Sensors and Materials, Vol. 38, No. 6 (2026) 3157–3174
MYU Tokyo

S & M 4495

Lightweight Lifecare Remote Monitoring System for Human Behavior Interaction Using Embedded Hidden Markov Model

Tanvir Fatima Naik Bukht,[1,2†] Yanfeng Wu,[1†] Bayan Alabdullah,[3] Khaled Alnowaiser,[4] Ahmad Jalal,[2,5*] and Hui Liu[6**]

[1]Guodian Nanjing Automation Co., Ltd, Nanjing, China
[2]Faculty of Computing and AI, Air University, E-9, Islamabad 44000, Pakistan
[3]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
[4]Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[5]Department of Computer Science and Engineering, College of Informatics, Korea University, Seoul 02841, South Korea
[6]Cognitive Systems Lab, University of Bremen, Bremen 28359, Germany

(Received November 13, 2025; accepted February 13, 2026)

Keywords: spatial fusion, temporal analysis, segmentation, human behavior interaction, texton maps

Remote monitoring in lifecare presents a vision of human behavior interactions in complex situations that were previously overlooked. In this research, we designed lifecare algorithms that recognize the activities from processed video sequence images. This groundbreaking technology enables remote monitoring systems to collect reliable data that can be applied across fields such as healthcare, sports, and security. We integrate an embedded hidden Markov model (E-HMM) with visual-sensor-based data input to enhance the efficiency of human behavior interaction systems. The proposed method begins by performing a hue saturation value color transformation to improve the clarity of video frames. The silhouette is extracted using hybrid techniques with sensor data, and signal features are extracted using texton maps, a local intensity order pattern, and oriented features from accelerated segment test and rotated binary robust independent elementary features. Fuzzy optimization is then carried out to choose the most discriminative signal features. An E-HMM is trained to identify actions correctly according to the given functions. Furthermore, since the suggested approach monitors the order of actions, it uses time-related data, which results in improved detection results, even in the presence of occlusions or when actions are performed at various speeds and scales. The sensor data used in the experiment, combined with the recognition algorithms, achieved the following results: Shakefive2: 0.97%, HMDB51: 0.90%, and Okutama Action: 0.68, 0.94, and 0.82%.

*Corresponding author: e-mail: ahmadjalal@au.edu.pk
**Corresponding author: e-mail: hui.liu@uni-bremen.de
†These authors contributed equally to this work.
https://doi.org/10.18494/SAM6026

ISSN 0914-4935
https://myukk.org/

1. Introduction

Human–computer interaction has been combined with technology enhanced through the integration of sensors.[1] Liu *et al.*[2] proposed a technique combining template matching, motion differencing, histogram of oriented gradients features, and tracking methods for sensor-driven human behavior analysis. The use of color-region approaches for moving-object recognition distinguishes the technique. We developed a human–human behavior interaction (HBI) system that can recognize complex human activities from the ShakeFive2, HMDB51, and Okutama datasets. HBI research is still a subject of intense discussion, but we have made significant progress.

The proposed system comprises the following key contributions by employing advanced techniques, including the combination of HSV transformation and median filtering, to enhance and extract the necessary information from the frames. Multiple state-of-the-art methods, including Multiple Object Tracking, Streakflow, structure from motion, visual body-audio excitation, and flow-based time-varying layer extraction, are employed to extract silhouettes from the processed frames accurately. Advanced techniques for feature extraction, such as Texton Maps, LIOP, and oriented FAST and rotated BRIEF (ORB), are used to extract informative features from the obtained silhouettes. A fuzzy-optimization-based approach is employed to discriminate between different features effectively. This approach optimizes the signal feature discrimination process. We subsequently identify it using our developed novel embedded hidden Markov model (E-HMM) method.

We proposed an E-HMM-based method for interaction recognition primarily because of its strength in modeling sequential data and the temporal dynamics inherent in human actions. E-HMMs effectively capture state transitions over time, which is critical for recognizing complex interactions. We focus on hybrid techniques of silhouette extraction and then a selective signal and sensor feature extraction procedure that uses a combination of methods, such as Texton maps and LIOP, to guarantee a rich and discriminative feature representation. One of the novel contributions to our work that has been developed is the incorporation of a three-layer E-HMM. Traditional HMMs are usually applied to sequential data; we expand their ability to include them in a three-layer novel model that is developed and enables us to capture temporal dependencies at various levels of abstraction. With this method, our model will be in a better position to comprehend the context and dynamics of human actions over time, particularly where human actions are complex action sequences or interactions. We have developed a lightweight monitoring system that is suitable for any edge device and completes tasks with minimal processing overhead, making it ideal for many applications.

This article is structured as follows. Section 2 is the literature review and Sect. 3 shows the framework, which consists of preprocessing, silhouette extraction, feature extraction, discrimination, HBI system experiments, and comparison results. In Sect. 4, we summarize our findings and insights.

2. Literature Review

Human activity can be defined as a set of activities performed by an individual or a group of people. HBI applies the latest methods, such as sensor data collection, machine learning, deep learning, life care monitoring, and data analysis, to create intelligent systems to improve monitoring and support. Human behavior interaction methods, including surveillance and suspicious activity monitoring[3] in healthcare, are used for identifying abnormal behavior in the open area and possible threats, and ensuring the safety of people. HBI assists healthcare professionals in monitoring patient movements, providing real-time feedback on physical therapy exercises, and adjusting treatment plans accordingly. Köping *et al.*[4] developed a smartphone-based framework for real-time activity identification, achieving 87.1% accuracy using extracted features. Reliability was enhanced by kernel principal component and linear discriminant analyses. In their study, Manzi *et al.*[3] presented an activity recognition system using depth camera skeleton data and machine learning techniques. This system classifies actions on the basis of postures using multi-class SVM and X-means algorithms.

