MATLAB 2018b are used for SVM classifier training and testing. *fitcecoc* returns trained model (*Mdl*) using the training features F_{tr} and the labels (1: KF and 0:MF). *predict* uses *Mdl* and the test features F_{ts} to predict class labels.

The Table-4(b) shows the classification accuracy. Here, we present the result of KF detection in the form of precision, recall and F1 Score like non-ML approach.

CNN As Classifier: To improve the accuracy, we try CNN. Fig.4 shows its train and test flow. The Algorithm 2 generates the necessary feature for each frame. These features are input to the proposed CNN [40] (epochs: 30, bath size: 100, momentum: 0.9, learning rate: 0.01). It is implemented in Python 3.7. The *VGG16* CNN architecture is modified to accommodate the type of data we are dealing with.

TABLE 4. Classification result of both ML and non-ML approach.

(a) Result (%) in Non-ML Approach

Adavus	BGS (KF)			WBGS (KF)		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Joining	83.69	64.46	72.82	79.94	64.03	71.10
Kartari	98.53	49.08	65.52	97.09	50.19	66.17
Nattal	92.16	67.11	77.66	91.26	65.96	76.57
Tattal	85.61	57.07	68.48	84.87	58.46	69.23
Mandi	89.54	74.74	81.48	89.86	74.10	81.22
Mettu	93.16	87.73	90.36	93.22	86.74	89.87
Natta	85.51	82.56	84.01	85.10	82.29	83.67
Paikal	92.14	57.69	70.95	89.64	44.74	61.05
Pakka	77.84	43.32	55.67	74.53	41.24	56.49
Sarika	72.47	68.74	70.55	72.39	68.52	70.40
Sarikkal	89.24	65.43	75.50	90.49	63.82	74.85
Tatta	81.49	87.54	84.41	78.26	90.16	83.79
Tei-Dhatta	84.19	46.82	60.17	83.22	49.71	62.24
Tirmana	77.57	56.48	65.36	77.34	56.34	65.19
Utsanga	67.63	49.81	57.36	70.15	49.55	58.08
Average	**84.72**	**63.90**	**72.02**	83.82	63.04	70.97

(b) Result (%) in ML Approach(SVM)

Adavus	BGS (KF)			WBGS (KF)		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Joining	92.84	74.57	82.71	95.40	69.44	80.38
Kartari	95.11	68.29	79.50	88.42	47.89	62.13
Nattal	95.05	78.18	85.79	80.01	74.27	77.04
Tattal	92.09	85.03	88.42	87.67	84.35	85.98
Mandi	90.61	85.49	87.98	86.38	80.19	83.17
Mettu	95.67	85.41	90.25	97.17	90.65	93.80
Natta	94.48	86.25	90.18	94.18	84.81	89.25
Paikal	96.65	41.55	58.11	96.08	44.74	61.05
Pakka	74.26	64.45	69.01	68.26	58.93	63.25
Sarika	86.44	76.01	80.89	89.23	70.17	78.56
Sarikkal	91.44	67.81	77.87	95.68	69.18	80.30
Tatta	98.81	97.17	97.98	97.74	96.10	96.91
Tei-Dhatta	89.97	82.52	86.08	91.37	68.75	78.464
Tirmana	73.71	71.95	72.82	76.22	66.46	71.01
Utsanga	81.87	60.89	69.84	71.05	50.02	58.70
Average	**89.93**	**75.04**	**81.16**	87.66	70.44	77.33

The original implementation of *VGG16* is trained with colored images with many features like objects, boundaries, and color gradients. However, our data mainly contains grayscale images with a small number of non-zero pixels. We highly reduce the model's complexity to fit it in two convolutional layers and three fully connected layers.

