SN Computer Science (2023) 4:621
https://doi.org/10.1007/s42979-023-02063-x

ORIGINAL RESEARCH

Keypoint Changes for Fast Human Activity Recognition

Shane Reid[1] · Sonya Coleman[1] · Dermot Kerr[1] · Philip Vance[1] · Siobhan O'Neill[2]

Received: 2 November 2021 / Accepted: 14 June 2023

Abstract
Human activity recognition has been an open problem in computer vision for almost 2 decades. During this time, there have been many approaches proposed to solve this problem, but very few have managed to solve it in a way that is sufficiently computationally efficient for real-time applications. Recently, this has changed, with keypoint-based methods demonstrating a high degree of accuracy with low computational cost. These approaches take a given image and return a set of joint locations for each individual within an image. In order to achieve real-time performance, a sparse representation of these features over a given time frame is required for classification. Previous methods have achieved this using a reduced number of keypoints, but this approach gives a less robust representation of the individual's body pose and may limit the types of activity that can be detected. We present a novel method for reducing the size of the feature set, by calculating the Euclidian distance and the direction of keypoint changes across a number of frames. This allows for a meaningful representation of the individuals movements over time. We show that this method achieves accuracy on par with current state-of-the-art methods, while demonstrating real-time performance.

Keywords Social signal processing · Activity recognition · MLP · Keypoints · Feature extraction

Introduction

The automated detection and classification of human activities from video have been an open problem in computer vision for over 2 decades. However, previous attempts to use computer vision to detect social signals come with a number of drawbacks. The computational cost, in particular, means most traditional methods do not scale well when there is a large number of individuals in a scene [1]. The recent emergence of deep learning techniques has enabled the development of methods for fast extraction of skeletal keypoints features, which (alongside continued increases in computing power) has been a breakthrough in enabling the development of real-time methods for human activity recognition that can readily scale to multiple people [2].

In Ref. [3], we presented a keypoint trajectories based approach from Ref. [4], where the set of keypoints for an individual, extracted over a given time period, is converted to a feature set of "keypoint changes". These keypoint changes encode a temporal history of the Euclidian distance and the direction of keypoint movement. We measured the keypoint changes using a reduced sample rate and reduced sample size, and we also measure the short-term keypoint changes between concurrent frames. In this way, we were able to maintain a sparse representation of an individual movement but were are also able to detect actions which are characterised by rapid movements. In this paper, we expand on this work, and present further results for the keypoint trajectories based approach from Ref. [3], performing

This article is part of the topical collection "Image Processing and Vision Engineering" guest edited by Sebastiano Battiato, Francisco Imai and Cosimo Distante.

✉ Shane Reid
Reid-S22@ulster.ac.uk

Sonya Coleman
sa.coleman@ulster.ac.uk

Dermot Kerr
d.kerr@ulster.ac.uk

Philip Vance
p.vance@ulster.ac.uk

Siobhan O'Neill
sm.oneill@ulster.ac.uk

[1] School of Computing, Engineering and Intelligent Systems, Ulster University Magee Campus, Derry, Londonderry, Northern Ireland

[2] School of Psychology, Ulster University Coleraine Campus, Coleraine, Northern Ireland

Published online: 14 August 2023

experiments with a range of additional machine learning methods in order to prove the robustness of these features for activity recognition. In addition, we conduct further experiments where we change the sample rate and the number of samples taken for each activity in order to obtain a better understanding of the impact that changing these values has on the classification accuracy. The remainder of the paper is organised as follows. In Sect. "Literature Review", we present the related literature. In Sect. "Methodology", we outline the proposed approach and the experimental design. In Sect. "Experimental Setup", we present the performance evaluation results and discussion. In Sect. "Accuracy Evaluation", we compare the results with other state-of-the-art methods, and in Sect. "Runtime Evaluation", we perform runtime analysis. Finally, in Sect. "Conclusion", we conclude the paper and discuss possible future work.

Literature Review

Human activity recognition, defined as the challenge of classifying an individual's activity from a video, is one of the oldest problems in the field of video processing. There have been a number of proposed approaches to solving this problem, with the majority based on either spatio-temporal features [5–7], optical flow [8–12] or deep learning [13–16]. These methods have been shown to achieve high accuracy on benchmark datasets but incur a significant computational cost. As such, their use for real-time applications is limited.

Feature extraction is an approach to reduce computational cost in image and video processing, for example, by compressing an image into a sparse set of interest points such as in Ref. [17]. Early attempts to do this used general interest point detectors such as SIFT and SURF. However, these methods had a number of drawbacks, most notably that there was no agreed standard for human representation [18]. To solve these problems, specialised "key point" detectors were developed, which can be applied to an image and a set of locations of key body joints for each individual within the image is returned. Two of the most popular approaches are OpenPose, which uses a bottom up approach based on part affinity fields [19], and AlphaPose, a top down approach based on the use of a technique known as pose flow [20], though recently transformer-based methods such as ViTPose [21] have demonstrated state-of-the-art performance for this task.

Generally, methods for activity recognition using keypoint features sample a number of consecutive frames, and then concatenate these values to form a feature vector, such as the method for fall detection implemented in Ref. [22]. Recently, Camarena et al. [17] presented an approach for fast human activity recognition based on the method used in [23]. In order to speed up this approach, they used a reduced feature set of six keypoints (those for the neck, right wrist, left elbow, left wrist, mid hip and left ankle), generated using OpenPose [19]. In doing so, they reduced the number of features used by approximately a factor of 5 and achieved an approximate 8 times improvement in speed over the original method [23], with a reduction in accuracy of only 1.4%. This enabled the approach to run sufficiently fast for real-time classification, a breakthrough for human activity recognition. In order to achieve this speed gain, their approach only sampled a small number of body keypoints. However, by doing this, they have a less generalizable representation of the individual's body pose; this may limit the type of activity that can be detected. For example, in a situation where it is necessary to detect whether an individual is kicking with their right leg, this approach would struggle as they have extracted no keypoints relating to the right leg. In contexts where it is necessary to detect a large range of different actions, using a reduced set of keypoints may not be feasible.