Khodabandelou *et al.*[5] introduced a fuzzy-logic-based deep learning algorithm that predicts lower limb exoskeleton users' daily activities using real-time locomotion data, estimating gait mode transitions, and evaluating performance using dynamic data. Neural network techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have recently become viable candidates for human recognition owing to developments in the area of deep learning.[6] They can learn structural representations of images, videos, and activity embodiments. The current CNN-based approaches are practical in domains such as medical image classification or object detection; however, to achieve this, we require a large amount of labeled data for training, and the computational complexity is high. Lastly, certain aspects of sensor-driven human behavior monitoring, such as motion history image (MHI), optical flow, and 3D CNNs, have been developed for determining human behavior. Additionally, various approaches have been proposed in some literature to improve human pose detection and recognition efficiency. These improvements include the integration of CNNs and HMM.[7] We explain in Table 1 how our proposed framework addresses these limitations in terms of response time, security, and accuracy rate, as well as the limitations of related approaches. Citations from relevant research support various points.

3. Proposed Human Monitoring System

Figure 1 shows the architecture of our proposed system. We convert the video dataset into frames and utilize the HSV color space to depict human silhouettes against the background more effectively. Second, a hybrid approach was employed, using multiple techniques to extract robust silhouettes, including MOT, Streakflow, SFM, VIBE, and FTLE methods, to capture human motion dynamics. In the third step, the local texture and appearance features extracted from Texton Maps, LIOP, and ORB techniques, and sensor-based motion data are used to recognize human actions. The fourth stage of the fuzzy optimization procedure is applied to select the most discriminative features, aiming to maximize classification accuracy and reduce computational complexity.

Table 1
Comparative analysis of related approaches and improvements of the proposed framework.

Approach	Limitations	Our framework improvements
Traditional feature extraction	It is based on handcrafted features (e.g., SIFT and HOG), which may not be robust against complex patterns or contextual information.[8]	The feature extraction methods used (Texton maps, LIOP, and ORB) are more robust in terms of data variations.
Machine learning techniques	Naive bayes: assumes independent features[9] Logistic regression: draws linear decision boundaries[10] Support vector machine (SVM): parameter selection is sensitive and computationally expensive for large datasets[10] Random forest: noisy data can make data very complex[11] Decision trees: unreliable because they frequently overfit the training data, and their structure can change significantly, even with slight variations in the data[12] Artificial neural networks (ANNs): sensitive to tuning and require a lot of data[13]	We implement a fuzzy optimization process to select the most discriminative features, which improves generalization and robustness across different datasets, thereby increasing accuracy and response time.
Deep learning techniques	CNNs require large amounts of labeled data and high computational cost, which can impede the response time.[12] In RNNs, capturing long-range dependencies is complex and can lead to vanishing gradients.[14] Shallow HMM models are limited in capturing complex temporal dependencies, resulting in low accuracy in action detection.[15]	We developed a novel three-layer E-HMM to capture complex temporal dynamics and improve response time and accuracy while maintaining security through model complexity and robustness against adversarial attacks.
Flow-based methods	Flow-based methods for interactions between multiple humans are not captured well.[16]	We utilize FTLE, in conjunction with other methods, to enhance the modeling of interactions.

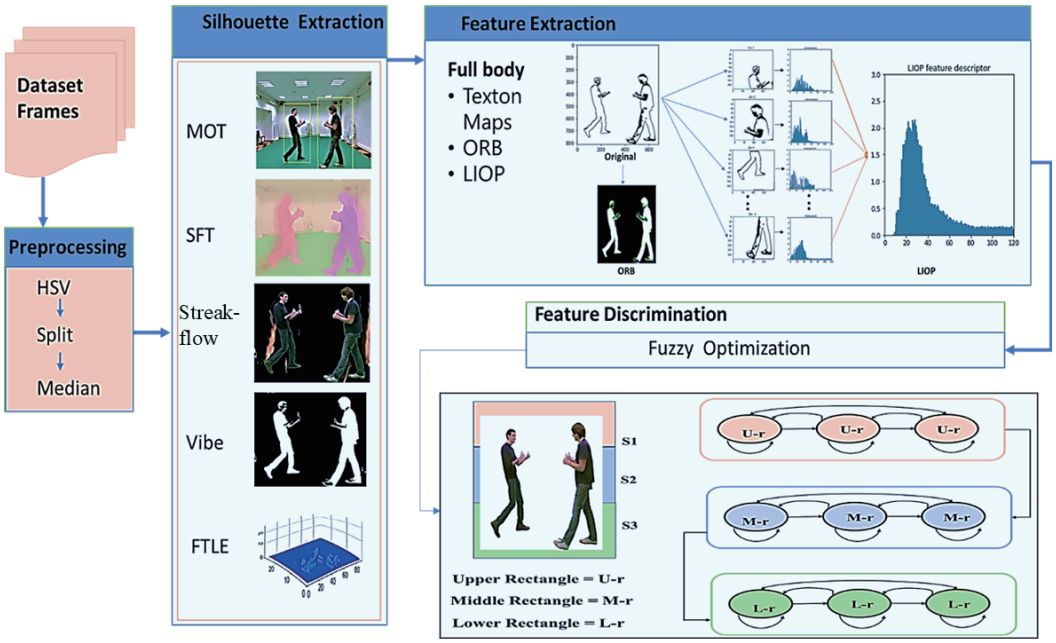


Fig. 1. (Color online) Architecture flow of our proposed HBI system involves preprocessing data frames, silhouette and feature extraction, feature discrimination, and E-HMM.

3.1 Preprocessing

Preprocessing techniques transform raw sensor data into a suitable format for characterizing and extracting features. Preprocessing entails the transformation of the color space to HSV, the most appropriate channel is chosen, and a median filter is applied. It allows making independent changes to tone, saturation, and value, thus improving the quality of the visual image by adding the perceived contrast between various areas or objects. The HSV color model has three components, namely, hue, saturation, and value, which divide the image into these components to provide contrast perception and enable the precise analysis of color information.

