IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 24, 2017, accepted September 28, 2017, date of publication October 4, 2017, date of current version October 25, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2759225

Human Action Recognition Using Adaptive Local Motion Descriptor in Spark

MD AZHER UDDIN, JOOLEKHA BIBI JOOLEE, AFTAB ALAM, AND YOUNG-KOO LEE
Department of Computer Science and Engineering, Kyung Hee University, Yongin 1732, South Korea

Corresponding author: Young-Koo Lee (yklee@khu.ac.kr)

This work was supported by the Institute for Information and Communications Technology Promotion Grant through the Korea Government (MSIT) under Grant R7120-17-1007 (SIAT CCTV Cloud Platform).

ABSTRACT Human action recognition plays a significant part in the computer vision and multimedia research society due to its numerous applications. However, despite different approaches proposed to address this problem, some issues regarding the robustness and efficiency of the action recognition still need to be solved. Moreover, due to the speedy development of multimedia applications from numerous origins, e.g., CCTV or video surveillance, there is an increasing demand for parallel processing of the large-scale video data. In this paper, we introduce a novel approach to recognize the human actions. First, we explore Apache spark with in-memory computing, to resolve the task of human action recognition in the distributed environment. Secondly, we introduce a novel feature descriptor, namely, adaptive local motion descriptor (ALMD) by considering motion and appearance, which is an extension of local ternary pattern used for static texture analysis, and ALMD also generate persistent codes to describe the local-textures. Finally, the spark machine learning library random forest is employed to recognize the human actions. Experimental results show the superiority of the proposed approach over other state-of-the-arts.

INDEX TERMS Human action recognition, spark, adaptive local motion descriptor, spark MLlib, random forest.

I. INTRODUCTION

Recently, due to the rapid advancement of the Internet, social media video services and intelligent CCTV for video surveillance system, the multimedia data such as video is increasing rapidly. Moreover, understanding video context and identifying video types has an important significance in the management of massive video data. In order to manage these videos and provide important services to the users, it is necessary to understand the human activities from the videos automatically. There are many applications, which focus on the action recognition, such as crowd behavior prediction, video surveillance, human-machine interaction, and sports game analysis.

Human action recognition is an intricate field since static object characteristics, time, and motion features have to be considered. Moreover, due to the environmental variations including different viewpoints, moving backgrounds and large intra-class variations of different actions, the recognition of human actions is even more difficult.

On the other hand, with the exponential growth of the multimedia data and videos from the different origins e.g. CCTVs, it increases the demand of distributed computing to provide the services efficiently. For example, in every minute almost 300 hours of video are uploaded [35]. Existing video processing system uses the Hadoop platform [13]–[16] in order to perform the distributed computing, however, it shows low efficiency in the iterative computation, which is essential in the machine learning. Moreover, it does not support the real time computation.

In this work, we propose a novel approach to classify the human actions on Apache Spark. Experimental results in [20] shows that Apache Spark outperforms Hadoop. In this work, all the videos are stored in the Hadoop distributed file system (HDFS) and dataset is loaded into a Spark cluster, which is represented by resilient distributed datasets (RDDs) [21]. In RDD partition, we perform frame extraction from video data, background subtraction and finally, Adaptive Local Motion Descriptor (ALMD) is introduced to extract the motion feature. These RDD operations are done in parallel using each worker node and provides in-memory based computations. Adaptive Local Motion Descriptor is inspired from Local Binary Pattern (LBP) [25], and Local Ternary Pattern [26] which are only able to extract the static texture information. But, our proposed descriptor

can extract both the texture and motion information by taking two consecutive frames (i.e. previous frame and next frame) to calculate the motion feature. Moreover, it can produce consistent patterns in uniform and near-uniform regions. Volume Local Binary Patterns (VLBP) in [6] also extract the dynamic information from the videos. However, it considers co-occurrences of neighboring pixels in successive frames of a volume at the similar time, which produces large feature information when the number of neighboring pixels applied is extended. Finally, in this paper, we have employed Spark MLlib (Machine Learning Library) Random Forest to classify the human actions in the distributed environment. In order to assess the performance of our work, four benchmark datasets are used, including the KTH dataset [10], UCF Sports action dataset [11], [12], UT-Interaction dataset [31] and UCF-50 dataset [32]. The key contribution of this paper is summarized as follows:

- In this paper, we introduce a novel approach for the human action recognition in spark.
- We propose a new dynamic descriptor, namely Adaptive Local Motion Descriptor (ALMD) that not only describe the texture but also motion. It also produces consistent patterns against intensity fluctuation.
- To the best of our knowledge, this is the first effort to classify the human actions in distributed environment using Apache spark and Spark MLlib Random Forest.
- Experimental result shows our proposed approach can significantly increase the accuracy over state of the art works.

The remaining paper is organized as follows. The following part reviews related literature. In Sections III and IV, we discuss the Architectural overview and proposed framework. Datasets and experimental results are discussed in Section V. Lastly, conclusions are drawn in Section VI.

II. RELATED WORK

Many researches have been already done for human action recognition. This section presents some prior related works for human action recognition and distributed framework for analyzing multimedia data, which is significant to proposed framework.

Now-a-days, in the area of computer vision, researcher started using distributed environment for performing analytical studies in multimedia data. Zhang *et al.* [13] proposed a cloud-based framework that can deal with the intelligent investigation and storage for video data. They used both the Hadoop and Storm platform for the batch processing and real-time processing respectively. However, their work only focuses on video processing rather than any particular application domain. Kim *et al.* [14] presented a distributed video transcoding system utilizing Hadoop that converts numerous video codec files into the MPEG-4 video files. They implemented MapReduce framework and also applied the multimedia processing library Xuggler. Tan and Chen [15] proposed a method for the parallel video processing with Hadoop based clusters including MapReduce. They showed that their method is capable to cope with large scale video and significantly improved the performance. They implemented the face detection, motion detection and tracking procedure as case study to demonstrate their implementation on the Apache Hadoop cluster. Liu *et al.* [16] proposed a distributed environment for the video management applying Hadoop. They also evaluated their work for the simultaneous I/O performance, file uploading and downloading. Wang *et al.* [17] investigated the MapReduce framework on the large scale video data including video event detection, video retrieval and image classification applications. However, most of the related work for analyzing multimedia data in a distributed framework focuses on Hadoop platform and no existing work used Spark Machine learning library (Spark MLlib) [22] for the classification of human actions.