Each convolution layer is a combination of convolution, non-linear *ReLU*, normalization, and pooling. Instead of *ReLU* which annihilates negative input features, we employed *LeakyReLU* which penalizes negative features instead of annihilating it. Each of the first two fully connected layers comprises of linear layer, non-linearity *ReLU*, drop-out, and batch normalization. We used batch-normalization only at fully connected layers and used layer-normalization after every convolution. Layer normalization normalizes the features of only a single sample, where batch normalization normalizes features of the sample in the entire batch along each channel. Also, a form of regularization called drop-out with a rate of 0.1 is used to prevent overfiting. During training, *Adam* is employed to optimize the network, and cross-entropy-loss is used for loss calculation

The dataset contains an unequal number of samples belonging to different labels. If the number of samples of each label differs significantly, it will affect the classifier to settle on poor generalization. To prevent this, we use random oversampling [41], [42]. It simply keeps the samples of the majority label intact and replicates the samples of the minority class. The random oversampling randomly select samples from the minority class until it gets samples equal to that of the majority class. The random distribution function is used such that every sample from the minority class has an equal probability of getting selected.

The Table-5 shows the classification result inform of precession, recall, and F1 score. The table also compares the results of SVM with the CNN

V. RESULT ANALYSIS

In this section, we analyze the results in various dimensions. First, we discuss the variations in the proposed approach's performance while varying the form of input data. Secondly, we compare our proposed approach with the contemporary ones.

A. PERFORMANCE ANALYSIS

In this section, few more experiments are performed to examine the effect of background subtraction on the proposed

TABLE 5. Classification result of CNN and SVM.

Adavus	Using CNN			Using SVM		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Joining	93.6	88.37	90.91	92.84	74.57	82.71
Kartari	90.28	99.24	94.55	95.11	68.29	79.50
Nattal	95.84	83.33	89.15	95.05	78.18	85.79
Tattal	93.28	67.22	78.14	92.09	85.03	88.42
Mandi	95.7	87.31	91.31	90.61	85.49	87.98
Mettu	88.4	95.14	91.65	95.67	85.41	90.25
Natta	92.36	93.18	92.77	94.48	86.25	90.18
Paikal	98.34	97.34	97.83	96.65	41.55	58.11
Pakka	80.25	59.27	68.18	74.26	64.45	69.01
Sarika	88.65	81.8	85.09	86.44	76.01	80.89
Sarikkal	93.62	75.63	83.66	91.44	67.81	77.87
Tatta	93.45	95.55	94.49	98.81	97.17	97.98
Tei-Dhatta	85.65	87.5	86.57	89.97	82.52	86.08
Tirmana	83.02	87.71	85.3	73.71	71.95	72.82
Utsanga	84.31	82.69	83.5	81.87	60.89	69.84
Average	**90.45**	**85.41**	**87.54**	89.93	75.04	81.16

method's performance. Here, we implement two variants for each of the proposed approaches (Non-ML and ML). Each of the proposed techniques is implemented with and without the background subtraction as the first step. The entire process boils down to five categories:

- Non-ML without Background subtracted data (Non-ML WBGS)
- Non-ML with Background subtracted data (Non-ML BGS)
- ML (using SVM) without Background subtracted data (ML WBGS)
- ML (using SVM) with Background subtracted data (ML BGS).
- ML (Using CNN) without Background subtracted (ML-CNN) data

When we compare Non-ML WBGS and Non-ML BGS, we observe that background subtracted data sets identify KF more correctly than the WBGS data. The 7(a) and Table-4(a) shows the comparison. Most of the *Adavu* videos perform slightly better when the background is subtracted from the scene. For BGS, the average accuracy is 84.72%, and for WBGS, it is 83.82%. Most importantly, the average F1 Score is above 70% in both cases, which implies our approach is balanced.

While comparing ML WBGS with BGS, we observe ML BGS approach performs much better. The comparison is visible in the Fig.7(b) and Table-4(b) as well. But, in between Non-ML BGS and ML BGS, in most of the videos, ML BGS's performance is above 90%, that is, it performs better. The Fig.7(c) shows comparison. In few *Adavus* (*Karatri, Pakka, Tirmana*), the Non-ML approach performs slightly better than ML. The reason may be, a very slow motioned frame sometimes get detected as a motion in ML, whereas in Non-ML, the threshold excludes that. On the other hand, the F1 score is improved by 9% in the ML approach, which is a good sign.