3.2 Silhouette extraction

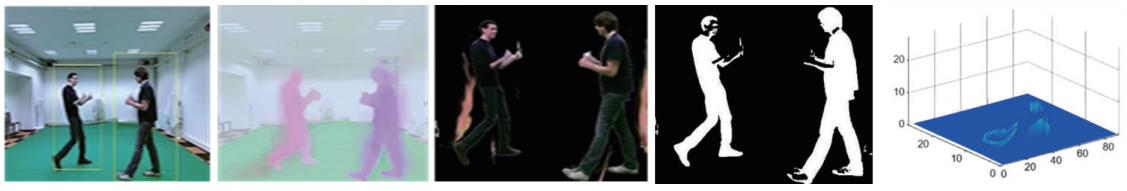
It is challenging to accurately detect human silhouettes, which is necessary for effective feature extraction. We developed a method to obtain accurate human silhouettes from the video data captured by imaging sensors, such as depth cameras and infrared motion detectors. Figure 2 illustrates the silhouette extraction results of the action "thumbs-up" of MOT, streakflow (STF), SFM, VIBE, and FTLE.

3.2.1 Silhouette detection and tracking

MOT based on Kalman filtering is being actively used in computer vision and activity recognition. In the case of noisy data, the Kalman filtering technique is a recursive method that is computationally oriented to estimate the status and state of the system.[17]

$$\hat{X}_t = \arg\max_{X_t} \sum_{i=1}^{N} P\left(D_t^i \,|X_t, D_{1:t-1}^{1:N}\right) \cdot P\left(X_t\right) \tag{1}$$

Equation (1) represents the conditional probability $P\left(X_t|X_{t-1}, D_{1:t-1}^{1:N}\right)$, where X is the state at time t, X_{t-1} is the previous state, i represents the index for the current state, and $D_{1:t-1}^{1:N}$ denotes the sequence of observed data from time step 1 to $t-1$.



(a) (b) (c) (d) (e)

Fig. 2. (Color online) Silhouette extraction results of the action "thumbs-up" of (a) MOT, (b) STF, (c) SFM, (d) VIBE, and (e) FTLE.

3.2.2 Silhouette prediction and association

The description of the flow behavior, including all directional and magnitude information of flow fields, is adapted from a method for the visual observation of fluid motion, known as STF representation of the flow. The method involves generating streaklines, the lines that a particle injects into the flow at each point over time. It integrates the amount of flow that occurs across space and time, or along a line, and provides information about the previously mentioned flow conditions at various points.

3.2.3 Spatial fusion and temporal analysis

Our research suggests that there are still several ways to address problems such as occlusion, noisy backgrounds, and variable illumination: the first is to utilize patterns, the second is appearance, and the third is motion. By predicting how individuals would move in a crowd, the technique is used to visualize the shape of a crowd in disciplines such as crowd management and urban planning, as it considers both internal and external forces that affect movement patterns.

$$F_i = m_i a_i = \sum_{j \neq i} F_{ij} + F_{ext,i} \tag{2}$$

Here, F_i depicts the total force F_i acting on individual i, which is equal to the sum of the forces exerted by other individuals (F_{ij}) and the external forces ($F_{ext,i}$).

3.2.4 Foreground segmentation

The VIBE algorithm distinguishes essential items in frames by removing the image's background.[18] The VIBE system is often used for this purpose. The process begins with random pixel values, which set up the image's background. Then, careful adjustments are made to the model to help single out essential items. Once the image's binary mask is obtained, it can be further analyzed to reveal object boundaries and characteristics.

3.2.5 Flow-based time-varying layer extraction

Flow field analysis is performed using the FTLE metric, which measures the stretching and folding tendencies of the fluid over a finite period. This generates valuable data about the behavior of the system and its components in the functioning process. To evaluate deformation rates at the local level, evaluating particle trajectories in combination with estimating flow fields enables the accurate computation of FTLE values.

In algorithm 1, first, the Kalman gain (K_t) is calculated by combining the predicted covariance matrix (P_t^{-1}) and the measurement noise covariance (R_t). The measurement vector (z_t) is updated with the observed object's position. For each tracklist (T_i) associated with the detection (D_j), the state estimate ($x_t|_t^{-1}$) is predicted using the state transition matrix (F_t), and the predicted covariance ($P_t|_t^{-1}$) is computed by applying the transition matrix and adding process noise (Q_t).

Algorithm 1
Silhouette extraction.

Input: Sequence of frames $F_1, F_2, ..., F_T$
Output: Extracted silhouettes for each frame

Initialization:
$\hat{x}_0\ [x_0\ v_0], P_0\ \ [Px_0\ 0\ 0\ Pv_0]$
For $t = 1$ to T **do**
Silhouette Detection and Tracking MOT:
Perform object detection using the MOT algorithm on frame F_t;
Associate detections with existing tracks.
For each detection, D_j **do**
Update: Update the trackless state
$K_t \leftarrow P_{t|t-1}H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1}$
$z_t \leftarrow [x_j + w_j \quad y_j + w_j]$
For each tracklist T_i associated with detection D_j **do**
$\hat{x}_{t|t-1} \leftarrow F_d x_{t-1}$
$P_{t|t-1} \leftarrow F_t P_{t-1} F_t^T + Q_t$
$\hat{x}_t \leftarrow x_{t|t-1} + K_t (z_t - H_t x_{t|t-1})$
end
end
(STF) → (SFM): → (VIBE): → (FTLE): → Post-processing and Refinement:
Refine extracted silhouettes;
Output:
Store extracted silhouettes for frame F_t.
end

Finally, the state estimate ($\hat{x}_t$) is updated by adjusting the predicted estimate based on the Kalman gain and the measurement residual, and to refine the tracking accuracy, we used $FTLE(x_0, T)$. The value is calculated for the time interval T at the starting position x_0, and $\partial X\left(t, x_0\right) / \partial x_0$ refers to the Jacobian matrix, which enables the linear mapping of the flow field at the particular trajectory point. The operator $||\cdot||$ is used to denote the norm of a matrix.

$$FTLE\left(x_0, T\right) = \frac{1}{T} \log \frac{\partial X\left(t, x_0\right)}{\partial x_0} \tag{3}$$

3.3 Feature extraction

The features of image recognition applied in this paper are the texture-based features obtained with the help of Texton Maps, LIOP, and ORB. The features were chosen according to their capacity to identify unique patterns and variations in visual and depth signal information.

