Multi-camera, multi-person, and real-time fall detection using long short term memory

Taufeeque, Mohammad, Koita, Samad, Spicher, Nicolai, Deserno, Thomas

Mohammad Taufeeque, Samad Koita, Nicolai Spicher, Thomas M. Deserno, "Multi-camera, multi-person, and real-time fall detection using long short term memory," Proc. SPIE 11601, Medical Imaging 2021: Imaging Informatics for Healthcare, Research, and Applications, 1160109 (15 February 2021); doi: 10.1117/12.2580700

SPIE.

Event: SPIE Medical Imaging, 2021, Online Only

Multi-camera, multi-person, and real-time fall detection using long short term memory

Mohammad Taufeeque[a], Samad Koita[a], Nicolai Spicher[b], and Thomas M. Deserno[b]

[a]Indian Institute of Technology, Bombay, India
[b]Peter L. Reichertz Institute for Medical Informatics of TU Braunschweig and Hannover Medical School, Braunschweig, Germany

ABSTRACT

Approximately 37 million falls occur each year worldwide requiring medical attention. Victims are often helpless and not able to call for help, which is a risk for elderly persons living alone. To detect falls at home, several approaches have been proposed. Video cameras are used increasingly. Recently, high accuracy in real-time human pose estimation in videos has been achieved by novel machine learning techniques. In this work, we propose a multi-camera system for video-based fall detection. We augment human pose estimation (OpenPifPaf algorithm) by support for multi-camera and multi-person tracking and a long short-term memory (LSTM) neural network to predict two classes: "Fall" or "No Fall". From the poses, we extract five temporal and spatial features which are processed by the LSTM. For evaluation of identification and tracking with multiple cameras, we used videos recorded in a smart home (living lab) with two persons walking and interacting. For evaluation of fall detection, we used the UP-Fall Detection dataset and achieve an F1 score of 92.5%. We observed a tendency towards false positive classifications due to lack of activities in publicly available datasets that look similar to falls but stem from normal activity. Moreover, the lack of variation in the activities also results in a higher amount of false positives. This requires the acquisition of more balanced datasets in future work. In conclusion, real-time fall detection from multiple camera inputs and for multiple persons is feasible using a LSTM neural network combined with features obtained via human pose estimation. Source code is available at https://github.com/taufeeque9/HumanFallDetection.

Keywords: Video processing, multiple cameras, fall detection, human pose estimation, machine learning, long short-term memory, neural networks, deep learning

1. INTRODUCTION

According to the World Health Organization (WHO), 37 million falls occur each year and require medical attention.[1] Approximately 650.000 people die on fall-related injuries, making it the second leading cause of deaths.[1] If a fall occurs, victims are often helpless and unable to call for help which is a safety risk since more than 30% of elderly people in Europe live alone. Thereby, current research aims for developing methods for unobtrusive home monitoring.[2]

Several approaches have been proposed for automatic fall detection at home.[3,4] The majority of methods use wearable sensors (e.g., accelerator, gyroscope), ambient sensors (e.g., microphone, pressure sensor, radar sensor), and cameras (e.g., color, depth, thermal). However, these approaches are often invasive (wearable technology), inaccessible (depth cameras) or expensive. This has led to the rise in popularity of video-based systems as depicted in Fig. 1. Home cameras are cheap, provide contextual information (e.g., room dimensions, furniture) and have the advantage that they cannot be forgotten or discarded like ambient sensors or wearables.

Historically, fall detection algorithms can be classified into three main categories. The first one is based on decision-based approaches. Albawendi et al.[5] extract best-fit approximated ellipse around the human body and, in addition, projection histogram features. Hazelhoff et al.[6] use Principal Component Analysis to determine the direction of the main axis of the body and the ratio of the variances in x- and y-direction followed by Gaussian multi-frame classifier. The main drawback of these approaches is that they rely too much on predefined

Send correspondence to Nicolai Spicher, e-mail: nicolai.spicher@plri.de, phone: +49 531 391-2125

Medical Imaging 2021: Imaging Informatics for Healthcare, Research, and Applications
edited by Thomas M. Deserno, Brian J. Park, Proc. of SPIE Vol. 11601, 1160109
 · doi: 10.1117/12.2580700