Wang *et al.* [1] proposed a method to illustrate videos by dense trajectories and also sampled the dense points from every frame and traced those points using displacement information. They also presented a new descriptor using the motion boundary histograms. Lan *et al.* [2] introduced hierarchical mid-level action elements to recognize the human actions. They proposed an unsupervised approach to produce this representation from videos. In [3], a video representation to recognize the actions using dense trajectories along with motion boundary information was introduced by the authors. They extracted the local motion feature of the video. Yang *et al.* [4] proposed a novel salient foreground trajectory (SFT) extraction method based on saliency detection to recognize the human actions. They also introduced low-rank matrix recovery to learn the discriminating features from complex video context. Baumann *et al.* [6] applied Volume Local Binary Patterns (VLBP) for human action recognition. They proposed various computation approaches for volume local binary patterns. However, in VLBP the size of feature vector is increased when the number of neighboring points applied is extended. Zhao and Pietikäinen [8] introduced dynamic texture classification using Local Binary Pattern from three orthogonal planes (LBP-TOP). But, there are redundant feature inside the overlapping orthogonal planes, which increases the computational complexity. Mattivi and Shao [7] presented extended LBP-TOP to recognize the human actions. In [9], Local Ternary Pattern from three orthogonal planes was presented to classify the human actions. In LTP-TOP features extracted from the x-t and y-t planes along with the information extracted using the standard LTP (Local Ternary Pattern). Chen *et al.* [5], the authors proposed a novel feature extraction method, namely Histograms of Oriented Gradients from three orthogonal planes (HOG-TOP) to get the dynamic information from the video data. Xiao and Song [34] presented hierarchical dynamic Bayesian network to identify the human actions.

III. ARCHITECTURAL OVERVIEW

Hadoop is an open source programming framework based on MapReduce. It provides automatic parallel, distributed

computing, task management and fault-tolerant mechanism. Hadoop consists of three parts: MapReduce, HDFS and YARN. MapReduce is parallel computation model, HDFS (Hadoop Distributed File System) is aimed to support large datasets on commodity hardware and YARN is used to manage the resources and schedule jobs. However, Hadoop is based on the acyclic data flow model. Due to this, it shows poor performance on iterative tasks which are common when handling the multimedia data. On the other hand, Spark [20] is a distributed framework which is designed for low latency and also support the iterative computation of the data. Spark provides a fault tolerant and efficient memory abstraction mechanism using Resilient Distributed dataset (RDD) [21] that are stored in memory. An RDD in Spark is fundamentally an unchangeable distributed collection of objects. In order to compute a result, in Spark, all work is expressed as either making new RDDs, altering existing RDDs, or calling processes on RDDs. Every RDD is divided into multiple partitions, which can be computed on several nodes of the cluster. It provides in-memory processing execution, so it stores the state of memory as an object through the jobs. Moreover, Spark can outperform Hadoop by 10x in iterative machine learning tasks [20]. Spark MLlib [22] is an open-source distributed machine learning library, which contains fast and scalable employments of standard approaches, including classification, clustering, regression, and dimensionality reduction. Fig. 1 shows the architecture for distributed environment to recognize the human actions. In our work, we have used the Apache Spark for distributed computing and Spark MLlib for classifying the human actions. On the other hand, Yarn is employed for managing the resources and HDFS is used for storing the video data and motion features from video data.

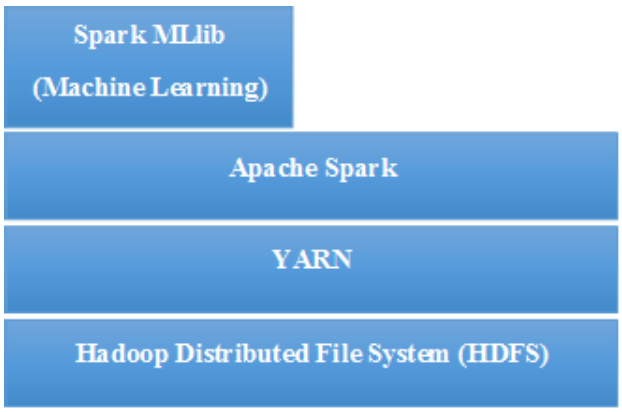

FIGURE 1. Distributed environment architecture to recognize the human actions.

IV. PROPOSED FRAMEWORK

In this section, we explain the proposed method which recognizes the human actions from videos. Fig. 2 shows the proposed framework for human action recognition in the distributed environment. In this paper, a novel approach for human action recognition is presented utilizing adaptive

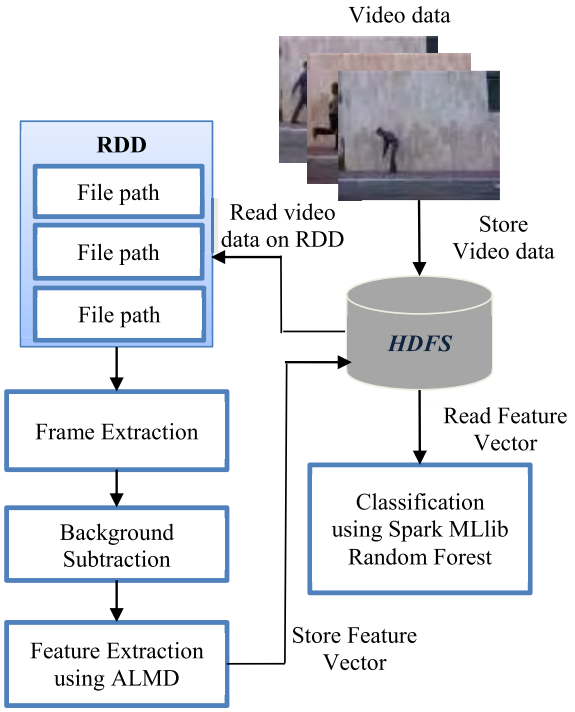

FIGURE 2. Proposed framework for Human action Recognition.

local motion descriptor in spark. Here, motion features are extracted by extending local ternary pattern descriptor and then Spark MLlib Random Forest classifier is employed to recognize the human actions.