3.3.1 Texton maps

Texton maps utilize the texture information extracted from photographs to assess outlines, enabling their application in tasks such as object recognition and scene comprehension. Texton mapping classifies similar texture patterns into textons, where $T(x, y)$ is used to compute the Texton Map value at pixel coordinates (x, y) by choosing the most suitable texton (represented as

t) from an adjacent region (referred to as N) that includes the pixel. The function $d(\cdot)$ is used to calculate the distance between the intensity of the input picture at coordinates (x, y) and the texton value at coordinates (t, u, v), where (u, v) are the coordinates within the neighborhood.

$$T(x,y) = \arg\min_{t} \sum_{(u,v)\in N} d\left(I(x,y), T(t,u,v)\right) \tag{4}$$

The Texton Map, $T(x, y)$, is obtained by applying the argmin algorithm to identify the texton that minimizes the distance.

3.3.2 Local intensity order pattern

Algorithm 2 extracts patches from an input image, normalizes them, and divides the normalized image into regions. This process then computes the LIOP descriptor for each region by calculating the LIOP value for each pixel in the region and concatenating pixel-level LIOP values. Finally, it concatenates the descriptors for all regions to obtain the LIOP feature vector for the entire image.

Here, p is the center pixel, c is the neighbor pixel, P is the number of pixels in the neighborhood, $g_i(p)$ is the intensity of the i-th pixel in the neighborhood of p, and $I(g_i(p) > g_c(p))$ is the indicator function that returns 1 if the condition is true and 0 otherwise. The condition inside the parentheses is $g_i(p) > g_c(p)$, which means that the intensity of the i-th pixel in the neighborhood of p is greater than that of the neighbor pixel c. If $g_i(p) > g_c(p)$ is valid, then $I(g_i(p) > g_c(p))$ returns 1, and the corresponding LIOP bit is set to 1 in the LIOP code. Otherwise,

Algorithm 2

Algorithm for computing LIOP descriptors for an input image.

Input: An input image *I*
Output: LIOP descriptors for the input image

Step 1: Extract patches of the input image.
    Patches ← extract **patches(*I*, patch size);**
Step 2: Normalize each patch using histogram equalization.
    Patches norm ← normalize patches(patches);
Step 3: Divide the normalized image into regions.
    Regions ← divide regions(patches_norm, num regions);
Step 4: Compute LIOP descriptors for each region.
    liop descs ← empty list;
foreach region in regions do
    liop desc empty list;
    foreach pixel in region **do**
        liop val ← calculate liop value (pixel);
        liop desc.append(liop val);
    end
    liop descs.append(liop desc);
end
Step 5: Concatenate the LIOP descriptors for all regions.
    liop features ← concatenate(liop descs);
Return the liop features.

if $g_i(p) > g_c(p)$ is false, then $I(g_i(p) > g_c(p))$ returns 0, and the corresponding LIOP bit is set to 0 in the LIOP code.

$$LIOP(p,c) = \sum_{i=1}^{P} \left[I\left(g_i(p) > g_c(p)\right) \cdot 2^{i-1} \right] \tag{5}$$

3.3.3 ORB

In ORB, the orientation component helps rotate the features to match those of the other orientations. ORB features offer certain advantages that enable the effective identification of distinctive image information, providing a foundation for accurate and fast visual data evaluation. In this research, we can locate the centroid Cen_{ij} of the image in ORB using the patch moment, as shown below. Figure 3 employs techniques such as Texton Maps, ORB, and LIOP descriptor to extract and understand the unique features of the image.

$$Cen_{ij} = \sum_{p,q} p^i q^j (x,y) \tag{6}$$

3.4 Feature optimization

Fuzzy-optimization-based feature selection has gained recognition as an effective strategy for identifying and selecting relevant features in complex datasets.

$$F(X) = \sum_{i=1}^{n} w_i \cdot \mu(x_i) \tag{7}$$

Here, $F(X)$ is the fuzzy optimization function, which depends on the input set X having n elements. Each element x_i is associated with a membership value $\mu(x_i)$, which is the grade to which x_i belongs to a particular fuzzy set. The function employs weights w_i to signify the

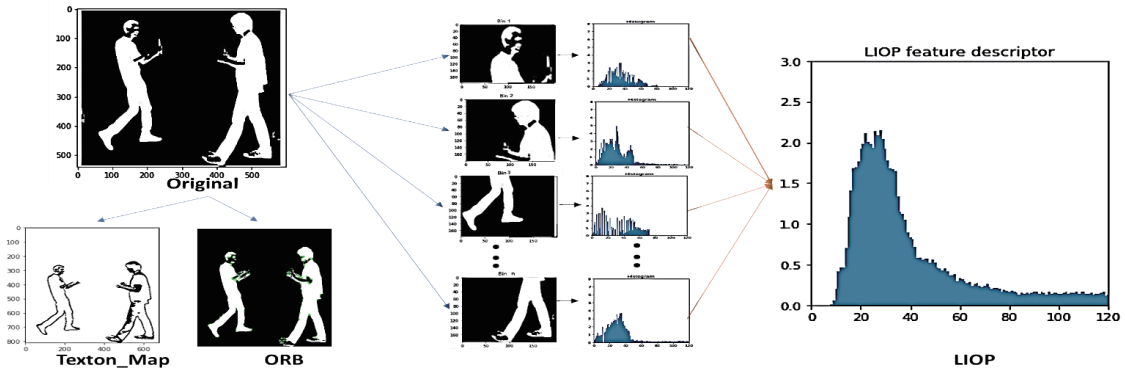


Fig. 3. (Color online) Feature extraction shown by Original frame, Texton Maps, ORB, and LIOP descriptor.

importance assigned to each component in the optimization process. Figure 4 presents the feature differentiation across the (a) ShakeFive2 dataset, (c) HMDB51, and (b) Okutama Action. Feature optimization is performed using fuzzy logic, which quantifies each feature's relevance using membership functions. The optimization process aims to minimize redundancy and maximize feature discriminability, as evaluated by classification results before and after optimization. The performance gain is calculated as

$$\Delta A = A_{optimized} - A_{baseline}. \tag{8}$$

Here, $A_{optimized}$ is the accuracy after feature optimization and $A_{baseline}$ is the accuracy before optimization.