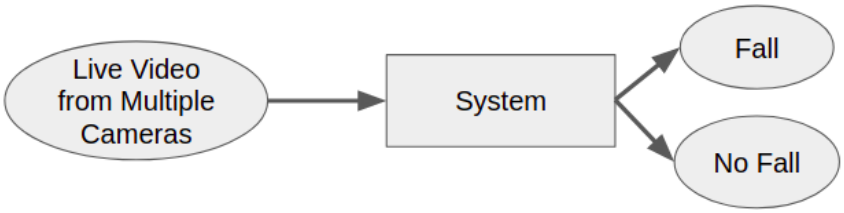


Figure 1: Principle of video-based fall detection systems

thresholds and cannot generalize well to unseen data. The second category of approaches are those which make use of handcrafted features extracted directly from raw images (e.g., position of the bounding box of a person, angle between the person's body and the vertical axis) along with traditional machine learning algorithms (e.g., Nearest Neighbor, Logistic Regression, Random Forest). For example, Miguel et al.[7] extract three features from optical flow images which are fed into a k-nearest neighbor algorithm. However, the features used by such approaches are unable to capture enough details about the posture of the person to make an informed decision. In addition, they do not handle time series data well. The third category represents end-to-end Deep Learning systems. The work proposed by Núñez-Marcos et al.[8] was one of the first papers implementing an end-to-end Deep Learning system with a stack of optical flow images being applied to a Deep Neural Network achieving high accuracy on different fall detection datasets. However, on the downside, is the risk of the model overfitting the training data and being sensitive to image parameters (e.g., brightness, contrast)

One of the biggest hurdles in the way of the success of video-based fall detection is the lack of adequate datasets. Videos of falls occurring in natural environments are very difficult to obtain. Therefore, instead of natural falls, datasets often contain videos of people enacting falls in laboratory environments. As a consequence trained models are often unable to generalize to real-world scenarios. Due to the controlled environment, recorded falls often lack variation in terms of the posture during the fall, as well as the angle from which the fall is captured. In addition, there are many activities that are similar to falls (e.g. kneeling, jumping) and therefore, are likely to be misclassified as datasets do not capture the full variety of activities.

Recently, machine learning techniques have advanced human pose estimation by enabling the detection of multiple human poses with high accuracy in real-time.[9] These techniques represent a human body by certain keypoints (e.g., ears, eyes, shoulders, elbows, knees) and have been applied already for human fall detection. Adhikari et al.[10] used the raw keypoints for training a LSTM neural network. Zhang et al.[11] reduced the pose information to an "inverted pendulum model" consisting of five keypoints (head, neck, buttocks, left/right knee) and analyzed temporal and spatial features connecting the keypoints in adjacent frames. Using this representation, they use the rotational energy and generalized force of motion as input for a logistic regression classifier. Both works report high precision and recall.

In this work, we aim to address certain limitations of prior work, namely over-fitting fall detection datasets, inability to distinguish between subjects in multiple cameras and high hardware requirements. Therefore, we propose a new methodology for video-based fall detection. As a proof of concept we developed an open-source algorithm* and, to the best of our knowledge, that work is the first open-source source algorithm to

- use pose information along with a state of the art neural network architecture to detect fall in videos,
- identify and track multiple subjects, along with coupling features from multiple uncalibrated cameras,
- have low hardware requirements so that it can also run without a graphics processing unit (GPU)

2. MATERIAL AND METHODS

In this section, we introduce our multi-camera, multi-person and real-time fall detection system followed by details on the dataset used for evaluation.

*The source code of the application is available at https://github.com/taufeeque9/HumanFallDetection