A. PREPROCESSING

Preprocessing step includes frame extraction from video, frame conversion from RGB color model to gray level, frame resize and background subtraction. At first, all the videos are saved in the Hadoop Distributed File System (HDFS), then the dataset is loaded into a Spark cluster. Partitions of this dataset are cached into worker nodes. The information of all the loaded data partitions is represented by an RDD. RDD operations are accomplished in parallel using each worker node. In each partition, frames are extracted from the videos and then these frames are resized to 854 × 480 for the KTH dataset, 720 × 404 for the UCF sports action dataset, 720 × 480 for the UT-Interaction dataset, and 320 × 240 for the UCF-50 dataset. After that, frame conversion is employed to change the RGB color to gray scale. Finally, background subtraction is done, which is applied to distinguish foreground object from background part. Here the foreground object is human. In, background subtraction frames are subtracted from their background part. To subtract background from a recent frame Gaussian probability distribution function is employed [23].

$$P(C(x,y)) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(C(x,y)-B_i(x,y))}{2\sigma^2}\right) \quad (1)$$

Where, the probability of belonging background of a pixel in (x, y) position of the current frame is denoted by P(C(x, y)), N is the number of background frames, B_i (x, y) and C(x, y) is the intensity of pixel in (x, y) point of the i^{th} background frame and the current frame respectively. Fig. 3 demonstrates an example of background subtraction.

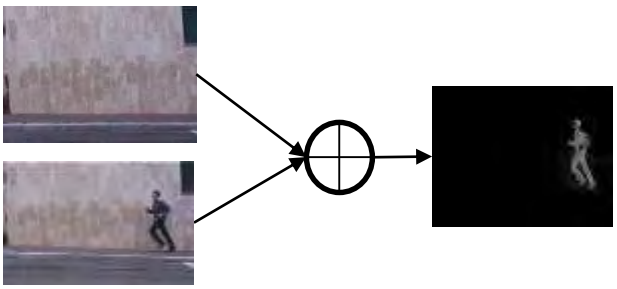

FIGURE 3. Background Subtraction.

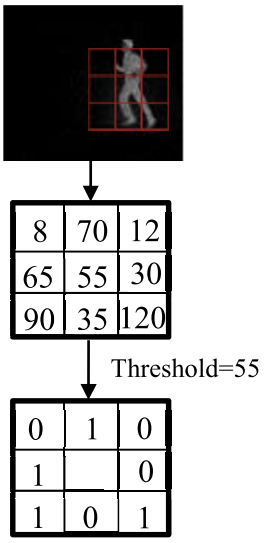

FIGURE 4. An LBP operator.

B. FEATURE EXTRACTION USING ADAPTIVE LOCAL MOTION DESCRIPTOR

In this step, dynamic texture features are extracted from the frames using the proposed Adaptive Local Motion Descriptor (ALMD). It produces consistent patterns against intensity fluctuation. By considering the dynamic selection of threshold value makes our approach flexible enough so that it can adapt with different real-time environments problem e.g. illumination variations and noise. In each worker node features are extracted. Motion is one of the important feature to recognize the human actions from videos, which identified by the alteration of pixel intensity values between two consecutive frames. Our proposed method, adaptive local motion descriptor describes the characteristics of the motion, which is inspired from the Histogram of Optical flow [24] and Local Ternary Pattern [26]. Dissimilar activities in the videos have different gray value changes between frames with respect to time. Therefore, we can distinguish the actions efficiently.

Local pixel information from each frame can be computed using Local Binary Pattern [25]. The basic Local Binary Pattern (LBP) computed by changing the pixel values of an image or frame by thresholding a circular neighborhood area [15]. The $LBP_{P,R}$ generates 2^P different output codes. LBP on pixel (x, y) is calculated as,

$$LBP_{P,R}(x, y) = \sum_{N=0}^{P-1} s(g_N - g_C)2^N \tag{2}$$

$$\text{Where,} \quad s(q) = \begin{cases} 1, q \geq 0 \\ 0, otherwise \end{cases} \tag{3}$$

Here, P and R are the total number and the radius of the neighboring pixels and g_N is neighbor pixel and g_C is center pixel. Fig. 4 illustrates the LBP operator. However, the weakness of the LBP operator is that, it is susceptible to noise, with slight variation in the intensities of the neighbors can completely change the resultant binary code and therefore it fails to produce the consistent patterns in uniform areas, where the difference between the center and the neighbor gray levels is ignorable. Moreover, LBP cannot describe the motion information of a video, it can only able to explain the static texture information. In order to resolve this issue, we propose adaptive local motion descriptor by comparing two consecutive frames and taking the median of the neighbor pixels as the threshold value and extending the Local Ternary Pattern [26], which can provide dynamic motion information as well as can produce steady pattern in uniform and near-uniform regions.

The Local Ternary Pattern (LTP) [26] is computed similar to LBP. However, the main difference is, it considers a new bit to manage the intensity variations. The LTP at a pixel C is calculated by the following equation,

$$LTP_{P,R}(x, y) = \sum_{N=0}^{P-1} s(g_N - g_C)3^N \tag{4}$$

$$\text{Where, } s(q) = -\begin{cases} 1, & if\ q \geq \alpha \\ 1, & if\ q \leq \alpha \\ 0, & otherwise \end{cases} \tag{5}$$

Fig. 5, illustrates an example of the LTP operator. Here, the value of threshold α is 5. After that, to reduce the size of the feature vector, an LTP code is generally divided into two patterns i.e., upper and lower pattern and these two patterns are used for building two histograms independently. Finally, these two histograms are combined to represent the feature vector.

Our proposed method Adaptive Local Motion Descriptor (ALMD) is computed similarly to LTP. At first, it takes two consecutive frames (i.e. previous frame and next frame) to calculate the motion feature. Then, the frames are split into small cells and for two cells at the same place, the neighbor pixels are compared with the next frame center pixel value and threshold value, which is chosen by averaging the median

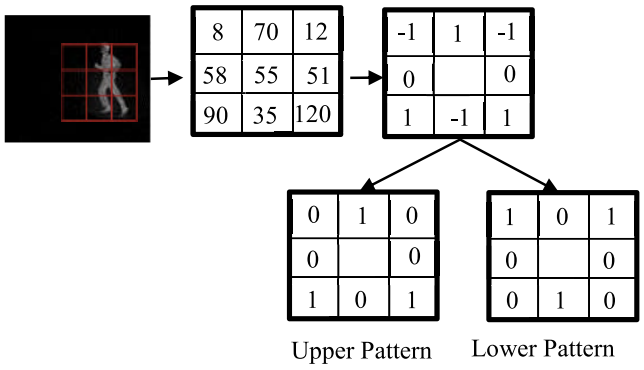

FIGURE 5. An LTP operator.

values, computed from the absolute difference between the center pixel and its neighboring pixels of the two respective frame cells.

