Journal of Ambient Intelligence and Humanized Computing manuscript No.
(will be inserted by the editor)

Automated Segmentation of *Bharatanatyam* Dance Videos

Machine Learning and Non Machine Learning Approach

Himadri Bhuyan [a] · Partha Pratim Das[a] · Jatindra Kumar Dash[b] · Rohan Majumdar[c]

Received: date / Accepted: date

Abstract Video segmentation is a necessary process to analyze and interpret any video. It uses certain criteria (dependent on the domain) to partition a video such that the consecutive frames that are semantically homogeneous form disjoint sets. Here we attempt to partition the momentarily stationary frames (*key frames*) from the motion frames in the *Bharatanatyam* dance videos. This is achieved by automatic extraction of the *key frame*s. The technique proposed in this paper is simple yet effective as compared to other methods. The proposed *key frame* localization is novel in the domain of dance video analysis. It is distinctive from common *key frame*s detection algorithms as used in other human motion videos. In the basic structure of the dance, momentarily stationary postures (*key frame*s) during performances are often not completely stationary and vary with the form of the dance and the performer. Hence, it is not easy to decide a global threshold (on quantum of motion) that will work across dancers and performances. The earlier approaches try to compute the threshold iteratively, whereas this paper proposes an adaptive technique to compute the threshold from the given video for the key frame detection or localization. The proposed method takes the RGB frames as input, converts them to gray-scale images and applies a modified version of three frame differencing approach via bit-plane extraction. These frames are then used in non-machine learning (non-ML) and machine learning (ML) approaches to segment the given dance videos. In the non-ML approach, an adaptive threshold is devised for segmentation whereas for the later technique, the binary classifier, SVM (Support Vector Machine) is used. The novelty of this paper is to segment the dance videos using ML where the input features are generated by the fusion of modified three frame differencing and bit-plane extraction. Again, devising an adaptive threshold for non-ML approach. Moreover, the feature used in ML proves to be very effective. The paper finally compares the proposed approach with other recent approaches. Eventually the ML technique emerges as a winner with around 90% accuracy in *key frame* detection.

Keywords Key frame, Bharatantyam, Adavu, Adaptive threshold, Machine learning

Himadri Bhuyan
E-mail: himadribhuyan@gmail.com
· Partha Pratim Das
E-mail: ppd@cse.iitkgp.ac.in, partha.p.das@gmail.com
· Jatindra Kumar Dash E-mail: jatinkdash@gmail.com
· Rohan Majumdar E-mail: rhnmjmdr669@gmail.com
[a] Indian Institute of Technology Kharagpur, Kharagpur 721302, West Bengal, India
Mobile: +91-9830030880, +91-9438911655
Tel: +91 (3222) 281998
[b] SRM University-AP, Andhra Pradesh, 522503, India
[c] JNTUH College of Engineering Jagttial, Telangana, 505501, India

1 Introduction

Bharatanatyam[1] is a very popular, oldest Indian Classical Dance (ICD) form. Using this dance form the dancer illustrate the *Hindu* religion themes and spiritual ideas by the help of the elegant footwork, impressive body postures, emotional facial expression and the hand gestures. All these well defined sets of postures, gestures, movements and their transitions are the units of an *Adavu*. It is the basic choreographic units of a dance sequence in *Bharatanatyam* and 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 complex postures, gestures and the attired (Figure 1) of the dancers are vital in the perspective of computer vision and image processing aspects while a computer analysis or/and interpretation of the dance form is the intention.

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

Fig. 1 Showing attired in *Bharatanatyam*

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 *key frame*s (KFs) during the *Key posture*s (KPs). This is a variant of the video segmentation (**?**) problem which can distinguish the momentarily stationary frames (*key frame*s) from the *motion frames* (MFs) in a given *Adavu* video. It may be noted that Key Frame segmentation is a necessary step for variety of other problems in *Bharatanatyam* dance analysis like:

- Distinguishing Key posture and transitions in-between,
- Recognition of *Adavu*s on the basis of the occurrence of KP sequences,
- Distinguishing KF to MF for automated motion and KP annotation,
- Identifying various limb trajectories in motion, and
- Dance transcription etc.

While interests in automated dance analysis is on the rise, most researchers overlook the KF and MF segmentation problem by assuming that the annotated frames are available from the video as input. We review these in Section 2. Incidentally, this segmentation problem is non-trivial with the challenges as follows:

- During the transition from one KP to another, at the starting of the transition there may be very slow motions. That slow motion may be falsely classified as KFs.
- Due to the existence of complex motions and postures, the distinction between KFs and MFs may not be easy.
- The complex dress style (especially below waist) contributes to the occlusion, and sometimes, it drives the non-visibility of foot/leg movements.
- At times, though dancer is in KP position, the movement of the dress materials may be misinterpreted as body movements.
- The non-availability of annotated *Bharatanatyam Adavu*s.

To start with, we create our own data set. The *Adavu*s are recorded using the Microsoft Kinect 1.0 **?** in 30 fps. The work deals with the RGB data streams only. The proposed approach use sequence of converted RGB to gray frames and compute the three frame temporal differencing (a modified version of **?**, **?**) of three consecutive frames in the sliding window fashion, resulting two temporal differentiated images. Now from these two differentiated images, we extract three MSBs and perform the bit-wise EX-OR. Since motion contributes to the intensity values in MSBs, the three MSBs are taken into account to identify the moving pixels. Bitwise Ex-OR gives zero if there exist no change in the pixel intensity and that implies no-motion. Now we do count the number of zero and non-zero pixels in the given frame. Here, the count values (Zero / Non-zero pixels) are associated with the middle frames among three consecutive frames. The paper devices an adaptive threshold on the basis of which, a frame is decided as *key frame* in Non-ML approach. In this due course this paper also tries to address the motion frame (*non-key frame*) detection briefly.

In the ML approach all the values in the frame after Ex-OR operation is taken as feature set and trained by SVM **?** with a predefined level (1: non-motion, 0: motion) and eventually a set of unknown frames are given as test set for prediction. The work also discusses the result in both the approaches (ML & Non-ML) while input frames are background subtracted. The Table- 1 provides the variants of the inter/intra comparative study done in this paper.

Table 1 Comparative study (NA: Not Applicable)

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

The paper makes four major contributions:

- Used bit-plane technique first time on dance data which is found to be very effective
- Adaptive threshold is devised successfully for Non-ML approach
- Explores ML technique
- A comparative performance analysis between contemporary techniques, shown in the Table- 1

The rest of this paper is organized as follows: It starts with the motivation behind the work and the related research that has been done in the same domain which is highlighted in the Section 2. After that it discusses about the data set in the Section 3. In the Section 4 the proposed method is explained. The Section 5 gives a comparative study of the results generated through different approaches. Finally the paper is concluded in the Section 6.

2 Motivation & Related work

The two frames differencing (**??????**) is the obvious approach to detect the motion and non-motion frames, but it requires a strict, static and manually adjustable thresholds which vary with each video. Similarly optical flow technique (**?**) also serves the same purpose but incur heavy computation load.

Much of the literature (**???**) on posture and gesture recognition research in dance do not consider the issue of segmentation and assume that pre-segmented sequences are available for analysis. In **?**, deep learning based *Convolution Neural Network* (CNN) algorithms are proposed to identify body postures and hand gestures in order to express the intended meaning of the ICD performances. A method is proposed in **?** to classify the different ICDs using Pose descriptor of each frame of an ICD video. In **?** develop a system to recognize 3D dance postures. Recently, **?** recognize the posture from 2D images of *Ballet* dance and claim their approach can be implemented in posture recognition from video sequences.

In all these above works, whether it is a problem of recognition or classification, firstly the segmentation of the videos were necessary and are supposed to be addressed.

In contrast to the above, Kahol et al. propose a method for automated segmentation of gesture from dance sequences in **?**. To represent the gesture, the acceleration, velocity and mass of the various body parts and the whole body are considered as well. The authors claim 93.3% accuracy for the detected gesture boundaries. In **??** use musical information for motion structure analysis and gesture segmentation. It is done by detecting the onset from the music signal and tracking the beats. The similar work is done by **?**, where they able to mark the start point of non-motion frames using onset beat detection method. In the same work, image differencing and instantaneous acceleration are used to segment the non-motion frames (*Key frame*). Here they tried manual thresholding for this segmentation and achieved an average accuracy of 74%. But, using on-set beat detection they score 84% in the detection of non-motion frames.

Though there is no much literature found in the segmentation of dance videos but we explore the works in non dance videos which help us to achieve our target. In **?** proposed a new method to detect the moving object using background subtraction and three frame differencing approach. In **?** they also detect the moving object using two frame differencing and bit plane extraction. As per **?**, the three-frame difference is much closer to the real moving target than that of two-frame difference.