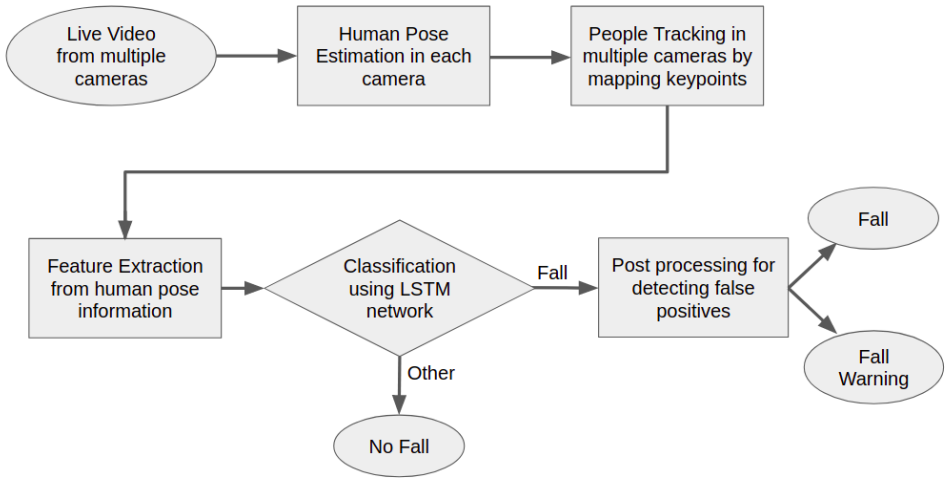


Figure 2: An overview of our proposed algorithm pipeline

2.1 Algorithm

Fig. 2 displays the algorithm pipeline with ellipsis denoting the input/output and rectangles indicating algorithmic steps which are presented in the following subchapters. The whole pipeline was implemented in Python3.

2.1.1 Human pose estimation

The inputs for this stage are the videos from each camera. We analyze each video as a sequence of images and process frames sequentially one after the other. Keypoints of human poses are extracted from each frame using the publicly-available, bottom-up human pose detector OpenPifPaf.[12] Its advantage lies within superior performance on low-resolution images as well lower hardware requirements compared to other state-of-the-art algorithms.[9] For each frame obtained from each camera, a list of keypoint sets is obtained with each keypoint set representing the position of joints of a single person.

2.1.2 Subject identification and tracking

Fig. 3 depicts the algorithm used for subject identification and tracking. In this work, we use two cameras only. However, the approach can be generalized to multiple cameras. At the end of the first stage, we obtain a list of keypoint sets for each camera, with each keypoint set corresponding to a different person. In the case that there are multiple people in the frames, if we are given any keypoint set from a camera, we do not know which keypoint set from the other cameras corresponds to the same person.

In this stage, we create a mapping between keypoint sets using a two-step procedure. At first, the list of keypoint sets is obtained with each set being mapped to the keypoint set of the same person in the previous frame. We use the spatial Euclidean distance between keypoints in consecutive frames as distance measure. Subsequently, we map the person-specific keypoint set from the first camera to the one with highest similarity in the second camera using a customized Gale-Shapley algorithm.[13] This matching approach correlates HSV (hue, saturation, value) color histograms.

2.1.3 Feature extraction

Once we have the mapped set of keypoints for each person in a video, five features for fall detection are extracted. These are based on temporal as well as spatial features and are independent of the pose detection algorithm. We derive three features from angle θ between the axis of the upper body and the vertical axis: (i) $\log(|\theta| + 1)$, (ii) θ'^2 where θ' denotes the first-order derivative of θ, and (iii) the lagrangian force on the person.[11] Additionally, we derive a bounding box from the head/knee keypoints and the outermost keypoints on the along the the axes of the image. From that bounding box we use the (iv) aspect ratio and (v) its derivative w.r.t. time as additional

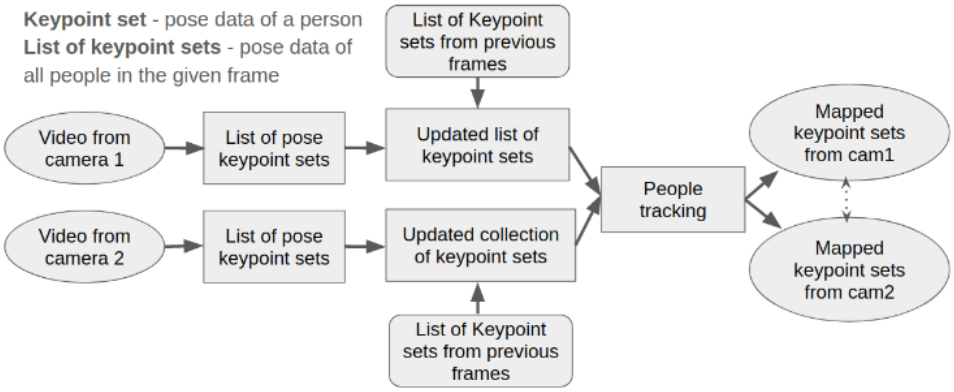


Figure 3: Concept of subject identification and tracking in multiple cameras (given here for two cameras)

features for classification. The set of these five features for each person forms the feature vector X_n where n represents a time step. Using this set of features we ensure to incorporate features that vary in different ways with respect to time. Features (iii) and (iv) correspond to the position of the subject (0^{th} order derivative w.r.t. time). Features (i) and (ii) correspond to the velocity/angular-velocity of the subject (1^{st} order w.r.t. to time), and feature (v) corresponds to acceleration of the subject (2^{nd} order derivative w.r.t. time).