Recently, the work of Reid et al. [24] showed that by reducing the frame rate and sample size used for keypoint-based activity recognition, the computational cost can be reduced sufficiently to perform real-time activity recognition on upwards of 14 individuals simultaneously. However, this approach also comes with downsides, the most obvious of which is that by reducing the sample rate; in this way, it may be difficult to detect actions which are characterised by rapid movements, such as clapping, where the movement may be completed between frames being sampled. In addition, this method may be prone to translation variance and could struggle to detect activities that occur in different parts of the image, unless translated data are included in training. Earlier methods for overcoming this issue using traditional keypoints involved measuring keypoint trajectories, but such approaches are limited by the fact that they are unable to track specific landmarks (e.g. elbows and hands) [25]. Later improvements to such methods achieved impressive accuracies on a number of benchmark datasets but were still hampered by poor runtime performance [26]. Due the recent breakthroughs in the area of human landmark detection, keypoint trajectories are once again coming into focus as a viable method for human action recognition [27, 28].

Methodology

Where we will describe the proposed keypoint-based approach for fast human activity recognition, which is based on the temporal history of keypoint changes (in terms of the Euclidian distance and direction). We use OpenPose for keypoint extraction [19] as it provides a high level of accuracy with very low computational cost that remains constant

when more individuals are detected, unlike with other methods such as AlphaPose [20].

For each individual within an image, OpenPose extracts a set of 25 body keypoints as shown in Fig. 1. For these experiments, we use the full set of 25 keypoints, as this provides the most detailed representation of the human pose. OpenPose first uses a feedforward neural network to predict a set of 2D confidence maps of body part locations and a set of 2D vectors of part affinity fields (PAFs) which encode the degree of association between parts. These confidence maps and the PAFs are then parsed by a greedy inference method to output the 2D keypoints for all individuals in the image. For more details on the model architecture please see [19].

It is worth noting, however, that the novel contributions of this paper are not reliant on any specific keypoint estimation approach and can be implemented with any methods, such as AlphaPose [20], Megvii [29], or similar techniques. Regardless of the method used for keypoint extraction, each keypoint is defined as

$$k_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{1}$$

where x_i and y_i are the 2D image coordinates of the extracted keypoint. We represent an individual's movement as the Euclidian distance between two keypoints $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$ defined as

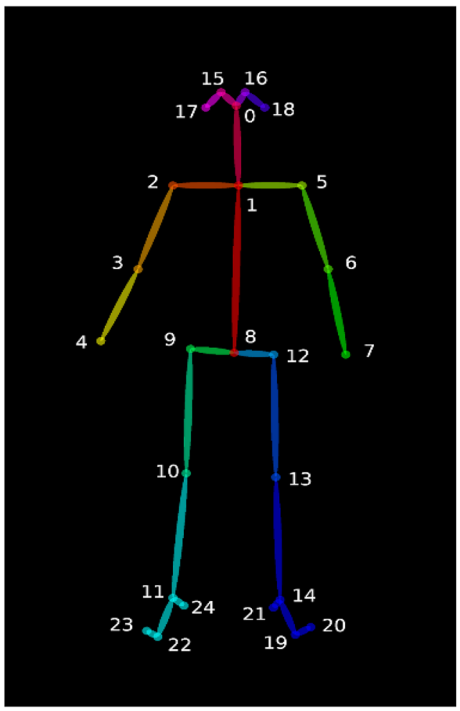

Fig. 1 The 25 keypoints extracted by OpenPose

$$\Delta(k_1, k_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2}$$

and the angle between them as

$$\theta(k_1, k_2) = \tan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{3}$$

where the tan function returns the unambiguous angle θ between the two keypoints on the Euclidian plane.

We can prove that these measurements are translation invariant as follows: suppose we define the value α as an arbitrary translation in the x coordinate, and the value β as another arbitrary translation in the y coordinate, we can define the translated pair of coordinates as

$$\begin{pmatrix} x_1 + \alpha \\ y_1 + \beta \end{pmatrix}\begin{pmatrix} x_2 + \alpha \\ y_2 + \beta \end{pmatrix} \tag{4}$$

then we can see that

$$(x_2 + \alpha) - (x_1 + \alpha) \equiv x_2 - x_1, \tag{5}$$

and by extension,

$$(y_2 + \beta) - (y_1 + \beta) \equiv y_2 - y_1 \tag{6}$$

meaning that the values for θ and Δ are invariant to translation in both the x and y coordinates. For two sets of γ keypoints, L and M, corresponding to an individual, the set of keypoint changes $\boldsymbol{C}$ is defined as

$$C(L, M) = \begin{Bmatrix} \Delta(L_1, M_1), \theta(L_1, M_1), \\ \Delta(L_2, M_2), \theta(L_2, M_2), \\ \dots \\ \Delta(L_\gamma, M_\gamma), \theta(L_\gamma, M_\gamma) \end{Bmatrix} \tag{7}$$