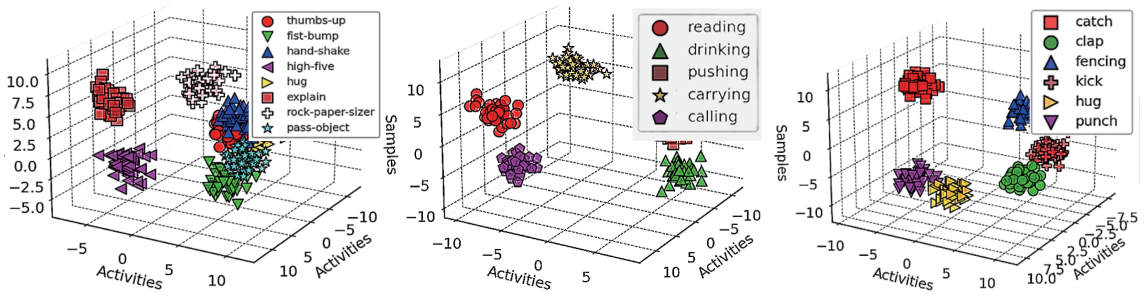
3.5 E-HMMs

E-HMMs are outlined as advanced stochastic models widely used in application areas such as speech recognition and bioinformatics.[7] The research is more appropriate when there are long dependencies within the sequence and between observations. Consequently, EHMMs serve as uniquely valuable tools for modeling and pattern recognition in those fields, where the capability to understand sequential data defines the decisive factor in many cases, even though the computational expense of the method constitutes its weakness.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \cdot A_{ij}\right] \cdot B_j\left(o'_{t+1}\right) \tag{9}$$

$$\beta_t(i) = \sum_{j=1}^{N} A_{ij} \cdot B_j\left(o'_{t+1}\right) \cdot \beta_{t+1}(j) \tag{10}$$

The HAR with three-layer E-HMM algorithm 3 can recognize human activities by using full-body features. The upper threshold is denoted as U and the lower threshold is denoted as L. Initial state probabilities: π_i, the likelihood of moving from state i to state j, denoted by A_{ij}, and



(a) (b) (c)

Fig. 4. (Color online) Discrimination of features over the (a) ShakeFive2 dataset, (b) Okutama Action, and (c) HMDB51.

Algorithm 3
Human activity recognition with three-layer E-HMM.

Require: Full-body features in a .csv file
Output: Recognized human activity

Divide Full-Body Features into Upper, Middle, and Lower Parts
Read the full body features
Sort the features, calculate thresholds for each layer:
U = mean (features) + std(features); L = mean (features) − std(features);
Divide features into upper, middle, and lower parts:
foreach feature f **do**
  if $f > U$ **then**
    Assign it to the Upper Rectangle (S1)
  else if f is between U and L, **then**
    Assign it to the Middle Rectangle (S2)
  else
    Assign it to the Lower Rectangle (S3)
  end
end
E-HMM for Each Layer;
foreach layer S1 ,S2 ,S3 do
  Initialize Model Parameters;
  $\pi_i; A_{ij}, B_i(o'_1)$
  Compute Forward Probabilities;
  Optimized feature sequence $O' = \{o'_1, o'_2, ..., o'_T\}$ $\alpha_1(i) = \pi_i \cdot B_i(o'_1)$ for $i = 1$ to N
  For $t = 1$ to $T - 1$ and $j = 1$ to N **do**
    $\alpha_{t+1}(j)$
  end
  Compute Backward Probabilities;
  $\beta_T(i) = 1$ **for** $I = 1$ to N **for** $t = T - 1$ to 1 and $i = 1$ to N **do**
    $\beta_t(i)$
  end
  Estimation of Emission Parameters;
  Update:
  For $i = 1$ to N, and k representing different observations, **do**
    $B_i(k) = E_{ijk}$
  end
  Estimation of Transition Parameters;
  Update:
  For i and j representing different states, **do**
    $A_{ij} = E_{ij}/E_i$
  end
  end decision;
  Used the Viterbi algorithm
end

the probability of emitting observation 'o_t' given the state is i, denoted by $B_i(o_t)$. The system categorizes the characteristics into three sections: the feature signal strengths are classified into three levels: high, middle, and low prominence. It uses an E-HMM for each step and computes the forward Eq. (9) $\alpha_{t+1}(j)$ and backward Eq. (10) probabilities. $\beta_t(i)$ is used to calculate the emission and transition probabilities, and uses the Viterbi algorithm for activity identification. In the current research, we employed an E-HMM to assess the new approach, as shown in Fig. 5. In Eq. (10), the recognition accuracy A_{Ehmm} is computed as Eq. (11), where TP denotes true positives, TN true negatives, FP false positives, and FN false negatives. Accuracy is averaged across all action classes for the final result.

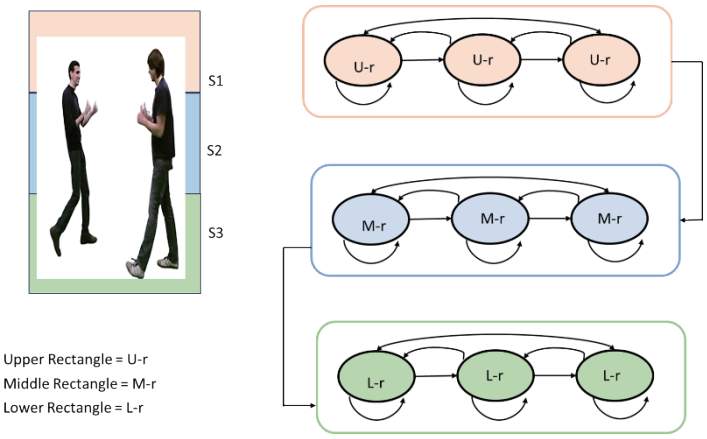


Fig. 5. (Color online) Tailed internal architecture of the three-layer E-HMM showing regional feature partitioning and hierarchical processing.