Finally, while comparing the ML-CNN Precision with the SVM (ML-BGS), CNN performs slightly better in most of the cases. Table-5 shows the comparison between these two approaches. Though in some *Adavus* (*Kartari, Mettu, Sarikkal, Tatta*) Precision of SVM is better, but in most of the *Adavus* CNN gives marginally better result. Fig. 5 shows the comparison of the precision. If we look into the overall F1 Score, then CNN is more balanced with 87.54%. However, in SVM, it is 81.16%. At the same time, Recall of CNN is also better than the SVM except *Tattal Adavu*. Fig. 6 shows the comparison of Recall and F1 Score between these two approaches.

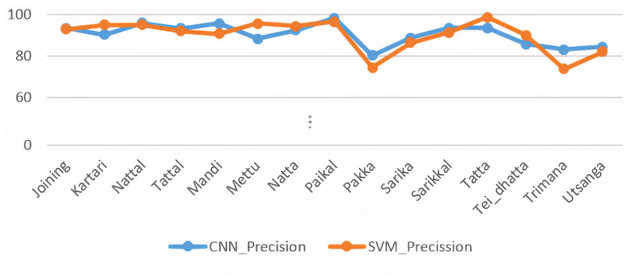


FIGURE 5. Precision comparison: CNN vs. SVM (ML-BGS).

B. A COMPARISON WITH CONTEMPORARY APPROACHES

From the literature, we identify some of the contemporary approaches which can be compared with our approach. In [3], the authors tried two image differencing techniques to detect KFs in *Bharatanatyam* dance which is less effective than our Non-ML approach, where we try with background subtraction. Fig.7(d) shows comparison. It also shows the number of *Adavu* variants associated within the braces, that is, a glimpse of the data set used by [3]. In some cases, the current approach tries more variants and more videos (Table- 2). In this comparison, we identify three videos; *Sarrika, Tatta & Utsanga* which perform better in [3] than the current method.

When we compare our ML(SVM: ML-BGS) approach with [3], the current approach performs much better. However, in the ML approach, almost all the video performance is close to or above 90% except *Pakka, Tirmana & Utsanga*.

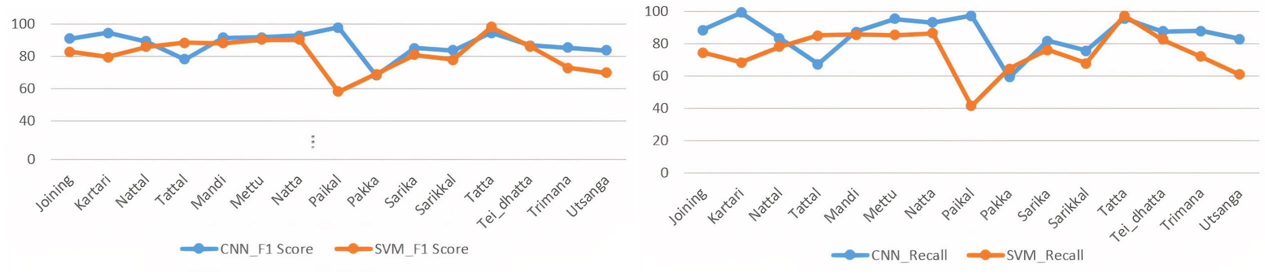
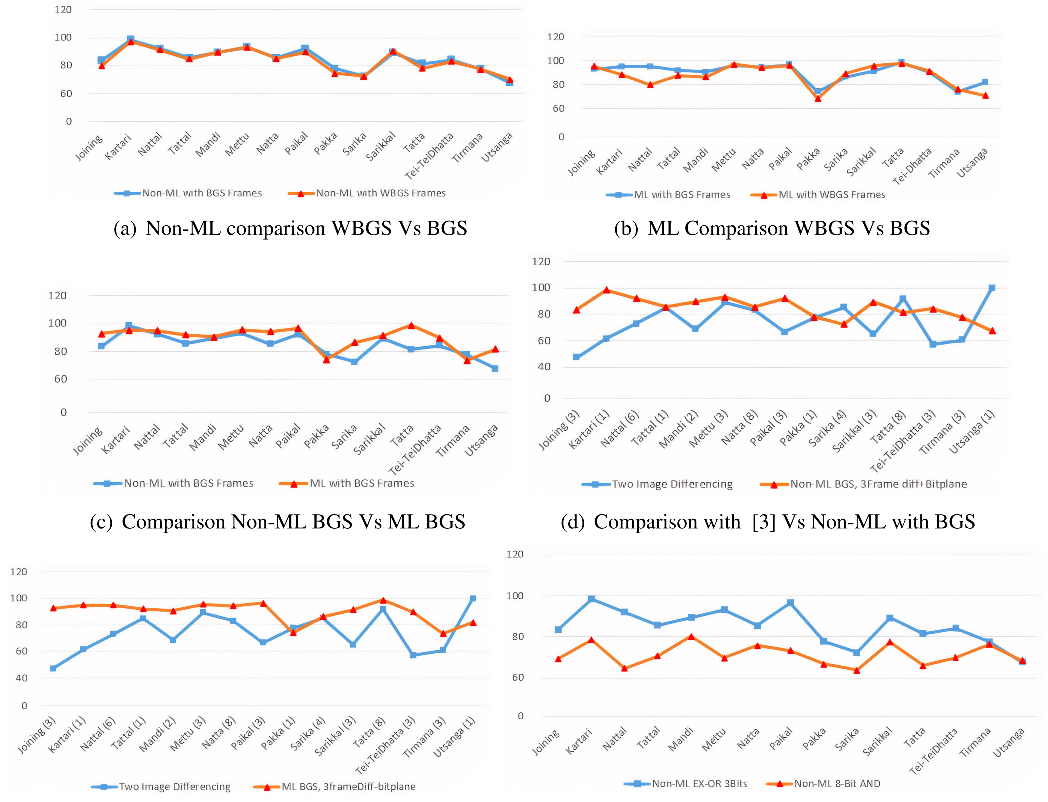