For our first set of experiments, we computed a coarse representation of the individual's movement by calculating n such sets of keypoint changes in order to build up a temporal history feature vector. The final feature vector at time t is defined as

$$\text{Coarse}_t = \begin{Bmatrix} \boldsymbol{C}(K_t, K_{t-m}), \\ \boldsymbol{C}(K_{t-m}, K_{t-2m}), \\ \dots \\ \boldsymbol{C}(K_{t-(n-1)m}, K_{t-nm}) \end{Bmatrix} \tag{8}$$

where K_t is the set of keypoints extracted for the video frame at time t and m is the fixed time difference between the two sampled frames. Therefore, for the second experiment, we attempted to compute a fine-grained representation of the keypoint changes, by measuring the keypoint changes from the intervening frames. This enables us to compute a more fine-grained representation of an individual's movement. Again, a set of n such keypoint changes is used in order to

build up a temporal history of the individuals' movement over time. This feature vector is defined as

$$\text{Fine}_t = \left\{ \begin{array}{l} \boldsymbol{C}\left(K_t, K_{t-\varepsilon}\right), \\ \boldsymbol{C}\left(K_{t-m}, K_{t-m-\varepsilon}\right), \\ \ldots \\ \boldsymbol{C}\left(K_{t-(n-1)m}, K_{t-nm-\varepsilon}\right) \end{array} \right\} \tag{9}$$

where ε is defined as a short time period such that $\varepsilon < m$. In the experiments, $\varepsilon = 1$. This enabled a fine-grained representation of the instantaneous keypoint changes.

For the final set of experiments, we test whether an approach trained on both short-term and long-term keypoint changes would be more effective than using either one individually. This would increase the number of features used; however, the representation would still be sparser than sampling every individual frame. For this experiment, the combined approach is represented as

$$\text{Combined}_t = \left\{\text{Fine}_t, \text{Coarse}_t\right\} \tag{10}$$

We used this methodology to investigate how changing the sample rate and number of frames impacted classification accuracy. To do this, we conducted a range of experiments using a number of different sample rates (rate at which frames are sampled, measured in frames per second) and sample size (number of frames sampled, defined as n in (9) and (10)). The set of sample rates used, measured in fps, were as follows:

$$\text{Sample Rate} = \{25.0, 12.5, 8.3, 5.0, 2.5, 1.6, 1.0, 0.5\} \tag{11}$$

Since the original frame rate of the dataset used was 25 fps, the corresponding values used for m in Eqs. (8) and (9) are as follows:

$$m = \{1, 2, 3, 5, 10, 15, 25, 50\} \tag{12}$$

The sequence length refers to the total number of frames subsampled. We use different sequence lengths in order to determine the optimal activity time period and to allow us to compare different sample rates over different time periods. The set of sequence lengths corresponds to the values used for n in Eqs. (8) and (9) is as follows:

$$\text{Sequence Length} = \{1, 3, 5, 10, 15, 30, 50, 75, 100\} \tag{13}$$

The longer the sequence length is, the longer the feature vector used for training and inference. These feature vectors were then used to train the following ML Classifiers: *k*NN, XGBoost, Multi-layer Perception (MLP), Classification and Regression Tree (CART), Random Forest and Support Vector Classification (SVC). The activity type (i.e. walking, jogging, and running) was used as the classification label. All reported accuracies used fivefold cross-validation.

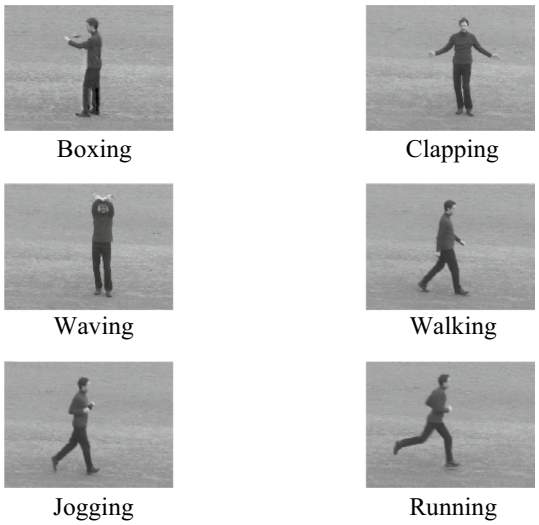

Fig. 2 Example frames of the six activities in the KTH dataset

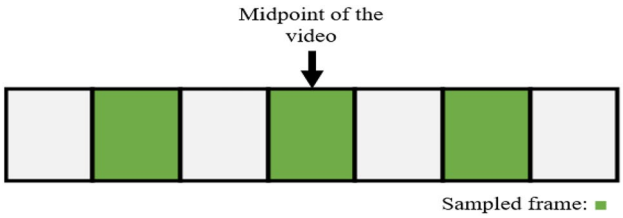

Fig. 3 Sampling strategy for the dataset

Experimental Setup

We evaluated the approaches using the well-established KTH dataset [8]. This dataset contains short video clips of 6 distinct actions: Walking, Jogging, Running, Boxing, Clapping and Waving. For each activity there are 25 sets of videos each containing a different individual. Each video set contains 4 videos, each with a different background: outdoor, outdoor with a different scale, outdoor with different clothes and indoor. This results in a total of 600 video clips, with an average length of 4 s, recorded at a rate of 25 fps. The videos have a resolution of 160×120 pixels. Figure 2 shows example frames from the dataset.

For each combination of sample rate and sequence length, a set of keypoint changes was taken from each video starting from the middle and working outwards, with an equal number of frames sampled before and after the midpoint as shown in Fig. 3. In this way, we ensure that all feature vectors contain the peak of the activity sequence and thus can evaluate the impact of changing the sample rate and sample size across an activity. These feature vectors were then used to train the six machine learning

models for each experimental setup to classify the activity. Hyperparameter tuning was performed to optimise each model using a grid search in order to find the best parameters. The values used for the grid search are outlined in Table 1.