From the above discussion, we explore the following shortcomings and the scope with respect to the earlier works which motivates us for the proposed work in this paper.

- Shortcomings
  - No adaptive threshold is developed yet. For each video, manual thresholds are defined for classification.
  - No automation to compute the threshold
  - Time consuming to detect proper threshold
- Scope
  - There is a scope of improvement in the performance
  - Adaptive threshold can be devised
  - Machine learning (ML) approach can be explored

3 Data Sets

Bharatanatyam Adavu videos are captured at 30 fps by Microsoft Kinect 1.0 (**?**) using Nuicapture software (**?**). Every recorded video comprises RGB, depth, skeleton, and audio streams. There exist 15 *Adavu*s of 58 variants. Each variant is performed by three different dancers. Though we recorded all the variants but used only 166 *Adavu*s by ignoring the erroneous recordings. Average number of frames in each video is 700-1000 frames. The data set is shown in the Table- 2. A part of the data set is also made available in **?**.

Table 2 Data Set

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

All these videos are manually annotated. As a sample, the annotation of *Mettu Adavu* is shown in Table- 3. A particular annotation file provides information about the occurrence of *Key frames*[2] (Non-motion frames) and *Non-Key frames* (motion frames). Any frame belongs to *key frame* is called as *Key posture* (KP). Since *Adavu* follows a rhythmic pattern, in between two KPs there exist motion frames. In other words, motion frames are followed by KPs. 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* = *Mettu*4 (M4), Dancer# = 1 (D1), KP# = 21 (P21), Cycle# = 1 (C1: The

[2] In an *Adavu* video, key frames are the momentarily stationary frames.

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.

Table 3 A Sample Annotation File

ID	Start Frame #	End Frame #
M4D1C1B01P21	59	61
M4D1C1B02P22	65	75
M4D1C1B03P12	103	120
M4D1C1B04P09	124	143
....	...	...
M4D1C2B14P26	972	986
M4D1C2B15P27	1006	1027
M4D1C2B16P28	1034	1065

4 Proposed Method

The flow chart of the proposed method is shown in the Figure-2. It takes an *Adavu* video as an input and segment the entire video into motion and non motion frames, but our objective is to extract the *key frame* only. The extraction of *Key frames* (non-motion) in the given *Adavu*, mainly comprises three parts:

- Pre-processing the video
  - Conversion of sequence of RGB frames to gray image
  - Background subtracted (BGS) image sequence
- Feature Extraction
  - Three frame differencing and Bit-plane extraction
  - Average filtering
- Classification
  - Non-ML technique
    - Adaptive threshold computation
    - Classification of KFs (Positive) and MFs (Negative) using adaptive threshold
    - Reduction of False Negative and False Positive through majority voting
  - ML technique: SVM

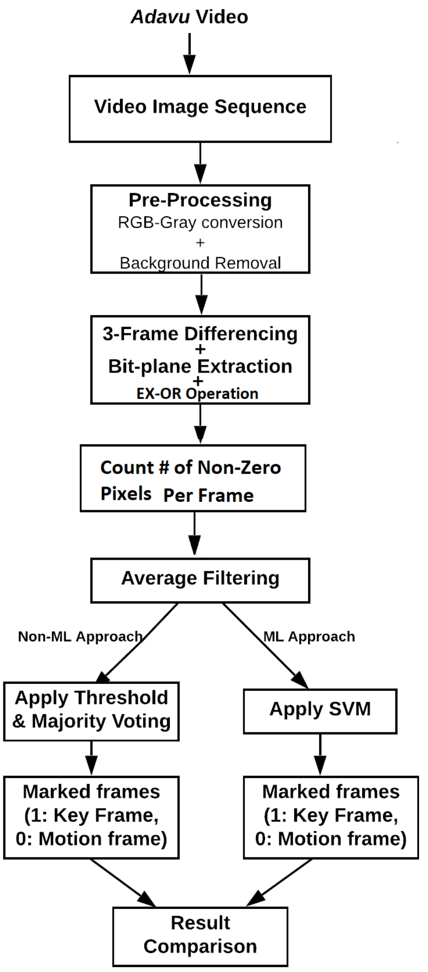


Fig. 2 Flow chart of proposed method

4.1 Pre-processing

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

4.1.1 RGB to Gray Image

The grey level is a key factor to detect the motion. Moreover, it helps in reducing the complexity of the used algorithms.

The color image of size 480×640 is converted to gray image of same size. Here, RGB values are converted to gray scale values of 8-bits (varies from 0 to 255) by forming a weighted sum of the R, G, and B components as shown in the Equation-1.

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

Where $1 \leq i \leq 480$ and $1 \leq j \leq 640$

4.1.2 Background subtracted Image

The paper remove the background in the gray image by retaining only the dancers information using the depth stream information of kinect (**?**) recorded data set. Kinect depth (**?**) information is stored in 16 bit where first three bits denote player index and next 13 bits holds the depth data in mm. Kinect makes use of player index to detect the whole user.

To indicate an user, it marks the pixels with an index 1 to 7. Using this index value, it can be determined 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 an user. RGB camera and the depth camera are not calibrated, so the pixels from each camera are not correspondent. So Kinect provides a mapping technology called depth frame to color frame map (**?**), 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*

4.2 Feature Extraction

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

4.2.1 Three frame differencing and bit-plane extraction

Temporal difference can detect the relative change in the successive frames. As per the literature (**?**), the result of three-frame difference is much closer to the real moving target than that of two-frame difference. In our proposed method, three consecutive frames (F_k, F_{k+1} and F_{k+2}) are considered which results two temporal differentiated images (**?**) $d_\alpha(i,j)$ and $d_{\alpha+1}(i,j)$. It is done in a sliding window fashion. Where

$$d_\alpha(i,j) = |F_{k+1}(i,j) - F_k(i,j)| \tag{2}$$

$$d_{\alpha+1}(i,j) = |F_{k+2}(i,j) - F_{k+1}(i,j)| \tag{3}$$

d_α and $d_{\alpha+1}$ generate pixel wise absolute intensity differences between two successive frames. Now 3-MSB values are extracted, since MSBs contribute most to the intensity values during motion. Next, Ex-OR operation (**?**) is done to compare and compute the pixel difference between bit-planes. It is nothing but the relative change of F_{k+1} with respect to F_k and F_{k+2}. After comparing the bit planes, the higher bit planes are merged together once again to obtain a gray scale image.

$$B_\alpha(i,j) = \sum_{b=5}^{7} 2^b * d_\alpha(i,j)_b \tag{4}$$

$$B_{\alpha+1}(i,j) = \sum_{b=5}^{7} 2^b * d_{\alpha+1}(i,j)_b \tag{5}$$

Where $b =$ Bit position, $d(i,j)_b =$ Intensity at the bit position b.
The steps followed for three frame differencing and the bit-plane extraction is shown in the Figure-3, which is the modified version of **?** and **?**. The motivation behind combining these methods is to exploit the strength of each of those methods. The same is explained in the Algorithm-2 & 3.

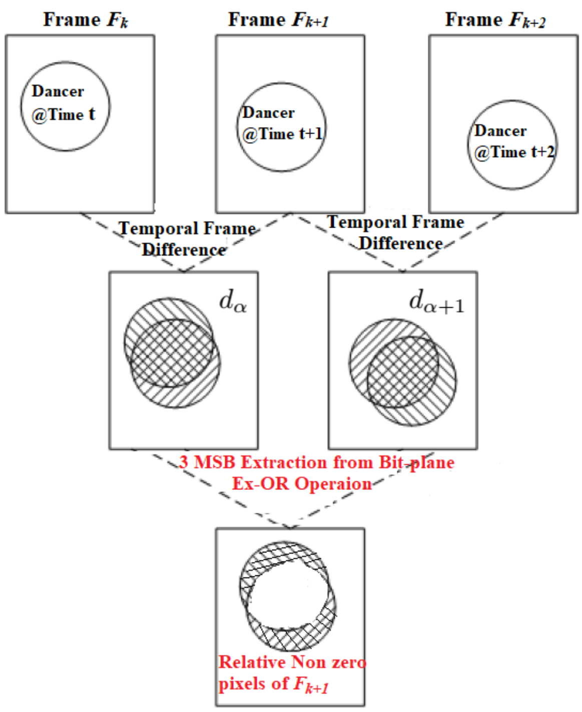


Fig. 3 Three Frame differencing & Bit-plane Extraction (Modified of (**?**))

In **?**, authors use two-frame differencing where as we use three frame differencing. In **?**, they take entire 8-bit pixel values into account and operate bit wise AND on the resultant frames $B_\alpha(i,j)$ & $B_{\alpha+1}(i,j)$, but we only consider 3 MSBs and do the Ex-OR operation and saved as the resultant value of F_{k+1} frame.

$$F_{k+1} = B_\alpha(i,j) \oplus B_{\alpha+1}(i,j) \tag{6}$$

The Figure-4 demonstrate the qualitative result for all three scenarios after frame differencing:

- Considering all 8-bit pixel values and followed by Bit-wise AND
- Considering all 8-bit pixel values and followed by Ex-OR
- Only picking up 3 MSBs and followed by Ex-OR operation

The AND operation doesn't show the relative change in pixel values between two consecutive frames, Figure-4 (a). Where as Figure-4 (b), the EX-OR operation shows the relative change in the position of the dancer. But when we consider only 3-bits for Ex-OR, it indicates only the moving pixels corresponds to the movement of the dancer, which is evident in the Figure-4 (c).