After that, the generated code is divided into the upper pattern and lower pattern for both the frames. After that, two upper patterns and two lower patterns from the previous frame and next frame are combined by applying the XOR (exclusive OR) operation. The equations (6)–(8) describe generation of ALMD code.

Here, n and r are the total number and the radius of the neighboring pixels and g_{p_l} is the neighbor pixels of the previous frame, g_{n_l} is the neighbor pixels of the next frame, g_{p_c} is the center pixel of the previous frame and g_{n_c} is the center pixel of the next frame. Uq is for the upper pattern, Lq is for the lower pattern and $\overline{x}$ is the average of the previous frame median $\overline{xp}$ and next frame median $\overline{xn}$, where $\overline{xp}$ is the median of $|g_{p_l} - g_{p_c}|$ values and $\overline{xn}$ is the median of $|g_{n_l} - g_{n_c}|$ values. Fig. 6, demonstrates an example of the ALMD operator, where, $\overline{x} = (\overline{xp} + \overline{xn})/2 = (20 + 30)/2 = 25$. Such a selection of $\overline{x}$ makes our method flexible enough so that it can adapt with different real-time environments problem i.e. illumination variations and noise.

On the other hand, algorithm 1 shows parallel processing of the human action recognition in spark. Here parallel processing is done on each tuple of RDD. The algorithm describes the frame extraction, background subtraction and motion feature extraction using ALMD (Adaptive Local Motion Descriptor). Background subtraction is done by applying equation (1) and motion feature is extracted using ALMD by applying equation (6, 7, and 8), as shown at the bottom of this page. Finally, flatMap function returns the feature vectors from the videos.

Algorithm 1 Feature Extraction

Input: ***RDD [Video_Name N, Video_Data D]***
Output: ***RDD [Video_Name N, Feature_Vector V]***
for all $(N, D) \in RDD[N, D]$ **parallel do**
  flatMap stage:
    $BI \longleftarrow Background\_Image$
    for $i = 1$ to $NumberOfFrames\text{-}1$
      $a \longleftarrow D[i]$
      $b \longleftarrow D[i+1]$
      $Ba \longleftarrow$ apply **BackgroundSubstraction**(***a***, ***BI***)
      $Bb \longleftarrow$ apply **BackgroundSubstraction**(***b***, ***BI***)
      $V[i] \longleftarrow$ apply **Upper_ALMD**(***Ba***, ***Bb***)
      $Q[i] \longleftarrow$ apply **Lower_ALMD**(***Ba***, ***Bb***)
    end for
  add (N, V, Q) to result RDD
end for

C. RANDOM FOREST FOR CLASSIFICATION

Random Forest is an ensemble classifier employing several decision trees that is applied for classification or regression. There are many advantages of random forest that includes production of a highly precise classifier and also can work on big datasets efficiently. The Random Forest algorithm was developed by Breiman [27]. In [28], the idea of "bagging" was proposed and then the random feature selection was presented by Ho [29], [30].

A Random Forest is an ensemble classifier involving a group of tree-structured classifiers $\{r\ (n, \mathbf{\Theta}_k), _{k=1,\ldots,L}\}$ where the $\mathbf{\Theta}_k$ are independent distributed random trees. Here every tree contributes with a vote for the final classification of input n. Similar to Classification and Regression Tree (CART), the Gini index for computing the final output in every tree is applied in random forest. The output of each tree is combined and voted by weighted values to determine the final class. Finally, majority votes is considered to take the final decision.

Every tree gives a vote on a class for the input feature n. The class probabilities are determined by majority voting, which can be written as below,

$$p\,(n) = \arg\max\,(\frac{1}{L}\sum\nolimits_{k=1}^{L} F_{r_k(n)=c}) \tag{9}$$

The decision function $r_k\,(n)$ returns the result class c,

$$F_{r_k(n)=c} = \begin{cases} 1, if & r_k\,(n) = c \\ 0, & otherwise \end{cases} \tag{10}$$

$$For\ Upper\ Pattern,\ ALMD_{n,r}\,(x_c, y_c) = \sum\nolimits_{l=0}^{n-1} \{Uq(g_{p_l} - g_{n_c}) \oplus Uq(g_{n_l} - g_{n_c})\}2^l \tag{6}$$

$$For\ Lower\ Pattern,\ ALMD_{n,r}\,(x_c, y_c) = \sum\nolimits_{l=0}^{n-1} \{Lq(g_{p_l} - g_{n_c}) \oplus Lq(g_{n_l} - g_{n_c})\}2^l \tag{7}$$

$$\text{Where, } Uq\,(a) = \begin{cases} 1 & if\ a \geq \overline{x} \\ 0 & otherwise \end{cases} \text{ and } Lq\,(a) = \begin{cases} 1 & if\ a \leq -\overline{x} \\ 0 & otherwise \end{cases} \tag{8}$$