Accuracy Evaluation

A. Long-term keypoint changes

Results using each of the machine learning methods with long-term keypoint changes are presented in Tables 2, 3, 4, 5, 6, and 7. We can see that with the long-term keypoint changes approach, the model accuracy does not necessarily improve as the time period is increased. The results indicate these methods are more sensitive to changes in the sequence length and the sample rate. The highest accuracy achieved for the Random Forest approach was 90.64%. The SVC and KNN approaches both performed much worse when compared to the XGBoost model. The SVC approach fails to achieve an accuracy of > 65% for any configuration of sequence length and sample rate. The *k*NN approach achieves an accuracy of over 70% for some configurations; however, a number of combinations fail to achieve 50% accuracy. The results for the CART, XGBoost and MLP classifiers are presented in Tables 14, 15 and 16. It can be seen that the CART approach performs inconsistently with the long-term keypoint changes approach. The

Table 1 Hyperparameters used for the grid search

Model	Parameter	Values
SVC	Kernel	Radial basis function, Linear, Polynomial
	C	1, 10, 100, 1000
	Gamma	0.001, 0.0001
KNN	Neighbours	5, 7, 9, 11
	Distance metric	Euclidian, Manhattan, Chebyshev
Decision tree	Splitter	Best, Random
	Max depth	None, 5, 10
	Criterion	Gini, Entropy
	Minimum samples per split	2, 5, 7
Random Forest	Estimators	10,50,100
	Criterion	Gini, Entropy
	Max depth	None, 5, 10
	Minimum samples per split	2, 5, 7
	Bootstrap	True, False
MLP	Activation	Identity, Logistic, Tanh, Relu
	Solver	lbfgs, sgd, adam
	Learning rate	Constant, scaling
XGBoost	Min child weight	5, 10
	Gamma	1, 1.5, 2
	Subsample	0.8, 1.0
	Colsamble by tree	0.8, 1.0
	Max depth	3, 5

Table 2 Long-term keypoint changes accuracy with Random Forest

		Random Forest							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	64.61%	64.61%	64.61%	64.61%	64.61%	64.61%	64.61%	61.44%
	2	68.28%	74.29%	78.79%	82.47%	82.97%	79.47%	64.94%	72.12%
	3	82.80%	85.81%	87.31%	87.65%	82.97%	70.44%	80.47%	83.64%
	5	88.81%	89.48%	88.82%	85.47%	77.13%	83.31%	85.81%	88.31%
	10	89.64%	88.48%	85.97%	81.64%	86.31%	86.14%	88.31%	89.31%
	15	88.65%	85.47%	84.14%	86.98%	87.14%	88.98%	90.15%	89.81%
	25	85.30%	85.13%	87.31%	87.14%	88.47%	90.64%	89.64%	87.97%
	50	85.97%	86.81%	87.47%	89.31%	89.31%	90.15%	87.65%	84.47%

Table 3 Long-term keypoint changes accuracy with SVC

		SVC							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	57.10%	57.10%	57.10%	57.10%	57.10%	57.10%	57.10%	30.06%
	2	35.55%	37.40%	42.41%	43.90%	50.43%	50.91%	30.38%	33.72%
	3	39.40%	44.59%	49.92%	56.27%	59.94%	34.56%	39.57%	45.74%
	5	54.92%	61.77%	64.11%	63.60%	35.55%	41.25%	48.91%	51.25%
	10	63.26%	57.76%	64.95%	44.23%	56.09%	61.60%	63.94%	62.94%
	15	61.78%	62.77%	47.92%	61.95%	63.77%	60.09%	61.78%	64.28%
	25	62.77%	52.42%	60.77%	58.26%	58.60%	64.78%	62.77%	62.10%
	50	50.09%	56.42%	61.76%	61.27%	62.61%	64.28%	62.11%	62.77%

Table 4 Long-term keypoint changes accuracy with kNN

		KNN							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	59.10%	59.10%	59.10%	59.10%	59.10%	59.10%	59.10%	32.06%
	2	39.22%	44.06%	52.77%	52.25%	54.09%	54.09%	31.39%	43.74%
	3	49.24%	57.93%	58.10%	63.11%	61.26%	32.88%	44.07%	56.76%
	5	69.78%	70.61%	63.28%	61.27%	36.39%	48.09%	59.10%	70.95%
	10	69.28%	57.26%	66.28%	39.91%	68.11%	75.12%	75.29%	54.77%
	15	57.94%	65.45%	50.25%	77.29%	67.12%	46.59%	43.08%	49.60%
	25	65.45%	67.95%	55.26%	40.73%	37.07%	43.74%	48.76%	60.44%
	50	71.62%	38.73%	33.72%	37.23%	42.58%	49.60%	57.94%	65.45%

Table 5 Long-term keypoint changes accuracy with CART

		CART							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	57.43%	57.43%	57.43%	57.43%	57.43%	57.43%	57.43%	49.10%
	2	57.10%	59.44%	67.44%	69.11%	69.95%	65.77%	49.58%	55.09%
	3	67.61%	73.61%	75.12%	77.13%	74.29%	49.75%	61.76%	67.45%
	5	73.46%	75.13%	75.96%	76.31%	55.75%	64.93%	68.95%	70.95%
	10	76.12%	75.45%	73.97%	61.10%	63.44%	67.61%	71.45%	78.30%
	15	75.45%	76.47%	59.93%	62.77%	70.94%	72.11%	75.29%	75.95%
	25	76.98%	57.60%	67.62%	70.29%	72.78%	77.13%	76.30%	75.96%
	50	61.45%	67.79%	71.78%	70.62%	75.46%	76.45%	76.46%	75.97%

Table 6 Long-term keypoint changes accuracy with XGBoost

		XGBoost							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	66.3%	66.3%	66.3%	66.3%	66.3%	66.3%	66.3%	64.6%
	2	66.4%	72.1%	77.3%	79.5%	79.8%	76.6%	64.3%	69.8%
	3	80.8%	85.5%	85.6%	86.6%	81.1%	69.6%	80.5%	83.0%
	5	88.5%	88.0%	87.8%	83.8%	75.8%	82.8%	84.6%	86.8%
	10	88.6%	88.0%	83.6%	81.8%	85.1%	85.3%	88.3%	89.3%
	15	88.5%	84.5%	83.5%	85.5%	87.3%	88.6%	89.1%	89.5%
	25	84.5%	82.8%	85.5%	88.8%	88.1%	88.0%	88.6%	87.5%
	50	83.6%	86.8%	87.8%	88.1%	88.8%	89.5%	88.0%	84.6%

accuracy was sensitive to sequence length and sample rate changes, but generally performed significantly better at the lower sample rates and sequence lengths. Finally, the MLP approach performed poorly when compared with the other approaches, failing to achieve an accuracy of over 70% for any combination of sequence length and sample rate.