Algorithm 2: Three frame differencing and Bit-plane Extraction, Procedure Name: $ThreeFrameBitplane()$

```
Input: F_k // Where k = 1,2,...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 × 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) = Call_Extract3MSB(d_k(i,j))
            B_{k+1}(i,j) = Call_Extract3MSB(d_{k+1}(i,j))
            ResultF_k(i,j) = B_k(i,j) ⊕ B_{k+1}(i,j)
return ResultF
```

Algorithm 3: Bit-plane Extraction, Procedure Name: $Extract3MSB()$

```
Input: d_k // d_k from Algorithm-2
Output: B_k
B6 = ⌊D_k/32⌋%2
B7 = ⌊D_k/64⌋%2
B8 = ⌊D_k/128⌋%2
B_k = 128*B8 + 64*B7 + 32*B6
return B_k
```

4.2.2 Average Filtering

The output of the Algorithm-2 provides the frames where the number of non-zero pixels are counted. The number of non-zero pixels per frame is shown in the Figure-5 for *Mettu Adavu*, Dancer-3 as a sample. To reduce the error further, average filtering is applied using a mask of size 3 and weight [1, 1, 1]. The Figure-5 shows result after applying the filter. The number of non-zero pixel count per frame is considered as the feature for classification.

4.3 Classification

On the basis of the non-zero pixel counts per frame, a frame can be marked as *key frame* or Non-key frame. It is obvious that for the frame containing small number of non zero pixels, the probability being a *Key frame* is high. Similarly, for the frame containing large number of non zero pixels, the probability of being a Motion Frame is high. The challenge here is how to determine the threshold that takes a decision on a frame being KF or MF. Most of the algorithm compute this threshold heuristically through observation over series of frames. In this work, we present an adaptive method to compute this threshold. The non-zero pixels per frame is considered as 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 where as in ML approach a trained SVM is used.

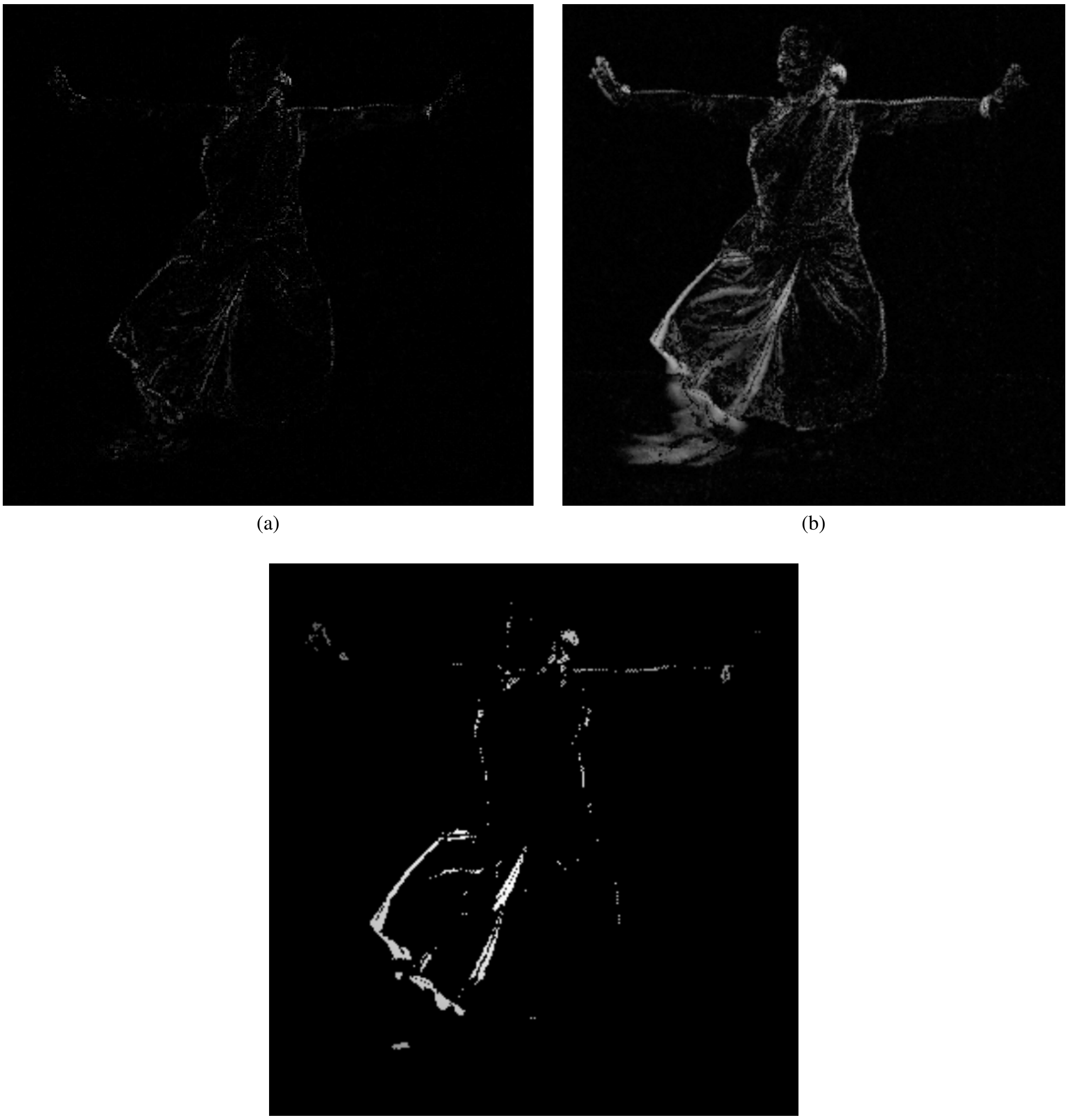
4.3.1 Non-ML Approach

The first step in this approach is to automatize the computation of threshold for each video for the segmentation. To start with, the paper extensively tries out with several thresholds iteratively and computes the accuracy for each. The variation in the accuracy and error of the segmentation by varying the thresholds in the step size of 10 for the data set Dancer-3, *Mettu-1* is shown in the Figure-6. From this extensive study, initially we detect the threshold that attains maximum accuracy. Next, when we do analyze entire range of thresholds and its corresponding accuracy, it is found that there exist a set of consecutive thresholds where no much variation is identified in the accuracy and these are close to maximum accuracy attended threshold. Moreover this maximum accuracy attended threshold is very close to the devised threshold as shown in the Equation-7, Where $ResultF_k$ is generated by Algorithm-2 and $\{k = n-2 | n =$ number of frame in an video$\}$

$$Threshold = \frac{\sum_{i=1}^{k} CountNonZeroPixels(ResultF_k)}{k} \tag{7}$$

The Algorithm-4 shows the use of $Threshold$ to mark a frame as *Key frame*(1) or motion frame (0).

To come to the conclusion for the Equation-7, we analyze across all the *Adavu* videos. To understand, here we use the *Mettu Adavu* as an example. In the Figure-7, which is a zoomed version of Figure-6, the green circle shows the



(a) (b) (c)

Fig. 4 A Comparison: (a) 8-bits AND Vs (b) 8-bits EX-OR Vs (c) 3-bits EX-OR operations after image differencing

Algorithm 4: Marking Key and Non-Key frames

Input: $ResultF_k$ // $ResultF$ from Algorithm-2
Output: $MarkedFrames$
for i = *1 to k* **do**
  $count\_nonzeroPixels = CountNonZeros(ResultF_k)$
  if $count\_nonzeroPixels \leq Threshold$ **then**
    $MarkedFrames[k] = 0$
  else
    $MarkedFrames[k] = 1$
return $MarkedFrames$ // It is an 1-D Array

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 (Using Equation-7) adaptive threshold comes out to be 906. For the threshold = 906, the accuracy = 91.57% which is close to the 92.12%. The Figure-8 shows a comparison study between the maximum possible accuracy Vs the accuracy using the $Threshold$ derived from Equation-7. The maximum accuracy difference found to be 2.94% and 3.56% in Mettu-3 Dancer-2 and Mettu-4 Dancer-2 respectively. Apart from these two videos, in the rest, the differences vary between 0.55% to 1.49%. Here the accuracy is computed using the following Equation-8.