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

The experiment was carried out with a lot of attention to detail and the data collected were subjected to different quality control measures. To evaluate the performance of the classifier in detail, several measures were used. Thus, to obtain a clear picture of the accuracy of the classifier, accuracy, recall, and F1-score were employed. The results of the proposed system classification for the Shakefive2 dataset are shown in Table 2.

The ShakeFive2 confusion matrix (Table 3) shows a strong performance. The gesture that was most difficult to perform (94% TP) was the Hug gesture because it was highly varied and covered during close interactions. In general, the system is able to cope with complicated interactions, and the misclassification is mostly restricted to the ambiguous hand positioning and overlapping motion paths.

The proposed E-HMM method for recognizing human interactions is visually depicted in Table 4, and Table 5 presents our quantitative findings. Tables 6 and 7 show the classification and confusion matrix results of HBI interaction. Tables 8 and 9 present the results of the classification for the non-Okutama Action.

As shown in Table 10, we performed an ablation study where we removed one component at a time to evaluate our model. Each row describes the model with an element missing, along with its accuracy on the ShakeFive2, HMDB51, and Okutama Action datasets. The table also illustrates the importance of each element in achieving high accuracy. Table 11 explains the method's complexity, execution time, energy efficiency, and error margins, providing insight into the system's computational time and performance. Execution time $T = \sum_{i=1}^{n} t_i$, where t_i represents the time for each step in the algorithm. Energy efficiency $E = P \times T/n$, where P is the power consumption of the system and n is the number of computations performed. The error margin (EM) deviation of the predicted value from the actual value is defined as $EM = \left| \frac{Y_{pred} - Y_{true}}{Y_{true}} \right| \times 100$, where Y_{pred} is the predicted value and Y_{true} is the true value.

Table 2
Detailed results of proposed system classification for the ShakeFive2 dataset.

Table classes	Results		
	Precision	Recall	F1-score
Explain	0.99	0.97	0.98
Fist bump	0.96	0.96	0.96
Handshake	0.98	0.98	0.98
High-five	0.99	0.97	0.98
Hug	1.00	0.94	0.97
Pass object	1.00	0.98	0.99
Rock-paper-scissors	0.99	0.98	0.98
Thumbs-up	0.88	0.99	0.93

Table 3
Confusion matrix for ShakeFive2.

Class label	Explain	Fist-bump	Hand-shake	High-five	Hug	Pass-object	Rock-paper-sizer	Thumbs-up
Explain	**0.97**	0.00	0.00	0.00	0.00	0.00	0.00	0.03
Fist bump	0.00	**0.96**	0.01	0.00	0.00	0.00	0.00	0.03
Handshake	0.00	0.00	**0.98**	0.00	0.00	0.00	0.00	0.02
High-five	0.00	0.01	0.00	**0.97**	0.00	0.00	0.00	0.02
Hug	0.00	0.02	0.01	0.01	**0.94**	0.00	0.01	0.01
Pass-object	0.00	0.01	0.00	0.00	0.00	**0.98**	0.00	0.01
Rock-paper-scissors	0.00	0.00	0.00	0.00	0.00	0.00	**0.98**	0.02
Thumbs-up	0.01	0.00	0.00	0.00	0.00	0.00	0.00	**0.99**

Table 4
Confusion matrix for HMDB51.

Class label	Catch	Clap	Fencing	Kick	Hug	Punch
Catch	**0.93**	0.07	0.00	0.00	0.00	0.00
Clap	0.00	**0.96**	0.00	0.00	0.02	0.02
Fencing	0.18	0.00	**0.82**	0.00	0.00	0.00
Kick	0.00	0.00	0.00	**0.85**	0.06	0.09
Hug	0.07	0.07	0.11	0.00	**0.57**	0.18
Punch	0.00	0.00	0.00	0.00	0.01	**0.99**

Table 5
Detailed results of proposed system classification for HMDB51 dataset.

Table classes	Results		
	Precision	Recall	F1-score
Catch	0.76	0.93	0.84
Clap	0.93	0.96	0.95
Fencing	0.75	0.82	0.78
Kick	1.00	0.85	0.92
Hug	0.80	0.57	0.67
Punch	0.91	0.99	0.95

Table 6
Classification for HBI Okutama Action dataset.

Table classes	Results		
	Precision	Recall	F1-score
Handshake	0.92	0.95	0.94
Hug	0.94	0.92	0.93

Table 7
Confusion matrix for HBI Okutama Action.

Class label	Hand-shake	Hug
Handshake	**0.95**	0.05
Hug	0.08	**0.93**

Table 8
Detailed result of proposed system classification for noninteraction.

Table classes	Results		
	Precision	Recall	F1-score
Running	0.75	0.93	0.83
Walking	0.79	0.72	0.76
Laying	0.67	0.75	0.71
Sitting	0.93	0.89	0.91
Standing	0.95	0.76	0.85

Table 9
Confusion matrix for noninteraction activities.

Class label	Running	Walking	Laying	Sitting	Standing
Running	**0.93**	0.03	0.00	0.00	0.03
Walking	0.14	**0.73**	0.09	0.05	0.00
Laying	0.06	0.19	**0.78**	0.00	0.00
Sitting	0.00	0.11	0.00	**0.89**	0.00
Standing	0.20	0.00	0.04	0.00	**0.76**

Table 10
Results of ablation experiment across different datasets for all the methods.