FIGURE 6. F1 score and recall comparison: CNN vs. SVM (ML-BGS).



(a) Non-ML comparison WBGS Vs BGS

(b) ML Comparison WBGS Vs BGS

(c) Comparison Non-ML BGS Vs ML BGS

(d) Comparison with [3] Vs Non-ML with BGS

(e) Comparison [3] Vs ML with BGS

(f) Current Approach (Non-ML 3-Bit Ex-OR) Vs 8-bit AND

FIGURE 7. Result analysis: ML using SVM and non-ML approaches.

These three are between 70-80%, but those are higher than those of [3]. It is quite evident in the Fig. 7(e).

References [4]–[7] apply the two-image differencing technique and the Bit-wise (8-bit Plane) AND operation to detect motion frames in the videos. However, these works do not consider dance data. In curiosity, we compare their technique with ours to identify the KFs. The comparison is shown in the Fig.7(f). Our methods performs better with 8%-20% accuracy except for two *Adavus* (*Tirmana* & *Utsanga*). In these two, the performance of the earlier and the current approaches are very close.

VI. CONCLUSION

The proposed methods are able to extract the *key frames* in *Bharatanatyam* dance videos successfully. The paper also discusses the contemporary approaches, [3]–[7] and compares with the proposed approach. In the earlier practices, there was no mechanism to compute an adaptive threshold. Moreover, the previous methods attempt to segment motion and non-motion frames in human activities (walking, running, etc.). The paper solves the *key frame* classification problem for a new domain (ICD), where choosing the global threshold to segment the video is challenging.

In contrast to the traditional methods, [3]–[8], our novel approach; a combination of the three-frame differencing and bit-plane technique, serves the purpose in a better way. The paper successfully identifies an adaptive threshold with less computation for non-ML strategy. Moreover, well defined ML classifiers (SVM and CNN) are able to identify the KFs in the videos independently without the threshold. Based on the performance, the proposed approach does not lag behind the earlier methods. Our ML approaches (SVM and CNN) give consistently good results – more than 90% precision in most of the *Adavu* videos. However, CNN wins the race with better Precision, Recall, and F1 score (Fig. 5 & 6) despite the high-performance hardware dependency, low computational speed/expensive computation [43]. Simultaneously, the non-ML approach's performance is also not bad (above 85%). Hence, both methods (ML/Non-ML) outperform the earlier ones.

The proposed approach is yet to be tested in non-dance videos. It will be an essential tool to segment the videos of any Indian Classical Dance form. Furthermore, in the ML approach, we use samples each with 3,07,200 features, which affects SVM and CNN's time complexity. It may significantly reduce with the histogram of the given feature, which needs to be tested. Finally, the depth information of Kinect may be used as an input in the approach to explore the performance.