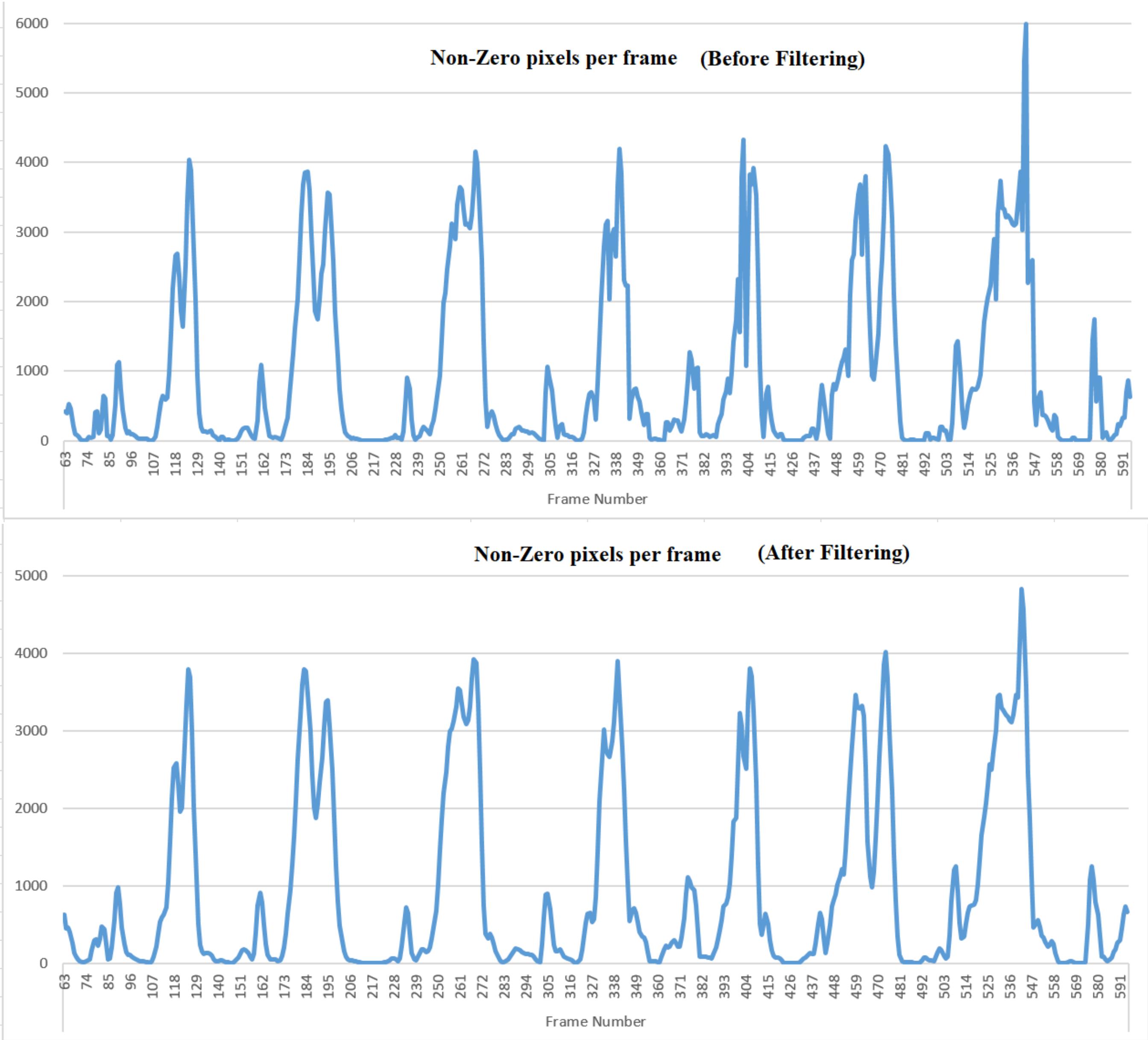


Fig. 5 # Non-zero Pixels per frame in *Mettu-1* Dancer-3: Before Filtering Vs After filtering

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (8)$$

Where TP = True positive (*Key frame*), TN = True Negative (Motion frame), FP = False Positive and FN = False Negative samples.

Majority voting: Reduction of False Positive and False Negative

As we discussed earlier in the Section-3, a sequence of *key frames* follows a sequence of motion frames (*non-key frames*) and vice versa. So in the automatic detection of *key frame* and motion frame, at times there may be a false *key frame* detection in between motion frames or a false motion frame detection in between *key frames*. To deal with such kind of situation we adopted the majority voting technique. We tried two variants of this technique.

- **Approach-1**:
  In this approach, we consider a window acquiring three frames and check if there exist a key frame in between two motion frame and vice versa. The Algorithm-5 achieves this using 3 frame sliding window.
- **Approach-2**:
  In this type we consider a 5 window frame and count the number of zeros and ones. If number of zeros are more than the ones then the center frame in the window is marked as *key frame*. If reverse is the case then the the

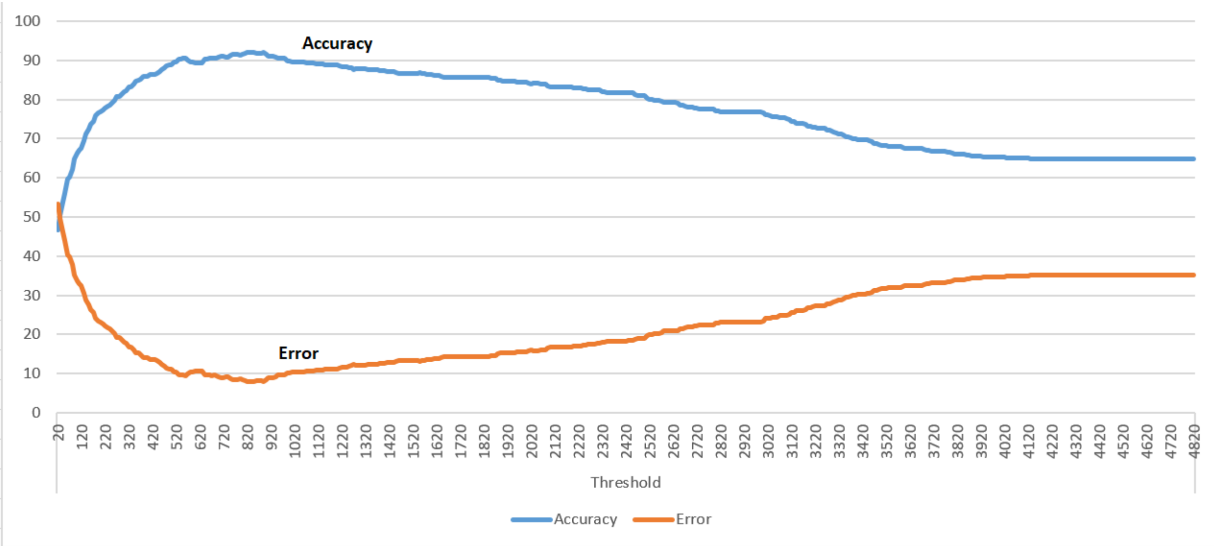


Fig. 6 Accuracy and Error against varying Threshold

Algorithm 5: Majority Voting, Approach-2

Input: $MarkedFrames[1,2...k]$ // *MarkedFrames* from Algorithm-4
Output: $UpdatedMarkedFrame[1,2...k]$
$UpdatedMarkedFrame = MarkedFrames$
for i = *1 to k-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 $UpdatedMarkedFrame$

---

center frame in the window is marked as motion frame. The approach is described in Algorithm-6.