B. Short-term keypoint changes

Table 7 Long-term keypoint changes accuracy with MLP

		MLP							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	59.93%	55.42%	58.76%	58.42%	58.26%	60.43%	57.92%	30.40%
	2	38.40%	39.73%	44.42%	48.75%	51.42%	53.92%	30.88%	33.05%
	3	38.74%	47.09%	54.09%	54.60%	58.44%	33.05%	41.24%	43.91%
	5	54.27%	60.27%	61.43%	64.28%	32.71%	36.56%	44.56%	47.24%
	10	59.26%	57.10%	66.45%	34.06%	45.41%	51.75%	54.09%	59.43%
	15	61.61%	66.61%	40.57%	48.24%	48.74%	53.25%	59.27%	64.61%
	25	68.11%	42.08%	46.57%	51.59%	54.60%	62.78%	63.11%	61.93%
	50	42.42%	47.41%	52.41%	57.93%	60.60%	65.94%	62.28%	67.78%

Table 8 Short-term keypoint changes accuracy with Random Forest

		Random Forest							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	61.44%	62.78%	65.12%	68.44%	72.95%	70.78%	70.78%	70.11%
	2	64.94%	67.95%	73.79%	75.30%	77.29%	77.45%	78.62%	78.97%
	3	70.44%	77.46%	78.80%	81.80%	82.30%	82.12%	82.64%	81.47%
	5	77.13%	81.30%	82.30%	82.63%	82.80%	84.31%	81.97%	80.81%
	10	81.64%	82.97%	83.97%	85.30%	84.97%	85.47%	83.64%	80.96%
	15	84.14%	86.14%	86.14%	85.47%	85.98%	84.64%	82.64%	80.80%
	25	85.13%	84.47%	86.64%	85.97%	86.14%	84.31%	83.14%	79.97%
	50	85.64%	85.81%	85.97%	86.47%	84.48%	83.31%	81.80%	80.63%

Table 9 Short-term keypoint changes accuracy with SVC

		SVC							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	30.06%	32.88%	30.22%	31.38%	27.72%	28.55%	29.22%	29.56%
	2	30.38%	29.38%	32.56%	30.89%	31.56%	32.89%	28.73%	35.56%
	3	34.56%	34.71%	33.39%	35.88%	32.72%	33.05%	38.40%	40.90%
	5	35.55%	33.38%	37.39%	35.39%	36.74%	45.25%	41.91%	44.92%
	10	44.23%	47.08%	44.08%	47.92%	44.73%	52.08%	52.59%	44.90%
	15	47.92%	51.42%	53.58%	53.42%	56.59%	54.26%	52.42%	44.90%
	25	52.42%	56.76%	57.10%	54.75%	57.77%	57.26%	50.09%	45.09%
	50	50.26%	54.43%	54.08%	56.60%	56.76%	54.26%	52.42%	44.90%

Table 10 Short-term keypoint changes accuracy with kNN

		*k*NN							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	32.06%	33.05%	34.23%	35.72%	34.06%	34.38%	35.90%	33.06%
	2	31.39%	32.89%	34.72%	38.57%	40.90%	48.75%	44.07%	30.05%
	3	32.88%	39.22%	40.91%	50.58%	61.10%	50.91%	26.21%	22.21%
	5	36.39%	38.41%	49.08%	62.44%	48.41%	27.88%	21.20%	21.54%
	10	39.91%	60.93%	70.62%	42.90%	24.04%	22.86%	22.37%	22.21%
	15	50.25%	74.12%	42.57%	24.87%	23.87%	22.86%	22.37%	22.21%
	25	67.95%	39.40%	23.71%	24.54%	23.20%	23.20%	21.20%	21.54%
	50	72.46%	26.54%	23.37%	23.87%	23.87%	22.86%	22.37%	22.21%

The results for the short-term keypoint changes along with the six machine learning approaches are presented in Tables 8, 9, 10, 11, 12, and 13. In Table 8, we can see that for the Random Forest, the short-term keypoints perform slightly better for some of the shorter sequence lengths, but generally performs slightly worse overall than the long-term keypoint approach. The SVC performs worse with the short keypoint changes, achieving accuracy of below 60% for all configurations compared with using long-term

Table 11 Short-term keypoint changes accuracy with CART

		CART							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	49.10%	49.24%	48.92%	50.93%	56.26%	53.75%	56.76%	54.76%
	2	49.58%	48.41%	56.59%	55.09%	56.42%	56.60%	58.60%	60.11%
	3	49.75%	55.60%	59.09%	59.59%	57.93%	64.43%	62.60%	59.93%
	5	55.75%	59.94%	62.28%	61.76%	59.77%	62.61%	60.45%	62.77%
	10	61.10%	59.43%	62.59%	61.43%	63.10%	61.77%	64.94%	58.94%
	15	59.93%	63.11%	62.44%	65.93%	63.60%	62.60%	65.77%	58.42%
	25	57.60%	61.44%	62.43%	63.94%	64.11%	62.93%	62.94%	63.11%
	50	61.45%	63.60%	62.10%	63.44%	63.28%	63.77%	64.94%	59.42%