REFERENCES

[1] T. Mallick, H. Bhuyan, P. P. Das, and A. K. Majumdar. (May 2017). *Annotated Bharatanatyam Data Set*. [Online]. Available: http://hci.cse.iitkgp.ac.in

[2] N. Dimitrova, L. Agnihotri, M. Barbieri, and H. Weda, *Video Segmentation*. Boston, MA, USA: Springer, 2009, pp. 3308–3313.

[3] T. Mallick, P. P. Das, and A. K. Majumdar, "Characterization, detection, and synchronization of audio-video events in Bharatanatyam Adavus," in *Heritage Preservation*. Singapore: Springer, 2018, pp. 241–268.

[4] S. Singh and R. Talwar, "Review on different change vector analysis algorithms based change detection techniques," in *Proc. IEEE 2nd Int. Conf. Image Inf. Process. (ICIIP-)*, Dec. 2013, pp. 136–141.

[5] P. L. Rosin, "Thresholding for change detection," *Comput. Vis. Image Understand.*, vol. 86, no. 2, pp. 79–95, May 2002.

[6] M. Zanetti and L. Bruzzone, "A theoretical framework for change detection based on a compound multiclass statistical model of the difference image," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 1129–1143, Feb. 2018.

[7] Y. Zhang, X. Wang, and B. Qu, "Three-frame difference algorithm research based on mathematical morphology," *Procedia Eng.*, vol. 29, pp. 2705–2709, Jan. 2012.

[8] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43–77, 1994.

[9] F. Guo and G. Qian, "Dance posture recognition using wide-baseline orthogonal stereo cameras," in *Proc. 7th Int. Conf. Autom. Face Gesture Recognit. (FGR)*, Apr. 2006, pp. 481–486.

[10] S. Samanta, P. Purkait, and B. Chanda, "Indian classical dance classification by learning dance pose bases," in *Proc. IEEE Workshop Appl. Comput. Vis. (WACV)*, Jan. 2012, pp. 265–270.

[11] S. Saha, S. Ghosh, A. Konar, and A. K. Nagar, "Gesture recognition from indian classical dance using Kinect sensor," in *Proc. 5th Int. Conf. Comput. Intell., Commun. Syst. Netw.*, Jun. 2013, pp. 3–8.

[12] A. Mohanty, P. Vaishnavi, P. Jana, A. Majumdar, A. Ahmed, T. Goswami, and R. R. Sahay, "Nrityabodha: Towards understanding indian classical dance using a deep learning approach," *Signal Process., Image Commun.*, vol. 47, pp. 529–548, Sep. 2016.

[13] A. Konar and S. Saha, "Radon transform based automatic posture recognition in ballet dance," in *Gesture Recognition*. Cham, Switzerland: Springer, 2018, pp. 35–64.

[14] P. V. V. Kishore, K. V. V. Kumar, E. Kiran Kumar, A. S. C. S. Sastry, M. Teja Kiran, D. Anil Kumar, and M. V. D. Prasad, "Indian classical dance action identification and classification with convolutional neural networks," *Adv. Multimedia*, vol. 2018, pp. 1–10, Jan. 2018.

[15] H. Bhuyan, M. Roy, and P. P. Das, "Motion classification in Bharatanatyam dance," in *Proc. Nat. Conf. Comput. Vis., Pattern Recognit., Image Process., Graph.* Singapore: Springer, 2019, pp. 408–417.

[16] F. Ma, "Action recognition of dance video learning based on embedded system and computer vision image," *Microprocessors Microsyst.*, vol. 81, Mar. 2021, Art. no. 103779.

[17] S. Shailesh and M. V. Judy, "Computational framework with novel features for classification of foot postures in Indian classical dance," *Intell. Decis. Technol.*, vol. 14, no. 1, pp. 119–132, Mar. 2020.

[18] G. Sun, Y. Wong, Z. Cheng, M. S. Kankanhalli, W. Geng, and X. Li, "DeepDance: Music-to-dance motion choreography with adversarial learning," *IEEE Trans. Multimedia*, vol. 23, pp. 497–509, 2021.