Algorithm 6: Majority Voting, Approach-2

Input: $MarkedFrames[1,2...k]$ // *MarkedFrames* from Algorithm-4
Output: $UpdatedMarkedFrame[1,2...k]$
$UpdatedMarkedFrame = MarkedFrames$
for i = *1 to k-4* **do**
  $W[1,2..5] = MarkedFrame[i,i+1,...i+4]$
  $count0 = countZeros(W)$
  $count1 = 5 - count0$
  $MF = MarkedFrames[i+2]$
  if $count0 > count1\ \&\&\ MF == 1$ **then**
    $UpdatedMarkedFrames[i+2] = 0$
  if $count1 > count0\ \&\&\ MF == 0$ **then**
    $UpdatedMarkedFrames[i+2] = 1$
return $UpdatedMarkedFrame$

Result: Non-ML Approach

We analyze across all the performances and the dancers and come to the conclusion that the Approach-2 majority voting technique performs slightly better than the Approach-1. It is quite evident from the Figure-9 where *Mettu Adavu* is taken as a sample and the comparison is done between the result prior to majority voting (Before tuning) and the result of post majority voting (After tuning). In the majority voting, with window size 5 (Approach 2), the results are either very close to initial results (Before tuning) or slightly higher. Here results are computed using Equation- 9 unlike Equation- 8.

The Table-4 shows the average accuracy of each variant of *Adavu*. In each variant the accuracy of *key frame* is shown along with the change in the accuracy after applying majority voting (Approach-2). The precision of *key frames* ($PrecisionKF$) are calculated using the the Equation-9. The recall and F1-Score are computed by Equation-, 10 and 11.

$$PrecisionKF = \frac{TP}{TP+FP} \tag{9}$$

$$RecallKF = \frac{TP}{TP+FN} \tag{10}$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{11}$$

Precision means the results which are relevant. The recall refers to the percentage of total relevant results correctly classified by the algorithm. In the other hand, F1 score takes both false positives and false negatives into account. In our case the cost of false positives and false negatives are very

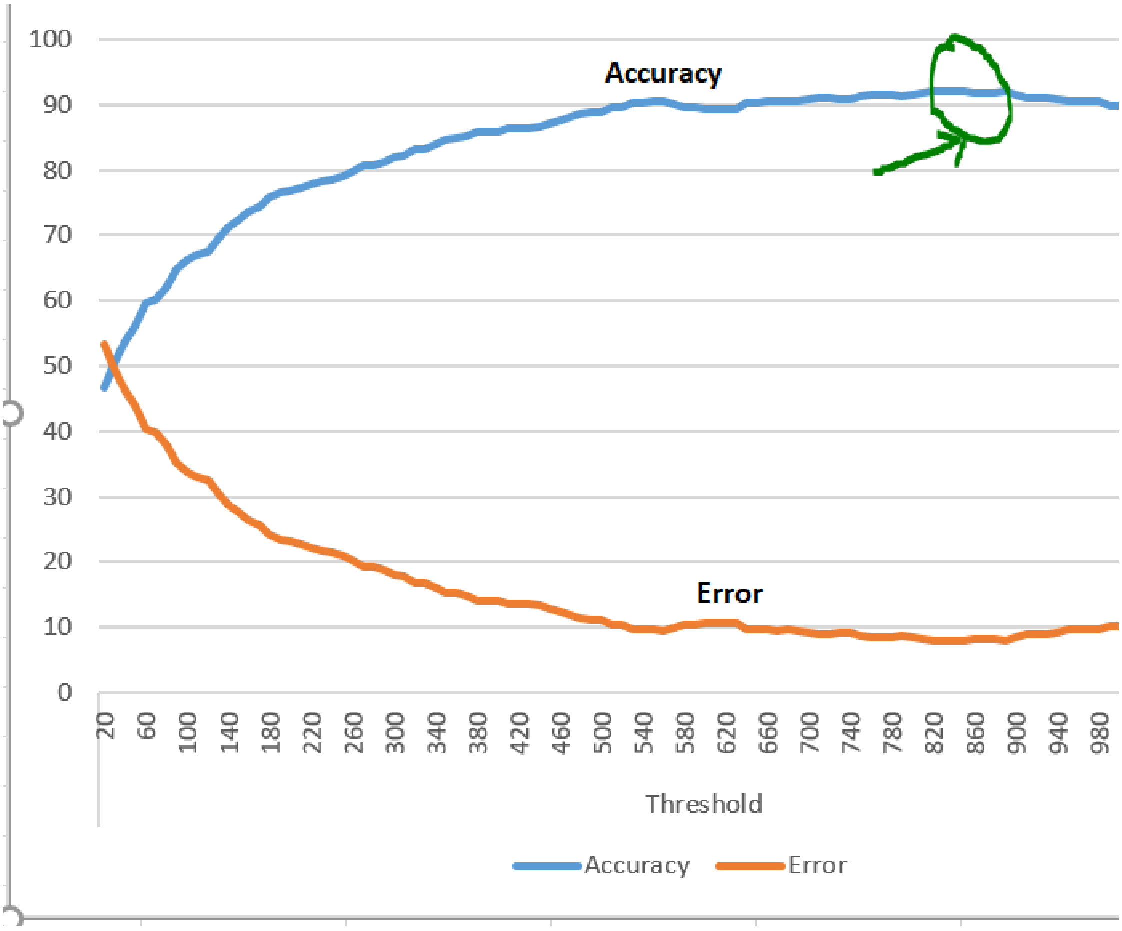


Fig. 7 Marked steady accuracy

different, because the *Key frames* is our point of interest where as the motion frame detection is secondary for us. So, it's better to look at both precision and recall in form of F1 Score and take the decision accordingly. F1 Score uses harmonic mean. So when precision and recall both gives good result, the F1 scores yield to be good. All the performances are calculated in percentage and reported in Table-4.

Table 4 Non-ML Approach Result (in %) of *Key frame* (KF) Extraction: Without Background subtraction (WBGS) Vs Background subtraction (BGS)

Adavus	BGS (KF)			WBGS (KF)		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Joining	83.694	64.461	72.829	79.941	64.030	71.106
Kartari	98.539	49.080	65.524	97.095	50.193	66.176
Nattal	92.162	67.115	77.669	91.267	65.961	76.577
Tattal	85.614	57.071	68.488	84.872	58.465	69.236
Mandi	89.549	74.748	81.482	89.869	74.105	81.229
Mettu	93.165	87.732	90.367	93.226	86.747	89.870
Natta	85.519	82.562	84.014	85.105	82.298	83.678
Paikal	92.141	57.69	70.955	89.649	44.742	61.054
Pakka	77.847	43.328	55.671	74.536	41.240	56.493
Sarika	72.474	68.740	70.558	72.394	68.522	70.405
Sarikkal	89.247	65.432	75.506	90.490	63.829	74.856
Tatta	81.499	87.544	84.413	78.261	90.160	83.790
Tei-Dhatta	84.199	46.821	60.178	83.225	49.712	62.245
Tirmana	77.571	56.484	65.369	77.341	56.340	65.191
Utsanga	67.631	49.811	57.369	70.158	49.557	58.085
Average	**84.723**	**63.908**	**72.026**	83.829	63.044	70.978

4.3.2 Machine Learning Approach

The best part of the supervised machine learning approach is, it learns the threshold for classification problem to solve from the given training data. In this paper we use SVM (**?**) that learns to classifying the incoming video sequence into *Key Frames* and Motion Frames. SVM is a supervised binary classifier. Like any ML model, the SVM also requires the features of each frame to train the model and test as well. So our job is to identify the proper feature set for the data set. The data set means the set of available *Key frames* and *non-key frames*. In this approach the features are the resultant pixel values generate for each frame by the Algorithm-2. The dimension of the feature is 307,200 since the size of the each frame = 480×640. It says, for each frame we provide 307200 values to train the SVM and test as well. During training the SVM, labeled (1: *key frame*, 0: *non-key frame*) features associated with each frame are provided. While testing, we do provide the unlabeled feature sets to our trained SVM model and the model predict whether a