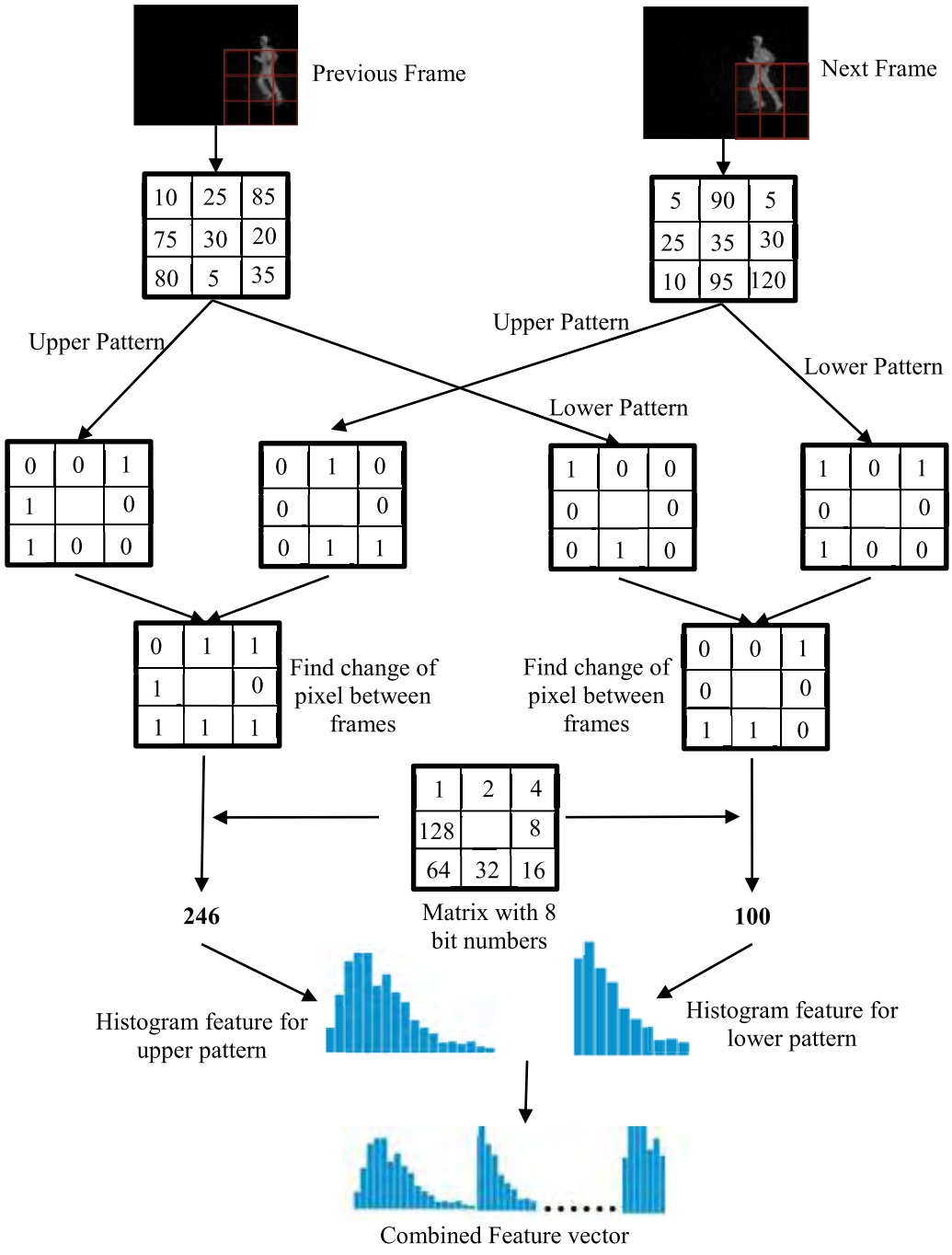

FIGURE 6. An example of Adaptive Local Motion Descriptor.

In Spark MLlib, multiple sub-trees of Random Forest are trained in parallel as each tree in a Random Forest is trained separately. Spark MLlib Random Forest takes the feature vector as input from HDFS which is generated by ALMD. A different subsample of the data is employed in random forest to train each tree. In place of replicating data explicitly, we save memory by applying a TreePoint structure which saves the number of copies of each occurrence in each sub-sample. During training, each node that requires to be split is inserted onto a queue. Some nodes are bring out of the queue considering the volume of memory needed for their necessary computation. Then the worker produces one pass over its subset of instances. The worker gathers information about dividing for each (tree, node, feature, split) tuple and for each node, the information for that node are accumulated to a specific worker. After that, the best (feature, split) pair is selected by the designated worker. Finally, the master gathers all informations about splitting nodes and updates the model.

V. EXPERIMENTAL RESULTS AND ANALYSIS

In order to evaluate the performance of human action recognition on spark, we apply Hadoop spark cluster to conduct the experiments with benchmark datasets of different scales. In our experiments, the cluster includes 4 nodes, one is the master node and all of them are worker nodes. Each node has the same configuration, with 4 cores, 32GB memory, 3.0GHz, Hadoop version is 2.7.1, while spark version is 1.6.2 and spark MLlib version is 1.3.0. Four benchmark datasets are employed for recognizing human actions, including the KTH dataset [10], UCF Sports action dataset [11], [12], UT-Interaction dataset [31] and UCF-50 dataset [32].

A. KTH DATASET

The KTH dataset [10] covers 600 videos including six human action classes: boxing, clapping, jogging, walking, running and waving, with every action class consisting of 100 sequences done by 25 persons in four different situations. An example of KTH dataset, which is taken from [10] is shown in Fig. 7. In our experiment for KTH dataset, 470 videos are applied for training and 130 videos are applied for testing.

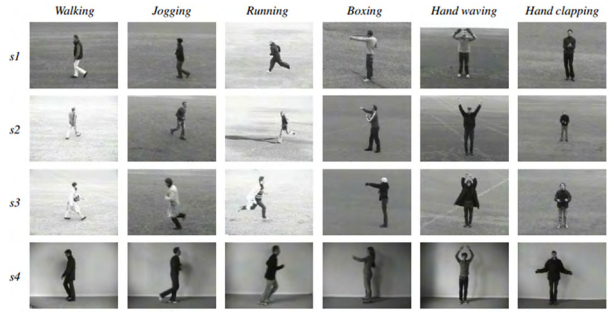

FIGURE 7. Sample frames from the KTH dataset [10].

B. UCF SPORTS ACTION DATASET

The UCF Sports action dataset [11], [12] contains 150 video sequences with the resolution of 720 × 480. It covers 10 human actions that include diving, golf swing, riding horse, walking, running, kicking, lifting, skateboarding, swing-bench, and swing-side. These actions are performed in different real environments that includes different viewpoints and also covering a lot of camera motion. An example of UCF Sports action dataset is presented in Fig. 8. In our experiment for UCF Sports action dataset, 100 videos are applied for training and 50 videos are applied for testing.

C. UT-INTERACTION DATASET

The UT-Interaction dataset [31] consist of 20 video sequences. Each of the videos contains continuous accomplishments of 6 action classes that include shake-hands, point, hug, push, kick and punch. The resolution of all the videos is 720 x 480. Numerous individuals with 15 different clothing conditions perform actions in the videos. An example of UT-Interaction dataset is shown in Fig. 9. In our experiment for UT-Interaction dataset, each video sequence is divided into different categories. So, for 20 video sequences total 120 videos are created. In the experiment, 90 videos are applied for training and 30 videos are applied for testing.

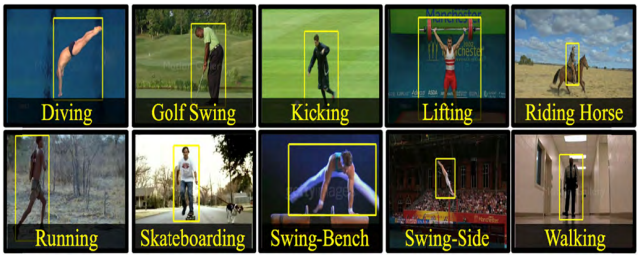

FIGURE 8. Sample frames from the UCF Sports action dataset [11].