Table 12 Short-term keypoint changes accuracy with XGBoost

		XGBoost							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	64.60%	63.60%	66.46%	68.62%	75.47%	73.29%	74.46%	74.13%
	2	64.27%	67.95%	72.46%	75.96%	80.13%	80.96%	81.47%	81.31%
	3	69.61%	77.46%	78.96%	83.14%	81.96%	84.29%	83.30%	81.47%
	5	75.79%	82.81%	82.64%	83.80%	85.65%	84.63%	83.13%	82.47%
	10	81.80%	81.97%	82.97%	83.64%	84.47%	86.31%	84.30%	81.46%
	15	83.47%	83.64%	83.81%	84.97%	84.64%	86.47%	84.64%	81.13%
	25	82.80%	86.64%	85.97%	86.31%	85.47%	84.97%	83.98%	82.31%
	50	83.97%	84.64%	85.98%	84.81%	86.47%	84.30%	81.63%	86.14%

Table 13 Short-term keypoint changes accuracy with MLP

		MLP							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	30.56%	33.56%	34.23%	34.23%	30.88%	30.73%	33.06%	31.21%
	2	29.22%	33.72%	33.55%	31.21%	32.39%	34.90%	32.06%	40.23%
	3	33.39%	30.20%	29.21%	33.55%	33.39%	35.89%	34.22%	39.06%
	5	32.88%	31.39%	37.56%	35.23%	35.56%	39.73%	33.06%	32.22%
	10	36.38%	40.24%	35.73%	38.73%	36.88%	35.38%	33.39%	36.56%
	15	39.73%	39.40%	40.07%	40.24%	37.72%	36.55%	34.56%	38.23%
	25	40.24%	40.40%	45.41%	42.91%	36.57%	36.73%	34.05%	35.06%
	50	40.24%	45.92%	43.74%	42.57%	40.73%	34.89%	35.05%	36.56%

keypoint changes. This SVC models find it difficult to classify short-term keypoint changes into respective activity classes. The *k*NN provides interesting results. As the time period increased, the accuracy diminished to below 30% for most configurations; however, for some of the configurations, the accuracy was over 74%. The CART and XGBoost both performed quite well when compared with the other approaches on the short-term keypoint changes. The CART performed quite poorly when compared with the long-term keypoint changes models, achieving a maximum accuracy of 65.93% and there is little consistency to the results. The XGBoost approach performed quite similarly to how it performed using long-term keypoint changes, albeit with a slightly lower accuracy across all configurations. However, when compared to the other short-term approaches this method performed the best. The MLP again performs very poorly using the short-term keypoint changes. This method achieved accuracies of $\leq 45.92\%$ for all configurations.

C. Combining long- and short-term keypoints

The results for each of the models using a combination of both short- and long-term keypoint changes are presented in Tables 14, 15, 16, 17, 18, and 19. We can see that the Random Forest performs on par with the long- and short-term approaches individually, achieving high accuracy for most configurations. The SVC performs better than when using the short- and long-term keypoint changes separately; however, the results are still significantly less than the other approaches.

Table 14 Combined short- and long-term keypoint changes accuracy with Random Forest

		Random Forest							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	62.61%	64.28%	70.11%	77.13%	81.80%	84.30%	85.47%	86.64%
	2	64.95%	72.62%	80.30%	83.81%	86.31%	86.47%	86.31%	87.31%
	3	68.95%	76.46%	83.30%	86.47%	86.98%	86.97%	87.14%	86.97%
	5	74.12%	82.97%	87.30%	87.64%	87.47%	88.14%	88.64%	88.64%
	10	82.30%	86.47%	88.31%	89.98%	88.48%	89.32%	90.31%	89.81%
	15	85.64%	88.81%	88.64%	90.48%	89.15%	89.81%	89.14%	89.65%
	25	84.81%	87.98%	89.15%	89.14%	89.32%	89.32%	87.98%	88.48%
	50	81.14%	84.80%	86.31%	85.81%	86.64%	86.64%	84.64%	85.31%

Table 15 Combined short- and long-term keypoint changes accuracy with SVC

		SVC							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	29.89%	29.88%	33.56%	35.39%	40.23%	45.91%	51.58%	55.10%
	2	35.39%	33.05%	43.23%	43.07%	53.09%	52.09%	60.27%	60.76%
	3	31.88%	35.22%	46.24%	46.74%	59.44%	57.43%	61.77%	62.59%
	5	32.89%	38.40%	48.08%	51.60%	59.76%	61.27%	58.59%	62.76%
	10	34.90%	40.58%	50.75%	57.76%	55.76%	60.10%	63.78%	61.27%
	15	35.73%	43.74%	52.93%	62.60%	57.77%	59.78%	59.10%	59.44%
	25	39.41%	46.09%	56.43%	53.43%	57.77%	57.77%	58.77%	57.77%
	50	44.91%	54.92%	57.60%	59.44%	58.43%	58.43%	59.78%	58.43%

Table 16 Combined short- and long-term keypoint changes accuracy with kNN

		*k*NN							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	32.06%	31.39%	32.88%	36.39%	39.91%	50.25%	67.95%	71.96%
	2	34.88%	36.73%	42.56%	41.24%	64.94%	76.96%	47.91%	31.88%
	3	36.23%	39.40%	47.59%	54.76%	76.96%	55.27%	30.71%	27.71%
	5	40.73%	44.24%	60.93%	69.28%	65.11%	33.89%	28.38%	28.88%
	10	45.74%	54.43%	71.28%	72.95%	35.40%	32.22%	32.39%	32.06%
	15	53.93%	61.78%	72.12%	56.10%	34.57%	33.90%	35.73%	33.90%
	25	50.92%	64.94%	48.26%	39.57%	38.40%	38.40%	38.40%	38.40%
	50	52.75%	51.60%	51.42%	56.26%	57.60%	57.60%	56.26%	57.60%