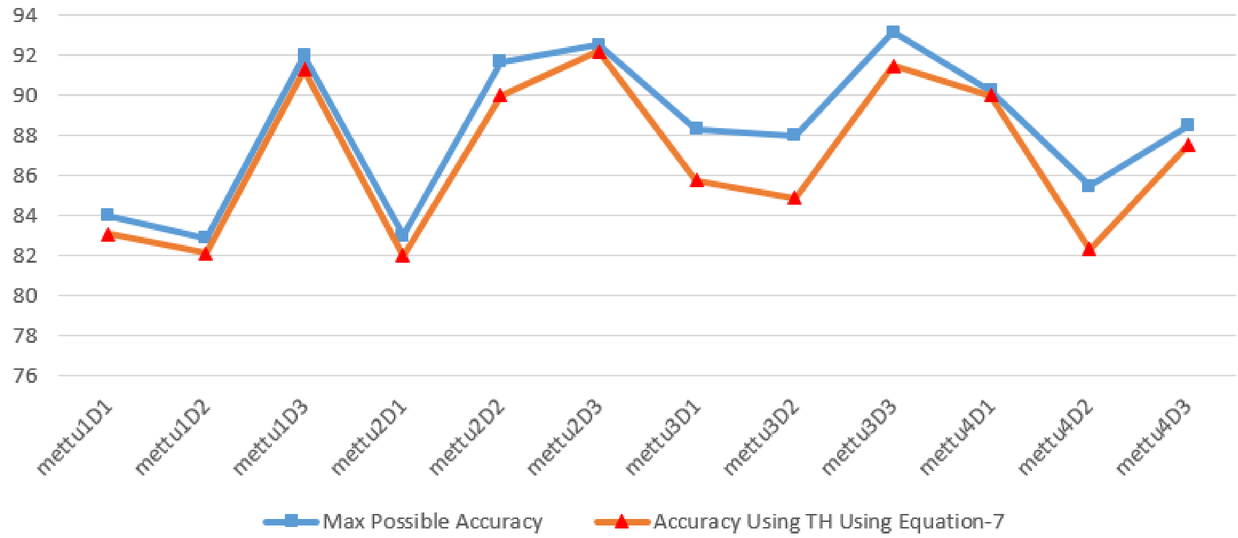


Fig. 8 Maximum Attended accuracy Vs Accuracy using adaptive threshold of *Mettu Adavu*s

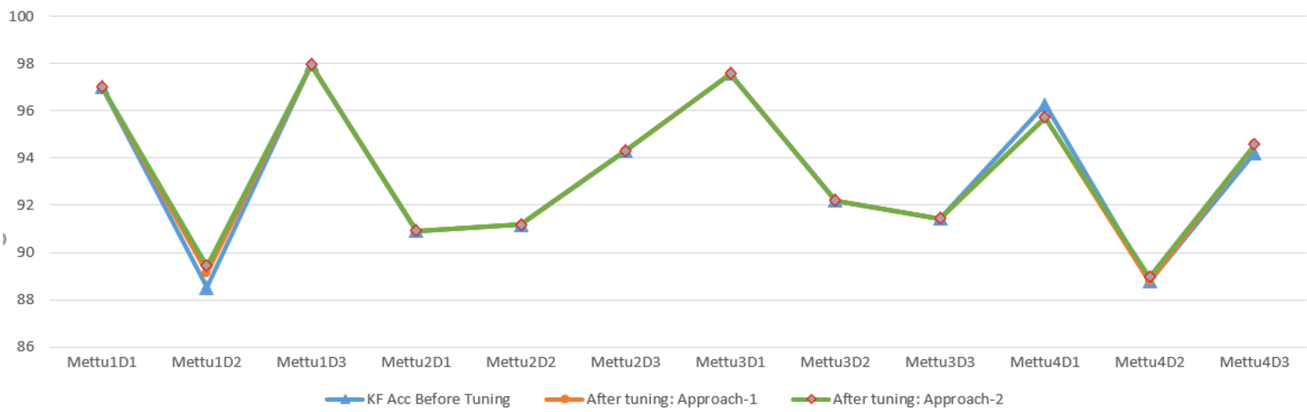


Fig. 9 Result comparison of *Mettu Adavu*: Before Tuning Vs After Tuning

given feature set belongs to *key frame* or *non-key frame*. The training and testing process is shown in the Figure-10.

The prediction accuracy of the segmentation is shown in the Table-5 Like non-ML approach here also we present the result of *key frame* detection in form of precision, recall and F1 Score using the Equation-9 to 11.

Table 5 ML Approach, Result (in %) of *Key frame* (KF) Extraction: Without Background subtraction (WBGS) Vs Background subtraction (BGS)

Adavus	BGS (KF)			WBGS (KF)		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Joining	92.846	74.573	82.712	95.409	69.445	80.382
Kartari	95.118	68.297	79.506	88.424	47.895	62.135
Nattal	95.054	78.180	85.795	80.018	74.276	77.040
Tattal	92.092	85.034	88.423	87.673	84.359	85.984
Mandi	90.614	85.495	87.980	86.380	80.196	83.174
Mettu	95.670	85.416	90.253	97.171	90.656	93.800
Natta	94.485	86.256	90.183	94.188	84.810	89.253
Paikal	96.65	41.55	58.116	96.083	44.74	61.052
Pakka	74.266	64.458	69.015	68.268	58.930	63.256
Sarika	86.442	76.019	80.897	89.237	70.170	78.563
Sarikkal	91.447	67.812	77.876	95.682	69.189	80.307
Tatta	98.813	97.178	97.989	97.740	96.104	96.915
Tei-Dhatta	89.971	82.529	86.089	91.373	68.752	78.464
Tirmana	73.718	71.955	72.826	76.225	66.464	71.011
Utsanga	81.871	60.897	69.844	71.053	50.020	58.709
Average	**89.937**	**75.043**	**81.167**	87.662	70.440	77.336

5 Result Analysis

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

5.1 Performance Analysis: WBGS Vs BGS

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

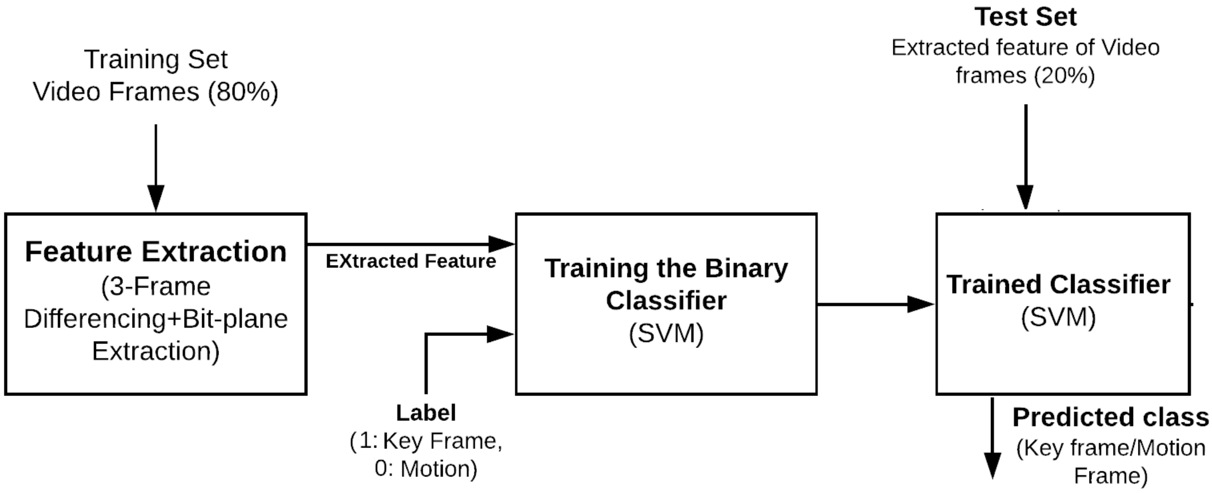


Fig. 10 Training and Test Flow to classify *Key frame* & Motion frame using SVM

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

- Non-ML without Background subtracted data (Non-ML WBGS)
- Non-ML with Background subtracted data (Non-ML BGS)
- ML without Background subtracted data (ML WBGS)
- ML with Background subtracted data (ML BGS)

When we compare Non-ML WBGS and Non-ML BGS, we observe background subtracted data sets are identifying *key frame* more correctly then the WBGS data. The comparison is shown in the Figure-11 and in the Table-4. All most all the *Adavu* videos perform slightly better when background is subtracted from the scene. For BGS the average accuracy is 84.72% and for WBGS it is 83.82%. Most importantly, average *F1 Score* in both the cases is above 70% which imply our approach is a balanced one. Here *F1 Score* plays an important role along with the precision because, the weightage of *Key frame*s are more as compare to motion frames, since our prime objective is to identify the *key frame*s.

While comparing ML WBGS with BGS, we observe ML BGS approach performs much better. The comparison is visible in the Figure-12 and in the Table-5 as well. But, in between Non-ML BGS and ML BGS, In most of the videos the performance of ML BGS is above 90%. As whole preforms better. The comparison is shown in the Figure-13. In few *Adavu*s (*Karatri, Pakka, Tirmana*), Non-ML approach performs slightly better as compare to ML, the reason may be, very slow motioned frame some time detected as motion in ML where as in Non-ML that is being excluded by threshold. In the other hand *F1 Score* improved by 9% in ML approach which is a good sign. So ML with BGS would be our final choice.