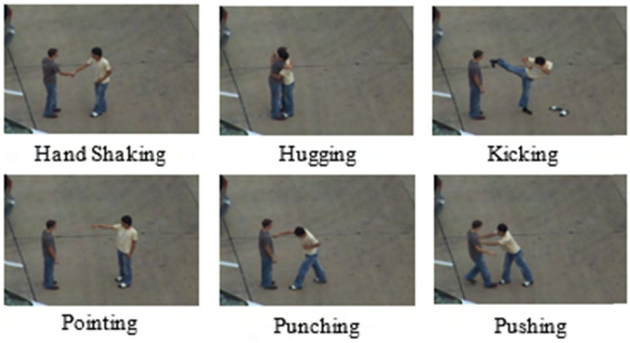

FIGURE 9. Sample frames from the UT-Interaction dataset [31].

FIGURE 10. Sample frames from the UCF-50 dataset [32].

D. UCF-50 DATASET

The UCF-50 dataset [32] is one of biggest action datasets that consist of 6681 videos including 50 action categories. All the videos are collected from the YouTube. The videos are divided into 25 groups covering 4 action clips from each group. The resolution of all the videos is 320 × 240. An example of the UCF-50 dataset is shown in Fig. 10. In our experiment for UCF Sports action dataset, 5000 videos are applied for training and 1681 videos are applied for testing.

Time taken in seconds for feature extraction, training and testing for the KTH dataset, UCF Sports action dataset,

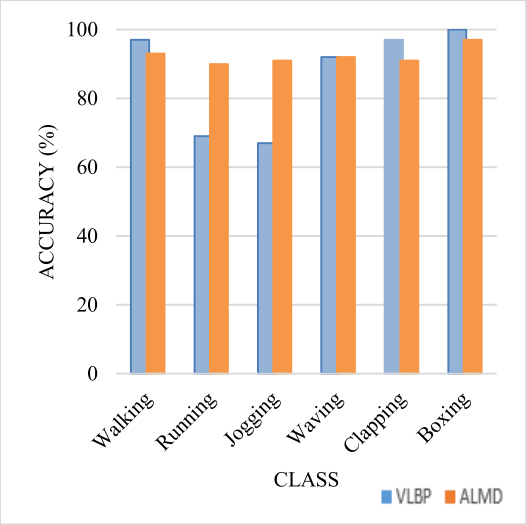

FIGURE 11. Comparison of the accuracy for each class using VLBP with eight neighbors [6] and ALMD (proposed approach) on the KTH dataset.

TABLE 1. Time (in seconds) required for the KTH dataset.

Nodes	1	2	3	4
Feature Extraction	2728s	1978s	1319s	812s
Training	76s	52s	34s	21s
Testing	62s	48s	31s	17s

TABLE 2. Time (in seconds) required for the UCF sports action dataset.

Nodes	1	2	3	4
Feature Extraction	672s	502s	368s	211s
Training	20s	17s	14s	11s
Testing	19s	15s	13s	9s

TABLE 3. Time (in seconds) required for the UT-Interaction dataset.

Nodes	1	2	3	4
Feature Extraction	642s	486s	329s	195s
Training	19s	16s	13s	10s
Testing	18s	13s	12s	8s

TABLE 4. Time (in seconds) required for the UCF50 dataset.

Nodes	1	2	3	4
Feature Extraction	26724s	17916s	11619s	7635s
Training	615s	418s	246s	171s
Testing	240s	173s	112s	68s

UT-Interaction dataset and UCF-50 dataset are illustrated in Table 1, 2, 3, and 4 respectively. Here, when number of node is increased, it takes less processing time, so the performance is increased.

All classification scores are achieved with a Random forest classifier. The performance of the classification is represented

FIGURE 12. Comparison of the accuracy for each class using SFT [4] and ALMD (proposed approach) on the UCF sports action dataset.

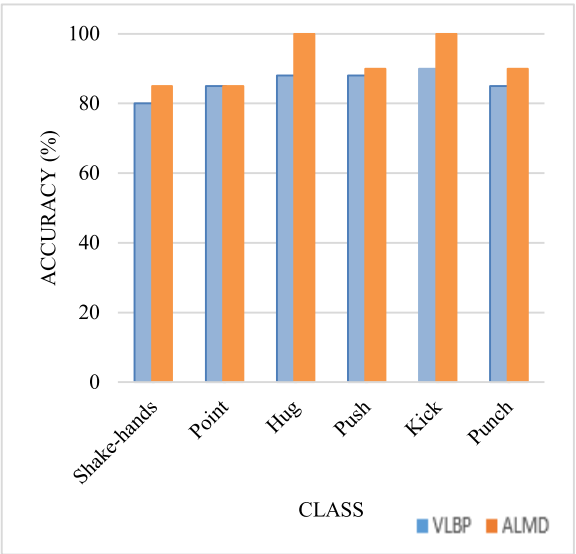

FIGURE 13. Comparison of the accuracy for each class using VLBP with eight neighbors [6] and ALMD (proposed approach) on the UT-Interaction dataset.

as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

Where, TP (True positive) is accurately recognized, TN (True negative) is accurately rejected, FP (False positive) is inaccurately recognized, and FN (False negative) is inaccurately rejected.

With the proposed methodology, the average recognition rates on the KTH dataset, UCF sports action dataset, UT-Interaction dataset and UCF-50 dataset are 92.3%, 92.8%, 91.67%, and 90.1% respectively. The confusion matrices on the UCF sports action dataset, KTH dataset, and

TABLE 5. Confusion matrix for the UCF sports action dataset.

	Diving	Golf Swing	Kicking	Lifting	Riding Horse	Running	Skate-Boarding	Swing-Bench	Swing-Side	Walking
Diving	1	0	0	0	0	0	0	0.05	0.03	0
Golf Swing	0	0.90	0	0	0	0	0.02	0	0	0.03
Kicking	0	0	0.88	0	0	0.05	0	0	0	0.05
Lifting	0	0	0	1	0	0	0	0	0	0
Riding Horse	0	0.05	0	0	1	0	0	0	0	0
Running	0	0	0.07	0	0	0.88	0	0	0	0.05
Skate-Boarding	0	0.05	0.06	0	0	0	0.83	0	0	0.05
Swing-Bench	0	0	0	0	0	0	0	0.95	0.06	0
Swing-Side	0	0	0	0	0	0	0	0.02	0.92	0
Walking	0	0.04	0	0	0	0.05	0	0	0	0.92

TABLE 6. Confusion matrix for the KTH dataset.