2.1.4 LSTM classification and training

Once we have the features of each person from both cameras, we predict whether a person has fallen or not using an LSTM neural network architecture depicted in Fig. 4. At every time step n, the LSTM network uses the features X_n extracted from the current frame to predict whether the person has fallen. During deployment, we store the hidden states for each person that has been observed. When a new frame is observed, for each person, their feature vector X_n along with their hidden state (h_{n-1}, h'_{n-1}) is fed to the LSTM network that gives the updated hidden state (h_n, h'_n) along with the probability vector y_n that contains for each activity, the probability that the person is performing that activity. If the probability of the "falling" activity is highest in the prediction, a fall is detected for that frame.

The network is trained on data in the form (X, Y) where X is the set of features for a person from consecutive frames acquired over a period of two seconds from the dataset and Y is the activity label (Tbl. 1) of the last frame. Hence the model takes in the feature vector X_n of a person and learns to predict a probability distribution over 7 different classes, one being the "Fall" class and the rest are the "No Fall" classes represented separately. The hidden state of the model is initialized with the zero vector when the model takes in the features from the first frame of the video clip.

Implementation Details

We implemented the architecture of the network using the PyTorch library.[14] The LSTM architecture given in Fig. 4 consists of two LSTM layers each having 48 hidden nodes with hidden states of dimension 256. The output of the LSTM layer is then passed through a fully-connected layer that transforms the hidden representation from 256 to 7 nodes. This is then followed by the softmax layer that gives the probability of each activity. We used a weighted Cross Entropy loss function with each label's term in the loss function being weighted by its rarity. If a label has a relative frequency f, then the loss coming out of that label has a weight $1/f$. This ensures that the resultant network is trained in a balanced way, giving equal importance to the classes "Fall" and "No Fall" irrespective of the number of videos present in the dataset.

We speed up the training process by using mini-batches of size 64. All the models were trained using the Adam optimizer[15] provided by PyTorch with a learning rate of 0.0001. The model is regularized by using an L2 weight decay of 0.01 on all the weight parameters. The model is additionally regularized by applying dropout with a drop-probability of 0.1 on the LSTM layer. All hyperparameters were obtained by performing a grid-search over the hyperparameter space.

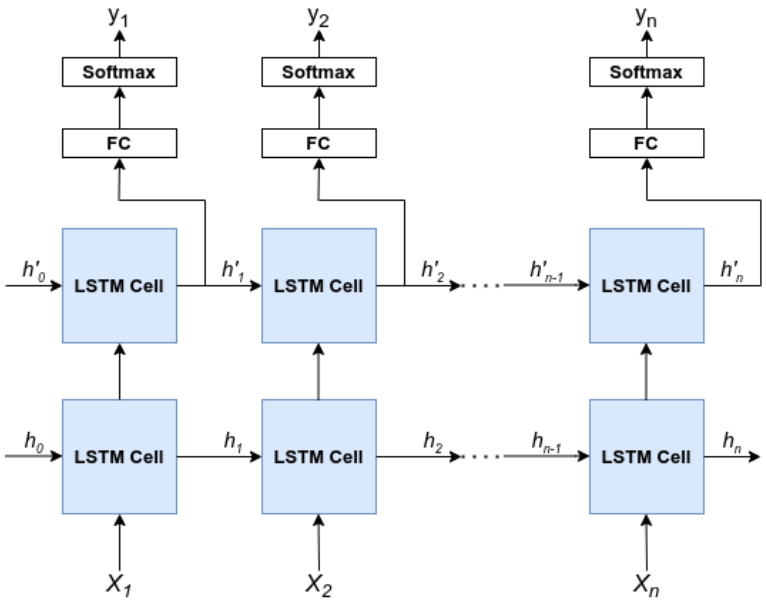


Figure 4: LSTM neural network architecture

2.1.5 Post Processing

During evaluation of the proposed LSTM network we observed a rather high amount of false positive fall detections due to actions being performed resulting in similar postures than falls (e.g. bending down abruptly). Hence, we implemented a post processing stage performing three sanity checks on frames that have been classified as falls by the LSTM Network. First, it checks whether the next K frames have also been classified as falls. K depends on the frame rate and we found a value corresponding to approximately 0.5s being adequate. Second, we check whether the angle between the person's body and the vertical $\theta > 45°$ in at least one of the cameras and, third, we compare if the person's height (distance from head to feet) is significantly smaller ($< 66\%$) than the moving exponential average value. If a frame was classified as "Fall" by the LSTM classifier and is rejected during post processing, the frame is re-labelled as "Fall Warning" (Fig. 2).