Table 17 Combined short- and long-term keypoint changes accuracy with CART

		CART							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	49.26%	49.58%	50.25%	56.59%	60.93%	60.27%	59.26%	62.45%
	2	50.77%	54.25%	61.11%	65.77%	64.44%	66.11%	65.45%	65.78%
	3	53.43%	58.10%	66.12%	67.45%	64.78%	69.95%	67.29%	70.62%
	5	60.94%	67.11%	69.61%	71.78%	70.61%	73.12%	71.28%	71.45%
	10	68.27%	71.61%	73.46%	74.79%	76.79%	75.12%	77.46%	74.96%
	15	68.28%	75.11%	74.11%	74.96%	75.62%	74.45%	76.46%	74.61%
	25	72.63%	76.13%	76.29%	74.12%	75.62%	75.12%	74.62%	75.95%
	50	68.11%	71.79%	71.95%	76.46%	71.95%	71.79%	74.96%	72.29%

The *k*NN performed similarly as to when it was applied using the short-term keypoint changes, where the accuracy was quite low when the time period increased. The results for the CART are presented in Table 17. We can see that CART achieved reasonable accuracy when compared with the other approaches; however, there was no major improvement over the use of individual short-term or long-term keypoint changes. The results for the XGBoost are presented in Table 18 where we can see the combined keypoint changes performed well with a maximum accuracy of 89.98%.

Table 18 Combined short- and long-term keypoint changes accuracy with XGBoost

		XGBoost							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	64.77%	64.60%	69.61%	76.96%	81.97%	83.47%	83.97%	82.81%
	2	64.26%	70.11%	79.80%	83.30%	84.81%	85.14%	85.31%	85.98%
	3	66.78%	73.79%	83.64%	85.47%	84.97%	86.32%	86.98%	86.48%
	5	73.11%	81.47%	86.30%	87.97%	87.31%	88.98%	87.98%	88.48%
	10	80.80%	86.48%	86.97%	87.47%	89.31%	89.48%	88.64%	89.15%
	15	83.63%	88.14%	88.64%	89.98%	89.48%	89.31%	88.64%	89.65%
	25	84.97%	87.31%	87.15%	88.15%	87.81%	87.98%	88.48%	87.98%
	50	80.31%	83.81%	85.98%	86.48%	84.81%	86.31%	86.48%	85.81%

Table 19 Combined short- and long-term keypoint changes accuracy with MLP

		MLP							
		Sequence length							
		3	5	10	15	30	50	75	100
Sample rate	1	32.06%	33.55%	33.72%	32.73%	34.90%	39.57%	41.40%	42.24%
	2	33.05%	33.39%	37.39%	37.90%	43.57%	44.57%	45.75%	51.58%
	3	33.39%	35.72%	41.40%	41.57%	44.24%	48.91%	52.10%	55.58%
	5	32.88%	35.90%	41.22%	44.41%	51.09%	55.93%	50.58%	59.43%
	10	35.06%	44.91%	47.92%	47.74%	52.59%	54.76%	57.60%	59.09%
	15	39.73%	48.08%	47.41%	50.58%	57.76%	55.09%	59.61%	58.59%
	25	45.08%	44.74%	50.24%	54.59%	58.60%	57.77%	61.27%	59.77%
	50	47.07%	50.42%	53.42%	57.09%	60.10%	58.43%	58.61%	59.26%

Finally, the results for the MLP are presented in Table 19. We can see that this approach performed poorer than the other ML models.

Runtime Evaluation

In Ref. [3], we computed the computation time of the combined approach using the Weizmann dataset which consists of 5701 frames. Experiments were conducted on an Intel XeonE5-1620 PC running Ubuntu version 18.04.3. The GPU used was a Nvidia Titan Xp with 16 GB RAM. This is consistent with other approaches such as Camarena et al. [17] who also used GPU accelerated hardware when testing the runtime of their approach. The time taken for the OpenPose library to compute the keypoints for the entire dataset was 227.3 s. This is a rate of 39.8 ms per frame and represents the most significant bottleneck of this approach. The time taken to compute the set of keypoint changes for the entire dataset is 1.7 s, approximately, 0.3 ms per frame. We found that the MLP algorithm took 1 s to classify the activities for the test set, which consisted of 701 frames. Therefore, classification is performed at a rate of 1.39 ms. The average computation time for the entire pipeline is 41.5 ms per frame, 24.0 frames per second. The runtime for the KTH dataset was also calculated and found to be the same. Hence, the approach is fast enough to perform activity recognition in real time.

Table 20 Comparison of approaches on the KTH dataset

Performance evaluation using the KTH dataset		
Approach	Accuracy (%)	Speed/FPS
[23]	95.7	3
[24]	90.20	24
Keypoint changes	90.48	24

Table 20 presents comparative results for the proposed approach and other state-of-the-art approaches using the KTH dataset. Table 7 shows that the approach of Wang et al. [23] achieves an accuracy of 95.7%. While this is higher than the proposed approach, the computational cost of this method prevents it from running in real time. We also compare our approach with that in Reid et al. [24] who used a reduced sample rate and sample size to achieve real-time performance using body keypoints, with the results demonstrating that the proposed approach performs better.

Conclusion

We have presented a number of experiments for human activity recognition based on calculating the keypoints changes (Euclidean distance and angle). We have also investigated how adjusting the number of samples, and the sample rate can affect the accuracy obtained when training a number of machine learning models for activity recognition. Further,

we have conducted runtime experiments and shown that this method is sufficiently fast for real-time applications. In the future work, we will investigate how this approach performs for multi-person activity recognition and adapt this approach for more complex activities and scenes.

Data availability Data used for this article can be obtained through implementation of the methods using the open source datasets and methods decribed.

Declarations

Conflict of interest The authors declare there is no conflice of interest.