	Walking	Running	Jogging	Waving	Clapping	Boxing
Walking	0.93	0.05	0.02	0	0	0
Running	0.04	0.90	0.06	0	0	0
Jogging	0.02	0.07	0.91	0	0	0
Waving	0	0	0	0.92	0.05	0.03
Clapping	0	0	0	0.05	0.91	0.04
Boxing	0	0	0	0	0.03	0.97

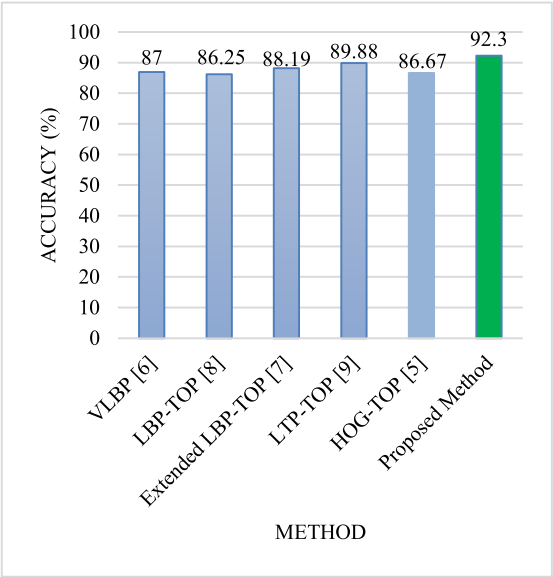

FIGURE 14. Comparison between proposed approach and existing works on the KTH dataset.

UT-Interaction dataset are presented in Table 5, 6 and 7. For the KTH dataset, boxing and walking shows better performance than the other actions, for the UCF sports action dataset, lifting and diving shows better performance compared to the other actions and for the UT-Interaction dataset hug and kick shows better performance compared to the other actions.

On the other hand, for the KTH dataset, running and jogging shows lower accuracy as both the action are almost similar to each other. Jogging is typically a kind of running with a steady speed. For UCF action dataset, skateboarding shows lower accuracy due to the complexity of skateboarding videos.

Comparison of the accuracy for each class using VLBP with eight neighbors [6] and ALMD (proposed approach)

TABLE 7. Confusion matrix for the UT-Interaction dataset.

	Shake-hands	Point	Hug	Push	Kick	Punch
Shake-hands	0.85	0.15	0	0	0	0
Point	0.15	0.85	0	0	0	0
Hug	0	0	1	0	0	0
Push	0	0	0	0.90	0.00	0.1
Kick	0	0	0	0	1	0
Punch	0	0	0	0.1	0	0.90

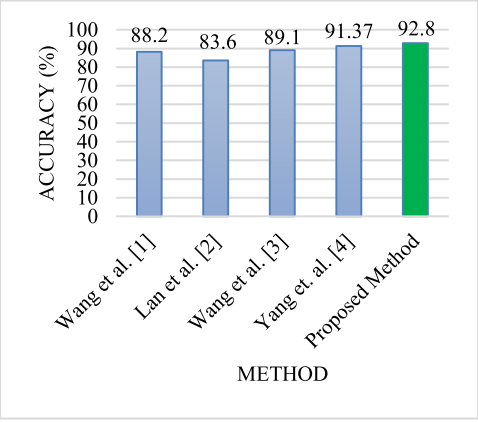

FIGURE 15. Comparison between proposed approach and existing works on UCF sports action dataset.

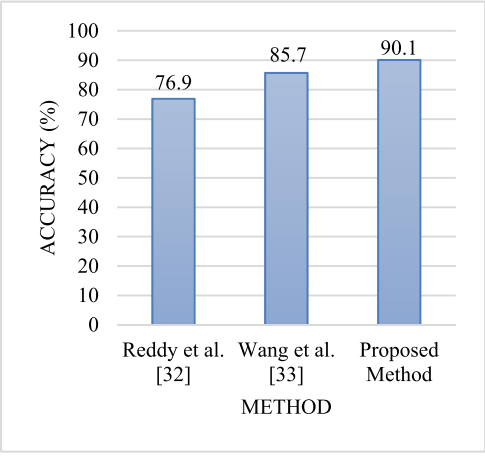

FIGURE 16. Comparison between proposed approach and existing works on the UCF-50 dataset.

on the KTH dataset and UT-Interaction dataset illustrated in Fig. 11 and Fig. 13 respectively. Fig. 12 shows the comparison of accuracy for each class using SFT [4] and ALMD (proposed approach) on the UCF sports action dataset. Fig. 14, Fig. 15 and Fig. 16 shows the comparison between our proposed method and other state-of-the-art works for the KTH dataset, UCF sports action dataset and UCF-50 dataset respectively. From these figures, we can see that our work outperforms other state-of-the-art works. On the KTH dataset, the average accuracy of proposed method is superior to LBP-TOP [8] and VLBP with eight neighbors [6] by 6.05% and 5.3% respectively. On the other hand, for the UCF sports action dataset, our approach slightly outperforms the SFT (salient foreground trajectories) [4] by 1.43% and greatly outperforms the Hierarchical Mid-level Action Elements [2] by 9.2%.

VI. CONCLUSION

In this paper, we presented a novel approach to recognize the human actions. We explored Apache Spark and Spark MLlib to solve human action recognition problem. Moreover, we also introduced a novel feature descriptor, Adaptive Local Motion Descriptor (ALMD) to extract both the texture and motion feature, which is also able to generate persistent patterns. Finally, experimental results showed that our proposed approach outperforms other state-of-the-art works. In the future, we will employ Kafka and Spark Streaming in order to solve the real time human action recognition problem from CCTV video.