5.2 A comparison with contemporary approaches

From the literature we identify some of the contemporary approaches which can be compared with our current approach. In **?** authors tried two image differencing technique to detect *key frame*s on *Bharatanatyam* dance, which is found to be less effective than our Non-ML approach where we try with background subtraction. The comparison is shown in the Figure-14. Along with each video it's variants are mentioned within the braces to provide a glimpse of data set used by **?**.

In some cases the current approach try more variants and more videos (Table- 2) as well. In this comparison, we identify three videos; Sarrika, Tatta & Utsang which perform better in **?** than the current one. But, when we compare our ML approach with the **?**, the current approach performs much better. However in ML approach all most all the videos performance is close to or above 90% except Pakka, Tirmana & Utsang which are in between 70-80%, but still those are higher than the result of **?**. It is quite evident in the Figure- 15.

Though the techniques in the literatures **?????** are not applied to dance data, but they apply simple two image differencing technique and do the Bit-wise AND operation by taking all 8-bit plane into consideration to detect the motion in the videos. In curiosity, we compare their technique in our data set and compare the result of *key frame* detection. The comparison is shown in the Figure-16. The accuracy difference is 8%-20% except the last two *Adavu*s (Tirmana & Utsang) which are very close to the current accuracy, but still the current approach performs better.

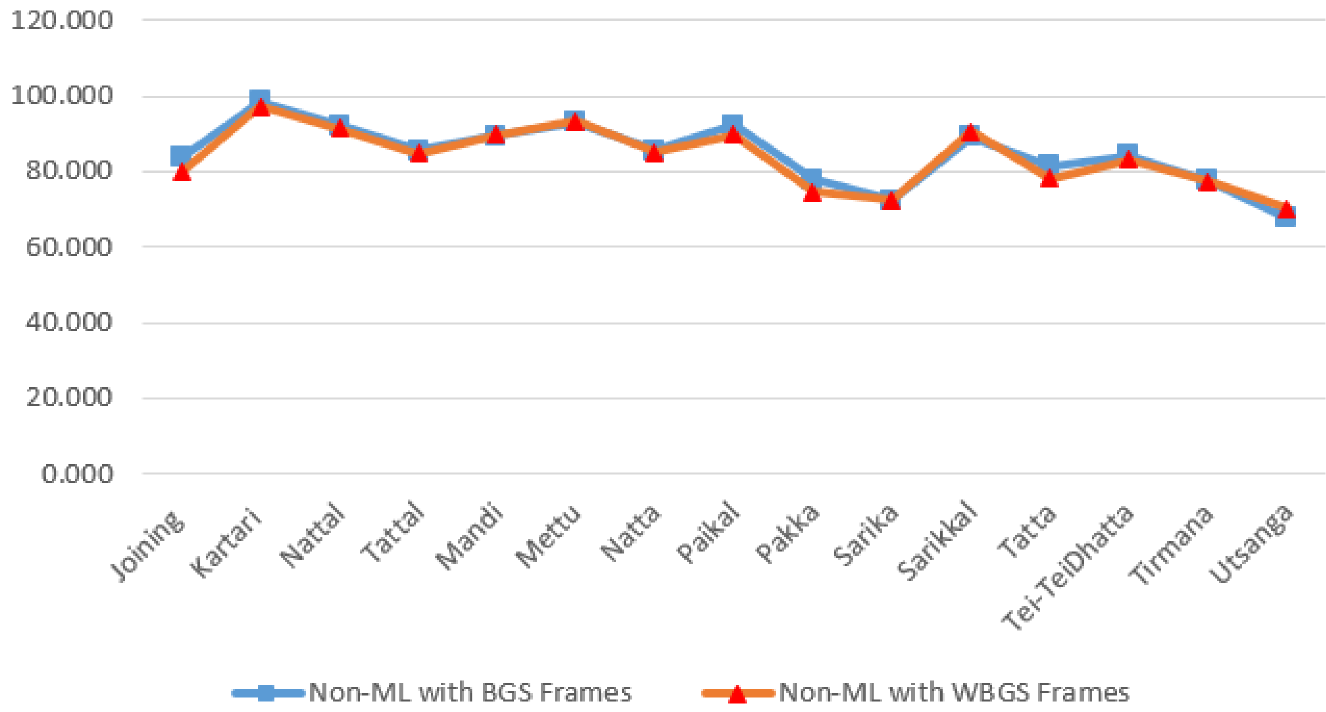


Fig. 11 Non-ML comparison WBGS Vs BGS

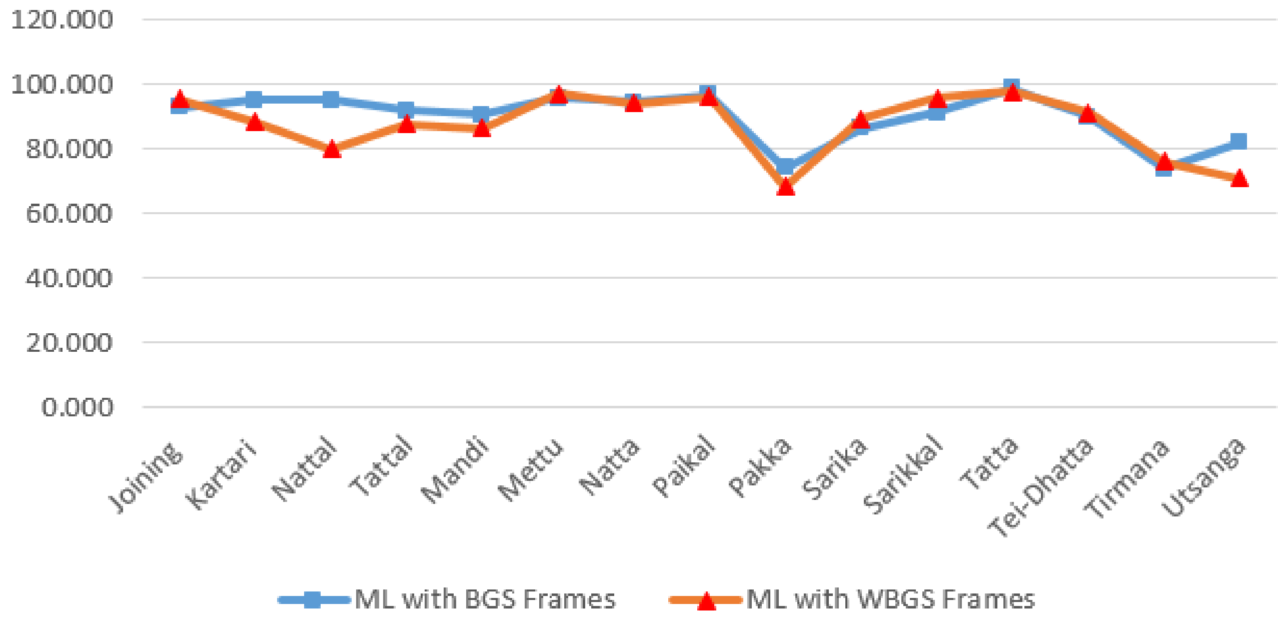


Fig. 12 ML Comparison WBGS Vs BGS

6 Conclusions

The proposed method can extract *key frames* in the *Bharatanatyam* dance videos successfully. The paper also discusses the contemporary approaches (**??????**) and compares with the proposed approach. The earlier approaches attempt to compute the threshold iteratively which is time consuming, since for each video, a manual threshold is defined to distinguish *key frame* and motion frame. There was no mechanism to compute an adaptive threshold.