References

1. Luo F, Poslad S, Bodanese E. Temporal convolutional networks for multiperson activity recognition using a 2-D LIDAR. IEEE Internet Things J. 2020;7(8):7432–42. https://doi.org/10.1109/JIOT.2020.2984544.
2. Minh Dang L, Min K, Wang H, Jalil-Piran M, Hee-Lee C, Moon H. Sensor-based and vision-based human activity recognition: a comprehensive survey. Pattern Recognit. 2020;108:1061. https://doi.org/10.1016/J.PATCOG.2020.107561.
3. Reid S, Coleman S, Kerr D, Vance P, O'neill S. Fast human activity recognition. Multimedia Model. 2015. https://doi.org/10.5220/0010420300910098.
4. Reid S, Vance P, Coleman S, Kerr D, O'Neill S. Towards real-time activity recognition. IEEE Conf Image Process Appl. 2020. https://doi.org/10.1109/IPAS50080.2020.9334948.
5. Zelnik-Manor L, Irani M. Event-based analysis of video. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2001;2(1229):123–30. https://doi.org/10.1109/cvpr.2001.990935.
6. Dollar P, Rabaud V, Cottrell G, Belongie S. Behavior recognition via sparse spatio-temporal feature. In: 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005, pp 65–72.
7. Laptev I. Local spatio-temporal image features for motion interpretation. Doctoral Thesis, Stockholm. 2004.
8. Schüldt C, Laptev I, Caputo B. Recognizing human actions: a local SVM approach. Proc Int Conf Pattern Recognit (ICPR). 2004;3:32–6. https://doi.org/10.1109/ICPR.2004.1334462.
9. Wang H, Kläser A, Schmid C, Liu CL. Action recognition by dense trajectories. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2011. https://doi.org/10.1109/CVPR.2011.5995407.
10. Ke Y, Sukthankar R, Hebert M. Efficient visual event detection using volumetric features. Tenth IEEE Int Conf Comput Vis (ICCV'05). 2005. https://doi.org/10.1109/CVPR.2007.383137.
11. Efros AA, Berg AC, Mori G, Malik J. Recognising action at a distance. Proc Ninth IEEE Int Conf Comput Vis. 2003;2:726–33. https://doi.org/10.1017/s1358246107000136.
12. Guo K, Ishwar P, Konrad J. Action recognition using sparse representation on covariance manifolds of optical flow. Proc IEEE Int Conf Adv Video Signal Based Surveill. 2010;2010:188–95. https://doi.org/10.1109/AVSS.2010.71.
13. D'Sa AG, Prasad BG. An IoT based framework for activity recognition using deep learning technique. 2019. http://arxiv.org/abs/1906.07247.
14. Lee DG, Lee SW. Prediction of partially observed human activity based on pre-trained deep representation. Pattern Recognit. 2019;85:198–206. https://doi.org/10.1016/j.patcog.2018.08.006.
15. Sheeba PT, Murugan S. Fuzzy dragon deep belief neural network for activity recognition using hierarchical skeleton features. Evol Intell. 2019. https://doi.org/10.1007/s12065-019-00245-2.
16. Subedar M, Krishnan R, Meyer PL, Tickoo O, Huang J. Uncertainty-aware audiovisual activity recognition using deep bayesian variational inference. In: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6301–10.
17. Camarena F, Chang L, Gonzalez-Mendoza M. Improving the dense trajectories approach towards efficient recognition of simple human activities. In: 2019 7th International Workshop on Biometrics and Forensics (IWBF), 2019, pp. 1–6.
18. Sun J, Mu Y, Yan S, Cheong LF. Activity recognition using dense long-duration trajectories. IEEE Int Conf Multimed Expo (ICME). 2010. https://doi.org/10.1109/ICME.2010.5583046.
19. Cao Z, Simon T, Wei SE, Sheikh Y. Realtime multi-person 2D pose estimation using part affinity fields. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 7291–9.
20. Xiu Y, Wang H, Lu C. Pose flow: efficient online pose tracking. In: British Machine Vision Conference, 2018, pp. 1–12.
21. Xu Y, Zhang J, Zhang Q, Tao D. ViTPose: simple vision transformer baselines for human pose estimation. 2022. arXiv preprint arXiv:2204.12484.
22. Ramirez H, Velastin SA, Aguayo P, Fabregas E, Farias G. Human activity recognition by sequences of skeleton features. Sensors. 2022;22(11):3991.
23. Wang H, Kläser A, Schmid C, Liu C. Dense trajectories and motion boundary descriptors for action recognition. Int J Comput Vis. 2013;103:60–79. https://doi.org/10.1007/s11263-012-0594-8.
24. Reid S, Vance P, Coleman S, Kerr D, O'Neill S. Towards real time activity recognition. In: Procedings of the Fourth IEEE International Conference on Image Processing, Applications and Systems (IPAS 2020), 2020.
25. Matikainen P, Hebert M, Sukthankar R. Trajectons: action recognition through the motion analysis of tracked features. IEEE Twelth Int Conf Comput Vis Workshop (ICCV Workshop). 2009. https://doi.org/10.1109/ICCVW.2009.5457659.
26. Jain M, Jégou H, Bouthemy P. Better exploiting motion for better action recognition. IEEE Conf Comput Vis Pattern Recognit. 2013. https://doi.org/10.1109/CVPR.2013.330.
27. Choutas V, Weinzaepfel P, Revaud J, Schmid C. PoTion: pose motion representation for action recognition. IEEE conference on computer vision and pattern recognition (cvpr). 2023.
28. Yi Y, Wang H. Motion keypoint trajectory and covariance descriptor for human action recognition. Vis Comput. 2018;34(3):391–403. https://doi.org/10.1007/s00371-016-1345-6.
29. Cai Y et al. Res-steps-net for multi-person pose estimation. In: Joint COCO and Mapillary Workshop at ICCV 2019: COCO Keypoint Challenge Track, 2019, p. 3.