Experiments	Silhouette extraction	Preprocessing (HSV)	Fuzzy optimization	Feature extraction	HMM embedding (three layers)	Shake five2	HMDB51	Okutama Action
Full Model	✓	✓	✓	✓	✓	0.97	0.90	0.94
Without preprocessing (HSV)	✓	✗	✓	✓	✓	0.81	0.87	0.78
Without silhouette extraction	✗	✓	✗	✓	✓	0.68	0.69	0.64
Without feature extraction	✓	✓	✓	✗	✓	0.65	0.74	0.80
Without pre + feature	✓	✗	✓	✗	✓	0.63	0.72	0.78
Without fuzzy optimization	✓	✓	✗	✓	✓	0.68	0.78	0.79
Without HMM Embedding	✓	✓	✓	✓	✗	0.78	0.85	0.67

Table 11
Time complexity, execution time, energy efficiency, and error margins of the algorithm.

Methods	Time complexity	Execution time (s)	Energy efficiency (J)	Error margins (%)
Preprocessing	$O(n^{3/2})$, n =pixels	0.320	5.1	Low (3.0%)
Silhouette Extraction	$O(mn)$, m = objects, n = frames	0.440	6.0	Moderate (5.2%)
Feature Extraction	$O(kn\log n)$, k = features, n = frames	0.630	6.5	Moderate (4.8%)
Feature Optimization	$O(\log nk)$, k = iterations, n = features	0.940	6.6	Minimal (3.5%)
Action Detection	$O(mn^2)$, m = states, n = samples	0.920	7.7	Moderate (5.7%)

Table 12
Comparison table of the accuracies of various action recognition approaches.

Datasets	Methods	Accuracy (%)
Okutama Action	DronCaps[19]	0.47
	ResNeXt101[20]	0.60
	Prompt Learning for Action Recognition (PLAR)[21]	0.71
	Sparse Weighted Temporal Attention model[22]	0.72
	Our	0.94
Shakefive2	Deformable DPM parts Model[23]	65–87
	Spatiotemporal deformable[24]	0.82
	Spatiotemporal localization[25]	0.82
	Our	0.97
HMDB51	Autoencoders (VideoMAE)[26]	0.62
	VGGNet + ConvNets[27]	0.77
	Temporal Attention Vectors (TAVs)[28]	0.77
	Deep Belief Network[29]	0.87
	Our	**0.90**

3.6 Comparison with state-of-the-art methods

To validate the effectiveness of the proposed three-layer E-HMM framework, we conducted a comparative analysis with several state-of-the-art methods on three public datasets. Table 12 shows the performance of our approach alongside existing models, including graph convolutional network (GCN)-based architectures[22] and spatio-temporal localization methods.[25]

The performance evaluations in Table 13 show that while both the E-HMM and GCN approaches achieve high accuracy, the E-HMM provides a superior balance of competitive results and lower computational costs, making it ideal for real-time, resource-constrained applications. Parameters of ShakeFive2, Okutama Action, and HMdb51 were fine-tuned, and environmental noise, including changes in illumination and occlusions, was added to the training to achieve a high accuracy above 90%. This procedure reduced overfitting and improved generalization.

As shown in Table 14, the proposed three-layer E-HMM achieves a processing speed of 18.2 ms per frame, which is approximately 2.3 times higher than GCN-based models and nearly 5 times higher than ResNeXt101. Furthermore, the memory footprint is significantly lower (145

Table 13
Accuracy and execution time determined by GCN action recognition approaches.

Datasets	Recognition accuracy (%)		Execution time (s)	
	3-layer E-HMM	GCN	3-layer E-HMM	GCN
Shakefive2	0.97	0.84	18034.63	25034.76
Okutama action	0.94	0.84	19531.23	24022.35
HMDB51	0.90	0.88	26075.18	29148.28

Table 14
Computational efficiency comparison with existing methods during inference.

Methods	Accuracy (Avg)	Processing time (ms/frame)	Memory usage (MB)
ResNeXt101[20]	0.60	85.4	1240
GCN-based[22]	0.72	42.1	850
VideoMAE[26]	0.62	110.5	2100
Proposed 3-layer E-HMM	0.94	18.2	145

MB), confirming the suitability of the framework for deployment in resource-constrained edge devices and real-time remote monitoring systems. Beyond its technical performance, the proposed system holds substantial social significance. By achieving high recognition rates on lightweight architectures, the framework enables the deployment of proactive lifecare solutions in resource-constrained environments. This allows for the real-time monitoring of elderly populations or patients in remote areas, facilitating independent living while ensuring rapid response to critical behavioral events without the privacy concerns associated with cloud-based processing.

4. Conclusion

A three-layer E-HMM-based method for recognizing human interactions achieved high recognition rates of 97% on ShakeFive2 and 90% on HMDB51, as well as 94 and 82% on Okutama Action. The technique requires sequential steps, which start by improving frames, followed by silhouette extraction, then moves toward sensor and signal feature extraction, tracked by combined features and classification using the E-HMM. The proposed approach may enhance the accuracy and operational reliability of monitoring applications, including biometrics, surveillance, and human–computer interaction. The research provides crucial insights into human behavior relationships, generating fundamental knowledge for future social behavior analysis research and advanced system development to understand human interactions. This research demonstrates the potential of sensor-driven human behavior monitoring systems, establishing pathways for future studies in complex human activities. The social significance of this high-recognition system lies in its potential to bridge the gap between advanced behavior analysis and practical, everyday lifecare. As a lightweight solution, it provides a scalable foundation for socially responsible AI that enhances public safety and improves the quality of life for vulnerable populations through reliable, real-time behavior interaction monitoring. While promising, the method's robustness in dynamic environments and computational

efficiency for resource-constrained devices has certain limitations. As future work, we will investigate the integration of multiview data fusion and deep feature reinforcement to improve robustness in these challenging scenarios.