The earlier approaches attempted to segment motion and non-motion frames in general human activities (walking, running etc). In accordance to the dance structure of any ICD, it is very difficult to choose a global threshold to segment

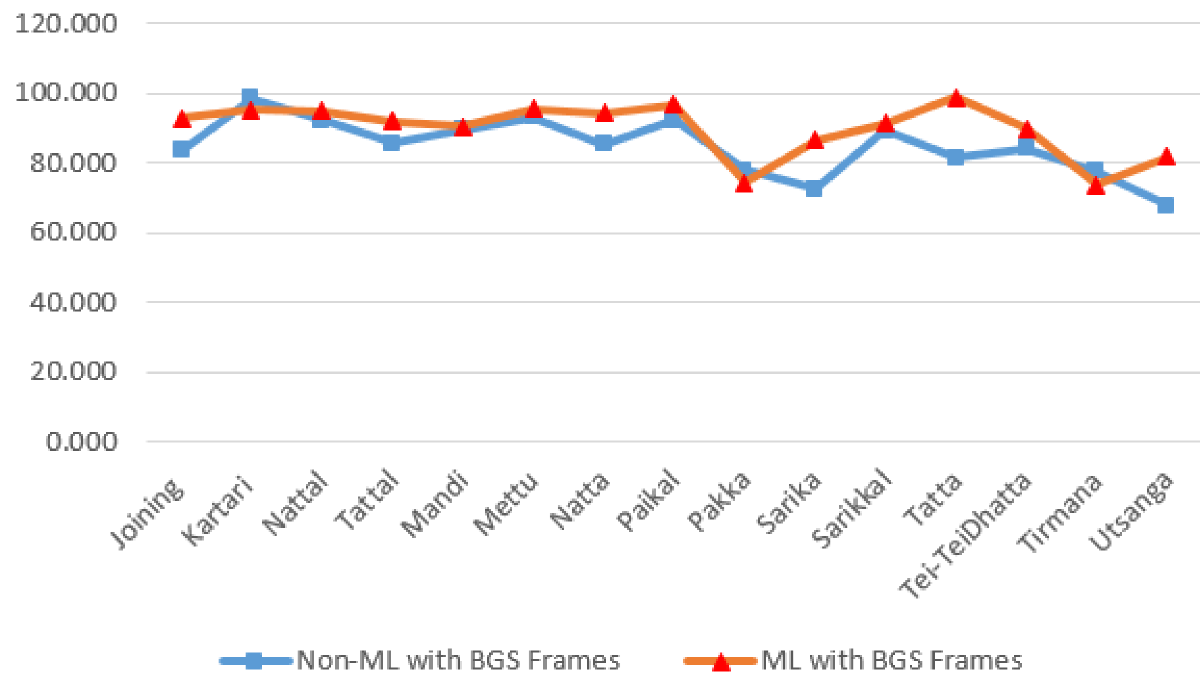


Fig. 13 Comparison Non-ML BGS Vs ML BGS

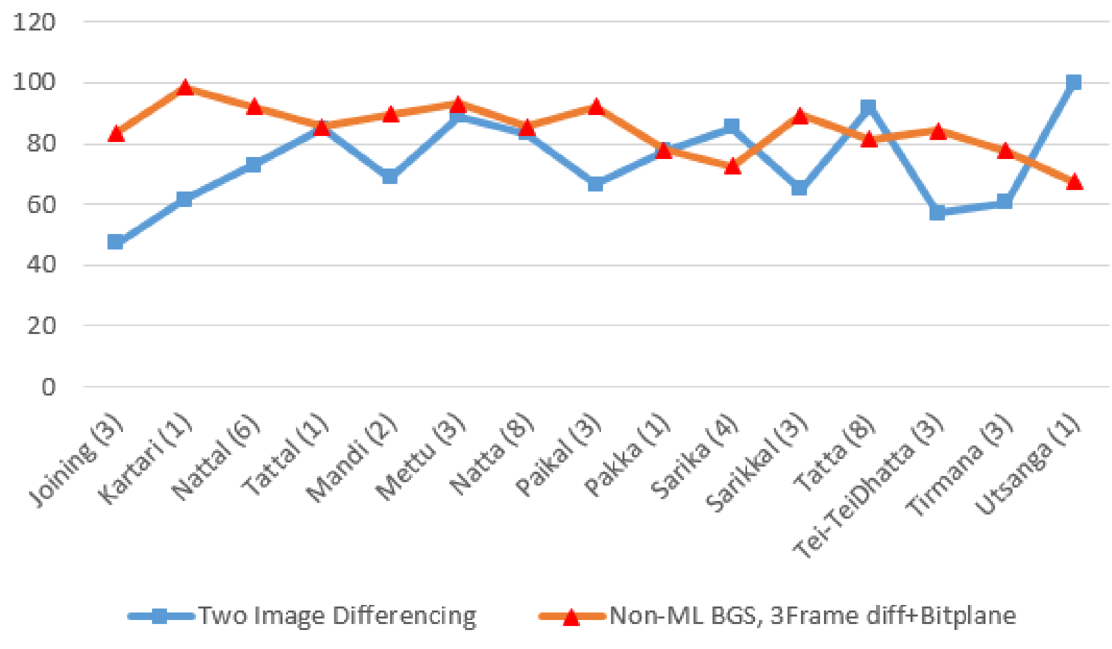


Fig. 14 Comparison **?** Vs Non-ML with BGS

the video. This paper used three frame differencing and bit-plane technique to extract *key frame*s which is novel in the perspective of dance data set. Along with this, an adaptive threshold is devised successfully for Non-ML approach and an ML approach is also explored.

In contrast to the traditional methods(**???????**), combination of the three frame differencing (a modified version of **?**) and bit-plane extraction technique is able to overcome the drawbacks. The paper successfully identifies an adaptive threshold with less computation for non-ML technique and a well-defined ML classifier, SVM (**?**), independent of thresholds to segment the videos. On the basis of the performance, the proposed approach does not lag behind the earlier approaches. Our ML approach gives consistently good result – more than 90% precision in most of the *Adavu* videos. Whereas the performance of non-ML approach is also not

Fig. 15 Comparison (?) Vs ML with BGS

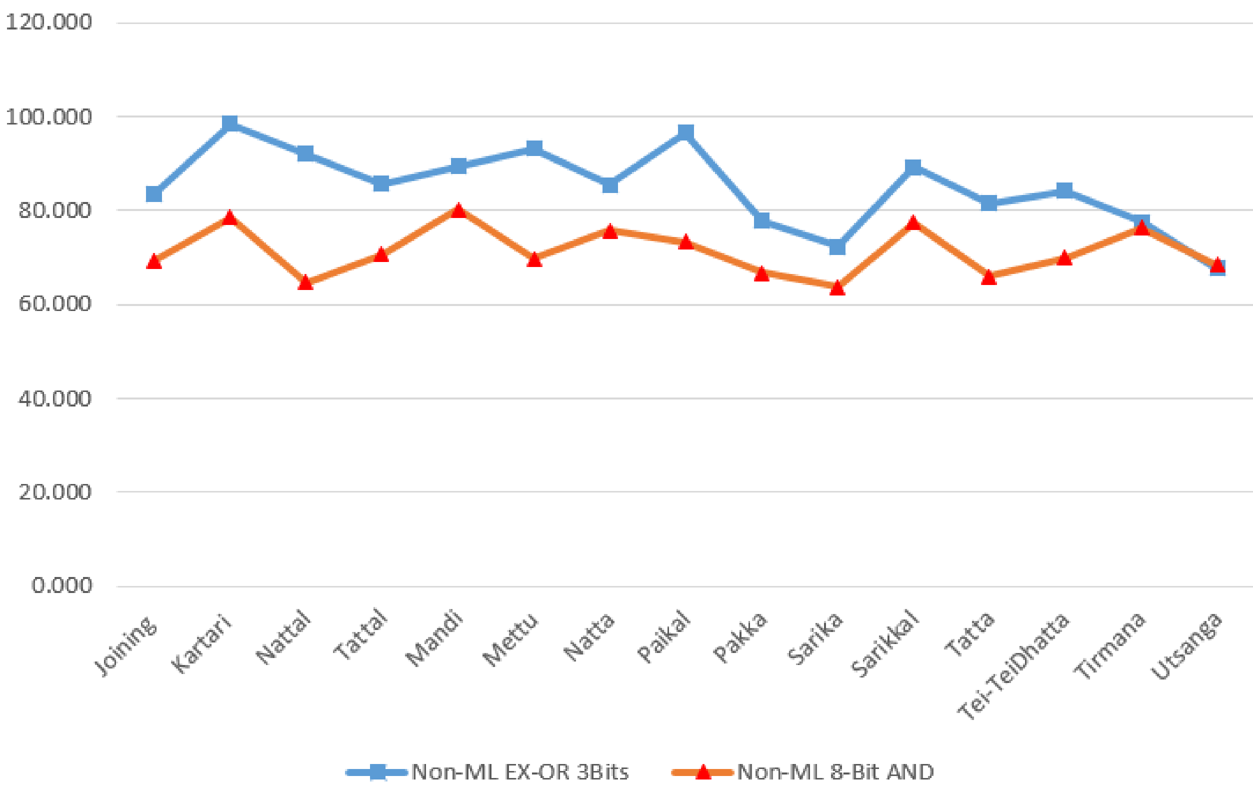


Fig. 16 Result Comparison: Current Approach, Non-ML 3-Bit Ex-OR Vs 8-bit AND

bad (above 85%). Hence, both approaches (ML/Non-ML) outperforms the earlier ones.

The proposed approach yet to be tested in non dance videos. It will be an important tool to segment the videos of any Indian Classical Dance form. Further, in ML approach we use a feature set of 307, 200 dimensions which affects the time complexity of SVM. This can be significantly reduced by using 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 proposed approach to explore the performances.