[19] A. D. Naik and M. Supriya, "Classification of indian classical dance images using convolution neural network," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Jul. 2020, pp. 1245–1249.

[20] L. T. Bhavanam and G. Neelakanta Iyer, "On the classification of kathakali hand gestures using support vector machines and convolutional neural networks," in *Proc. Int. Conf. Artif. Intell. Signal Process. (AISP)*, Jan. 2020, pp. 1–6.

[21] K. Kahol, P. Tripathi, and S. Panchanathan, "Automated gesture segmentation from dance sequences," in *Proc. 6th IEEE Int. Conf. Autom. Face Gesture Recognit.*, May 2004, pp. 883–888.

[22] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Rhythmic motion analysis using motion capture and musical information," in *Proc. IEEE Int. Conf. Multisensor Fusion Integr. Intell. Syst., (MFI)*, Aug. 2003, pp. 89–94.

[23] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Detecting dance motion structure through music analysis," in *Proc. 6th IEEE Int. Conf. Autom. Face Gesture Recognit.*, May 2004, pp. 857–862.

[24] C. Cui, J. Li, D. Du, H. Wang, P. Tu, and T. Cao, "The method of dance movement segmentation and Labanotation generation based on rhythm," *IEEE Access*, vol. 9, pp. 31213–31224, 2021.

[25] X. Lian, T. Zhang, and Z. Liu, "A novel method on moving-objects detection based on background subtraction and three frames differencing," in *Proc. Int. Conf. Measuring Technol. Mechatronics Autom.*, Mar. 2010, pp. 252–256.

[26] N. Dastanova, S. Duisenbay, O. Krestinskaya, and A. P. James, "Bit-plane extracted moving-object detection using memristive crossbar-CAM arrays for edge computing image devices," *IEEE Access*, vol. 6, pp. 18954–18966, 2018.

[27] A. Kar, N. Rai, K. Sikka, and G. Sharma, "AdaScan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3376–3385.

[28] U. Gawande, K. Hajari, and Y. Golhar, "Deep learning approach to key frame detection in human action videos," in *Recent Trends in Computational Intelligence*. London, U.K.: IntechOpen, 2020.

[29] M. Jian, S. Zhang, L. Wu, S. Zhang, X. Wang, and Y. He, "Deep key frame extraction for sport training," *Neurocomputing*, vol. 328, pp. 147–156, Feb. 2019.

[30] A. Diriba, "Keyframe-based saliency detection for human action recognition using deep learning," Ph.D. dissertation, Dept. Comput. Sci. Eng., ASTU, Guwahati, Assam, 2020.

[31] X. Qi, C. Liu, and S. Schuckers, "IoT edge device based key frame extraction for face in video recognition," in *Proc. 18th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2018, pp. 641–644.

[32] S. Bahroun, R. Abed, and E. Zagrouba, "KS-FQA: Keyframe selection based on face quality assessment for efficient face recognition in video," *IET Image Process.*, vol. 15, no. 1, pp. 77–90, Jan. 2021.

[33] X. Jiwei, X. Jiyuan, F. Yi, and C. Dongfang, "Research on video face retrieval method based on deep learning and key frame," in *Proc. 4th Int. Conf. Digit. Signal Process.*, Jun. 2020, pp. 75–80.

[34] Microsoft. (Nov. 2010). *Tracking Users, With Kinect Skeletal Tracking*. [Online]. Available: https://msdn.microsoft.com/en-us/library/hh438998.aspx

[35] Inc. Cadavid Concepts. (Nov. 2014). *Record, Export, Playback, and Analyze Microsoft Kinect Sensor Data Easily Using Nui Capture*. [Online]. Available: https://nuicapture.jaleco.com/

[36] Microsoft. (Nov. 2016). *Kinect, Regarding Depth Mapping*. [Online]. Available: https://social.msdn.microsoft.com/Forums/en-US/fc4f2971-aa1d-4506-ba37-720fd7819822/question-regarding-depth-mapping?forum=kinectv2sdk

[37] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation," in *Proc. Australas. Jt. Conf. AI*. Berlin, Germany: Springer, 2006, pp. 1015–1021.