Acknowledgments

The publication was supported by the Open Access Initiative of the University of Bremen and the DFG via SuUB Bremen. We also acknowledge the Princess Nourah bint Abdulrahman University Researchers Supporting Project (No. PNURSP2026R440), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

References

1 Z. Li, Y. Wang, N. Zhang, Y. Zhang, Z. Zhao, D. Xu, G. Ben, and Y. Gao: Remote Sens. **14** (2022) 2385. https://doi.org/10.3390/rs14102385
2 J. Liu, J. Yang, Y. Zhang, and X. He: Proc. 2010 20th Int. Conf. Pattern Recognit. (2010) 3744–3747. http://hdl.handle.net/10453/16277
3 A. Manzi, P. Dario, and F. Cavallo: Sensors **17** (2017) 1100. https://doi.org/10.3390/s17051100
4 L. Köping, K. Shirahama, and M. Grzegorzek: Comput. Biol. Med. **95** (2018) 248. https://doi.org/10.1016/j.compbiomed.2017.12.025
5 G. Khodabandelou, H. Moon, Y. Amirat, and S. Mohammed: Eng. Appl. Artif. Intell. **118** (2023) 105702. https://doi.org/10.1016/j.engappai.2022.105702
6 X. Wu, W. Li, D. Hong, R. Tao, and Q. Du: IEEE Geosci. Remote Sens. Mag. **10** (2022) 91. https://doi.org/10.1109/MGRS.2021.3115137
7 S. Morales García, C. Henao Baena, and A. Calvo Salcedo: TecnoLógicas. **26** (2023) 56. https://doi.org/10.22430/22565337.2474
8 R.E. Nogales, and M.E. Benalcázar: Int. J. Comput. Intell. Syst. **16** (2023) 153. https://doi.org/10.1007/s44196-023-00319-1
9 E. M. G. Younis, S. Mohsen, E. H. Houssein, and O. A. S. Ibrahim: Neural Comput. Appl. **36** (2024) 8901. https://doi.org/10.1007/s00521-024-09426-2
10 G. James, D. Witten, T. Hastie, R. Tibshirani: Statistical learning, G. James, D. Witten, T. Hastie, and R. Tibshirani, Eds. (Springer New York, New York, 2023) pp. 15–67. https://doi.org/10.1007/978-3-031-38747-0_2
11 U. Oleh, R. Obermaisser, and A. S. Ahammed: Algorithms **17** (2024) 434. https://doi.org/10.3390/a17100434
12 P. K. Singh, S. Kundu, T. Adhikary, R. Sarkar, and D. Bhattacharjee: Arch. Comput. Methods Eng. **29** (2022) 2309. https://doi.org/10.1007/s11831-021-09681-9
13 M. Kaseris, I. Kostavelis, and S. Malassiotis: Mach. Learn. Knowl. Extr. **6** (2024) 842. https://doi.org/10.3390/make6020040
14 K. Hu, J. Jin, F. Zheng, L. Weng, and Y. Ding: Artif. Intell. Rev. **56** (2022) 1833. https://doi.org/10.1007/s10462-022-10210-8
15 T. Xue, and H. Liu: Lecture Notes in Electr. Eng. **878** (2022). https://doi.org/10.1007/978-981-19-0390-8_108
16 T. Kalsum, Z. Mehmood, F. Kulsoom, H. N. Chaudhry, A. R. Khan, M. Rashid, and T. Saba: J. Intell. Fuzzy Syst. **40** (2021) 9311. https://doi.org/10.3233/JIFS-201799
17 L. Zheng, M. Tang, Y. Chen, G. Zhu, J. Wang, and H. Lu: Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (2021) 2453–2462. https://doi.org/10.1109/CVPR46437.2021.00248
18 A. Zhou, W. Xie, and J. Pei: Remote Sens. **15** (2023) 2354. https://doi.org/10.3390/rs15092354
19 A. M. Algamdi, V. Sanchez, and C.-T. Li: Proc. 2020 IEEE Int. Conf. Image Process. (2020) 3174–3178. https://doi.org/10.1109/ICIP40778.2020.9190864
20 S. Kapoor, A. Sharma, A. Verma, and S. Singh: Pattern Recognit. **140** (2023) 109505. https://doi.org/10.1016/j.patcog.2023.109505
21 X. Wang, R. Xian, T. Guan, and D. Manocha: arXiv preprint arXiv:2305.12437 (2023). https://arxiv.org/abs/2305.12437
22 S. K. Yadav, A. Luthra, E. Pahwa, K. Tiwari, H. Rathore, H. M. Pandey, and P. Corcoran: Neural Networks **159** (2023) 57. https://doi.org/10.1016/j.neunet.2022.12.005

23 C. Van Gemeren, R. Poppe, and R. C. Proc. 2016 ACM Multimedia Conf. (2016) 35–44. https://webspace.science.uu.nl/~veltk101/publications/art/nccv2015_p35L.pdf
24 C. Van Gemeren, R. Poppe, and R.C. Veltkamp: Human Behavior Understanding, M. Chetouani, J. Cohn, and A. Salah, Eds. (Springer, Cham, 2016) pp. 116–133. https://doi.org/10.1007/978-3-319-46843-3_8
25 C. van Gemeren, R. Poppe, R. Veltkamp: J. Image Video Process. **16** (2018). https://doi.org/10.1186/s13640-018-0255-0
26 Z. Tong, Y. Song, J. Wang, and L. Wang: Adv. Neural Inf. Process. Syst. **35** (2022) 10078. https://doi.org/10.48550/arXiv.2203.12602
27 S. Zhao, Y. Liu, Y. Han, R. Hong, Q. Hu, and Q. Tian: IEEE Trans. Circuits Syst. Video Technol. **28** (2018) 1839–1849. https://doi.org/10.1109/TCSVT.2017.2682196
28 Y. Bo, Y. Lu, and W. He: Proc. 2020 IEEE Winter Conf. Appl. Comput. Vision (2020) 584–593. https://doi.org/10.1109/WACV45572.2020.9093481
29 B. Alabdullah, M. Tayyab, Y. AlQahtani, N. Al Mudawi, A. Algarni, A. Jalal, and J. Park: Comput. Mater. Continua **81** (2024) 309. https://doi.org/10.32604/cmc.2024.053538