REFERENCES

[1] H. Wang, A. Kläser, C. Schmid, and C. L. Liu, "Action recognition by dense trajectories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 3169–3176.
[2] T. Lan, Y. Zhu, A. R. Zamir, and S. Savarese, "Action recognition by hierarchical mid-level action elements," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4552–4560.
[3] H. Wang, A. Kläser, C. Schmid, and C. L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *Int. J. Comput. Vis.*, vol. 103, no. 1, pp. 60–79, 2013.
[4] Y. Yi, Z. Zheng, and M. Lin, "Realistic action recognition with salient foreground trajectories," *Expert Syst. Appl.*, vol. 75, pp. 44–55, Jun. 2017.
[5] J. Chen, Z. Chen, Z. Chi, and H. Fu, "Facial expression recognition in video with multiple feature fusion," *IEEE Trans. Affect. Comput.*, to be published.
[6] F. Baumann, A. Ehlers, B. Rosenhahn, and J. Liao, "Computation strategies for volume local binary patterns applied to action recognition," in *Proc. 11th IEEE Int. Conf. Adv. Video Signal-Based Surveill. (AVSS)*, Aug. 2014, pp. 68–73.
[7] R. Mattivi and L. Shao, "Human action recognition using LBP-TOP as sparse spatio-temporal feature descriptor," in *Proc. Int. Conf. Comput. Anal. Images Patterns (CAIP)*, 2009, pp. 740–747.
[8] G. Zhao and M. Pietikäinen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 915–928, Jun. 2007.
[9] L. Nanni, S. Brahnam, and A. Lumini, "Local ternary patterns from three orthogonal planes for human action classification," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5125–5128, 2011.
[10] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. 17th Int. Conf. Pattern Recognit.*, Aug. 2004, pp. 32–36.
[11] M. D. Rodriguez, J. Ahmed, and M. Shah, "Action MACH a spatio-temporal maximum average correlation height filter for action recognition," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
[12] K. Soomro and A. R. Zamir, "Action recognition in realistic sports videos," in *Computer Vision in Sports*. Berlin, Germany: Springer, 2014, pp. 181–208.
[13] W. Zhang, L. Xu, P. Duan, W. Gong, Q. Lu, and S. Yang, "A video cloud platform combing online and offline cloud computing technologies," *Pers. Ubiquitous Comput.*, vol. 19, no. 7, pp. 1099–1110, Oct. 2015.

[14] M. Kim, Y. Cui, S. Han, and H. Lee, "Towards efficient design and implementation of a Hadoop-based distributed video transcoding system in cloud computing environment," *Int. J. Multimedia Ubiquitous Eng.*, vol. 8, no. 2, pp. 213–224, 2013.
[15] H. Tan and L. Chen, "An approach for fast and parallel video processing on Apache Hadoop clusters," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2014, pp. 1–6.
[16] X. Liu, D. Zhao, L. Xu, W. Zhang, J. Yin, and X. Chen, "A distributed video management cloud platform using Hadoop," *IEEE Access*, vol. 3, pp. 2637–2643, 2015.
[17] H. Wang, Y. Shen, L. Wang, K. Zhufeng, W. Wang, and C. Cheng, "Large-scale multimedia data mining using MapReduce framework," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2012, pp. 287–292.
[18] M. Z. Uddin and J. Kim, "A local feature-based robust approach for facial expression recognition from depth video," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 3, pp. 1390–1403, 2016.
[19] M. Z. Uddin and J. Kim, "Human activity recognition using spatiotemporal 3-D body joint features with hidden Markov models," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 6, pp. 2767–2780, 2016.
[20] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. HotCloud*, vol. 10. 2010, pp. 1–7.
[21] M. Zaharia *et al.*, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. Netw. Syst. Design Implement. (NSDI)*, 2012, p. 2.
[22] X. Meng *et al.*, "MLLIB: Machine learning in apache spark," *J. Mach. Learn. Res.*, vol. 17, pp. 1235–1241, Jan. 2016.
[23] M. Z. Uddin, J. J. Lee, and T.-S. Kim, "Shape-based human activity recognition using independent component analysis and hidden Markov model," in *Proc. Int. Conf. Ind., Eng. Other Appl. Appl. Intell. Syst.*, 2008, pp. 245–254.
[24] J. Perš, V. Sulić, M. Kristan, M. Perše, K. Polanec, and S. Kovačič, "Histograms of optical flow for efficient representation of body motion," *Pattern Recognit. Lett.*, vol. 31, no. 11, pp. 1369–1376, 2010.
[25] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
[26] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1635–1650, Jun. 2010.
[27] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[28] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[29] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, Aug. 1995, pp. 278–282.
[30] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.
[31] M. S. Ryoo and J. K. Aggarwal, "Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities," in *Proc. 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 1593–1600.
[32] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of Web videos," *Mach. Vis. Appl.*, vol. 24, no. 5, pp. 971–981, 2013.
[33] L. Wang, Y. Qiao, and X. Tang, "Mining motion atoms and phrases for complex action recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2680–2687.
[34] Q. Xiao and R. Song, "Action recognition based on hierarchical dynamic Bayesian network," in *Multimedia Tools and Applications*. New York, NY, USA: Springer, 2017, pp. 1–14.
[35] *YouTube Statistics–2017*. Accessed: Aug. 17, 2017. [Online]. Available: https://fortunelords.com/youtube-statistics/
[36] M. Z. Uddin, M. M. Hassan, A. Almogren, A. Alamri, M. Alrubaian, and G. Fortino, "Facial expression recognition utilizing local direction-based robust features and deep belief network," *IEEE Access*, vol. 5, pp. 4525–4536, 2017.
[37] M. H. Kabir, M. S. Salekin, M. Z. Uddin, and M. Abdullah-Al-Wadud, "Facial expression recognition from depth video with patterns of oriented motion flow," *IEEE Access*, vol. 5, pp. 8880–8889, 2017.

MD AZHER UDDIN received the B.S. degree in computer science and engineering from International Islamic University Chittagong, Bangladesh, in 2011. He is currently pursuing the Combined M.S. and Ph.D. degree with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. He also served as a Lecturer with the Computer Science and Engineering Department, International Islamic University Chittagong from 2012 to 2015. His research interests include image processing, machine learning techniques, and big data.

JOOLEKHA BIBI JOOLEE received the B.S. degree in computer science and engineering from International Islamic University Chittagong, Bangladesh, in 2015. She is currently pursuing the master's degree with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. Her research interests include image processing, machine learning, and augmented reality.

AFTAB ALAM received the master's degree in computer science with specialization in Web engineering from the Department of Computer Science, University of Peshawar, Pakistan. He is currently a Ph.D. Scholar with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. His research interests include graph mining, graph summarization, social information seeking, social networks, and big data.

YOUNG-KOO LEE received the B.S., M.S., and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 1992, 1994, and 2002. Since 2004, he has been with the Department of Computer Science and Engineering, Kyung Hee University. His researches have been concentrated on data mining, online analytical processing, and big data processing.