[38] MATLAB. *Fitcecoc: Fit Multiclass Models for Support Vector Machines*. Accessed: Apr. 2021. [Online]. Available: https://www.mathworks.com/help/stats/fitcecoc.html#bue3ojr-2

[39] MATLAB. *Predict: Svm Classify Observations*. Accessed: Apr. 2021. [Online]. Available: https://in.mathworks.com/help/stats/classificationecoc.predict.html

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

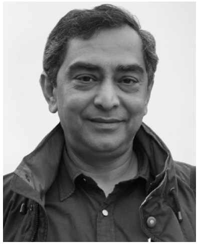
[41] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data," *Data Mining Knowl. Discovery*, vol. 28, no. 1, pp. 92–122, Jan. 2014.

[42] Python and Imbalanced-Learn. *Random Over Sampler Implementation Using a Python Package Called Imbalance-Learn*. Accessed: Apr. 2021. [Online]. Available: https://pypi.org/project/imbalanced-learn/

[43] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized CNN: A unified approach to accelerate and compress convolutional networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4730–4743, Oct. 2018.

HIMADRI BHUYAN received the B.Tech. degree in computer science and engineering from JITM, Paralakhemundi, in 2006, and the master's degree from the Indian institute of Technology Kharagpur (IIT Kharagpur), in 2013, where he is currently pursuing the Ph.D. degree.

From 2006 to 2007, he worked as a BTS Engineer with Telecom Network Solutions (TNS), India. From 2007 to 2008, he worked as an Associate System Engineer with IBM Global Services, Kolkata, India. From 2008 to 2015, he worked as an Assistant Professor with the Computer Science and Engineering Department, NMIET, Bhubaneswar. He is also working on the analysis and interpretation of the Bharatanatyam, an Indian classical dance form, using machine learning. His research interests include image processing, computer vision, machine learning, and digital heritage. He received the Financial Assistantship from MHRD, Government of India.



PARTHA PRATIM DAS received the B.Tech., M.Tech., and Ph.D. degrees from IIT Kharagpur, in 1984, 1985, and 1988, respectively.

From 1988 to 1998, he served as a Faculty Member with the Department of Computer Science and Engineering, IIT Kharagpur, and has guided five Ph.D. students. From 1998 to 2011, he was with Alumnus Software Ltd., and Interra Systems Inc., in senior positions before returning to the Department as a Professor. His current research interests include digital heritage (Indian classical dance), computer vision, image processing, object tracking and interpretation in videos, and medical information processing. He has received several recognitions, including the UNESCO/ROSTSCA Young Scientist, in 1989; the INSA Young Scientist Award, in 1990; the Young Associate-ship of Indian Academy of Sciences, in 1992; the UGC Young Teachers' Career Award, in 1993; the INAE Young Engineer Award, in 1996; the Interra Special (Process) Recognition, in 2009; and the Interra 10 Years' Tenure Plaque, in 2011.

JATINDRA KUMAR DASH received the bachelor's degree in electronics and communication engineering from the Institution of Engineers, India, in 1999, the Master in Engineering degree in computer science and engineering from the Government College of Engineering, Tirunelveli, India, in 2001, and the Ph.D. degree from the Department of Electronics and Electrical Communication Engineering, IIT Kharagpur, India. He worked as a Visiting Researcher with the University of California, Berkeley, USA. He also worked as a Research Consultant with the Sponsored Research and Industrial Consultancy (SRIC), IIT Kharagpur. Before this, he worked as an Assistance Professor and the Head of the Department of Computer Science and Engineering, School of Engineering, Centurion University of Technology and Management, Parlakhemundi. He is currently working as an Associate Professor with the Department of Computer Science and Engineering, SRM University AP, India. He has more than 15 years of teaching and three years of research experience. His research interests include image processing, pattern recognition, texture analysis, and medical imaging.

JAGADEESH KILLI is currently pursuing the five-year dual degree program (B.Tech. and M.Tech.) and the M.Tech. degree in computer science and engineering with the Indian Institute of Technology Kharagpur (IIT Kharagpur). In Summer 2020, he completed an internship at Goldman Sachs, as a Web Developer. He selected as full time analyst for the same firm. He did his M.Tech. project on digital heritage under the guidance of Prof. Partha Pratim Das. His research interests include application of deep learning, machine learning, and systems security.