2.1.6 Dataset

We used the UP Fall Detection Dataset[16] to train and evaluate the proposed method. It consists of videos from two cameras of 17 subjects performing 11 different activities with three trials for each activity. Tbl. 1 summarizes all the activities along with the duration that each activity lasts for in the dataset. We grouped the activities 1-5 in one "Fall" class and kept the remaining classes of "No Fall" activities as they are.

Since this dataset does not contain multiple persons interacting with each other, we recorded videos in a smart home environment for evaluating subject tracking. We synchronized two perpendicular off-the-shelf cameras and record two persons walking and interacting in a typical living room shown in Fig. 5b.

Table 1: Activities in UP Fall Detection Dataset and our classification in "Fall" and no "No Fall" events

ID	Description	Dur. (s)
1	Falling forward using hands	10
2	Falling forward using knees	10
3	Falling backwards	10
4	Falling sidewards	10
5	Falling sitting in empty chair	10

(a) "Fall"

6	Walking	60
7	Standing	60
8	Sitting	60
9	Picking up an object	10
10	Jumping	30
11	Laying	60

(b) "No Fall"

Table 2: Results with the roman literals defining the features used for classification (chapter 2.1.3)

Methods/Metrics	**LR i-ii)**	**LSTM i-ii)**	**LSTM i-iii)**	**LSTM i-iv)**	**LSTM i-v)**
Accuracy	92.61	97.85	97.70	**98.28**	98.22
Precision	66.06	83.24	82.90	88.04	**89.76**
Recall	54.45	90.54	90.01	91.48	**95.62**
F1 Score	60.65	86.63	85.61	89.61	**92.56**

3. RESULTS

3.1 Metrics

Within the UP Fall Detection Dataset each frame of every video is assigned a label from the activities given in Tbl. 1. Therefore, we convert the multi-class labels to binary "Fall" vs. "No Fall" labels as shown in Tbl. 1 and define the following metrics for evaluation:

- **Accuracy**: The ratio of correctly classified frames to all classified frames.
- **Precision**: The ratio of correctly classified "Fall" frames to all frames classified as "Fall".
- **Recall**: The ratio of correctly classified "Fall" frames to all frames with ground-truth label of "Fall".
- **F1 Score**: The harmonic mean of precision and recall.

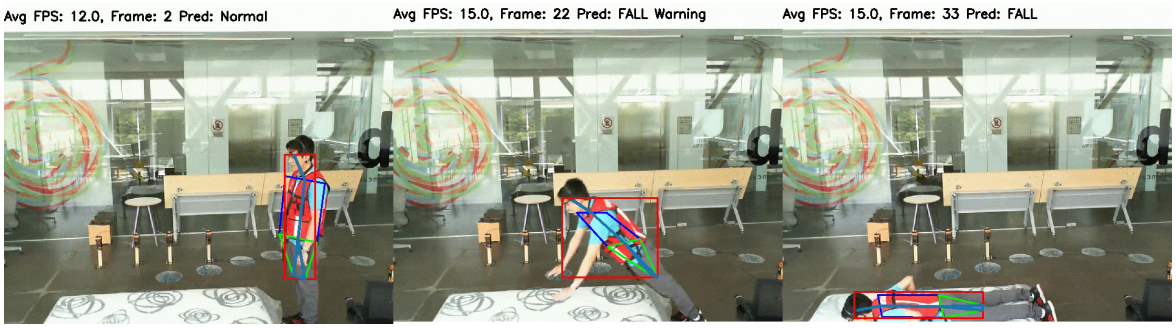
3.2 Evaluation

Tbl. 2 shows the results of using different models with different number of features on the UP Fall detection dataset. The results are obtained by performing 5-fold cross-validation. We obtain the best results using all five features with the LSTM network. As reference, we applied the logistic regression (LR) classifier proposed by Zhang et al.[11] As can be seen, the LSTM clearly improves all metrics. Additionally, it can be observed that increasing the number of features applied increases the overall performance. The best scores are obtained using all five features resulting in a F1 score of 92.56% with Fig. 5a showing example results of this configuration. We did not quantify the results of person identification and tracking in our video recordings but observed a high level of agreement with Fig. 5b showing an example.

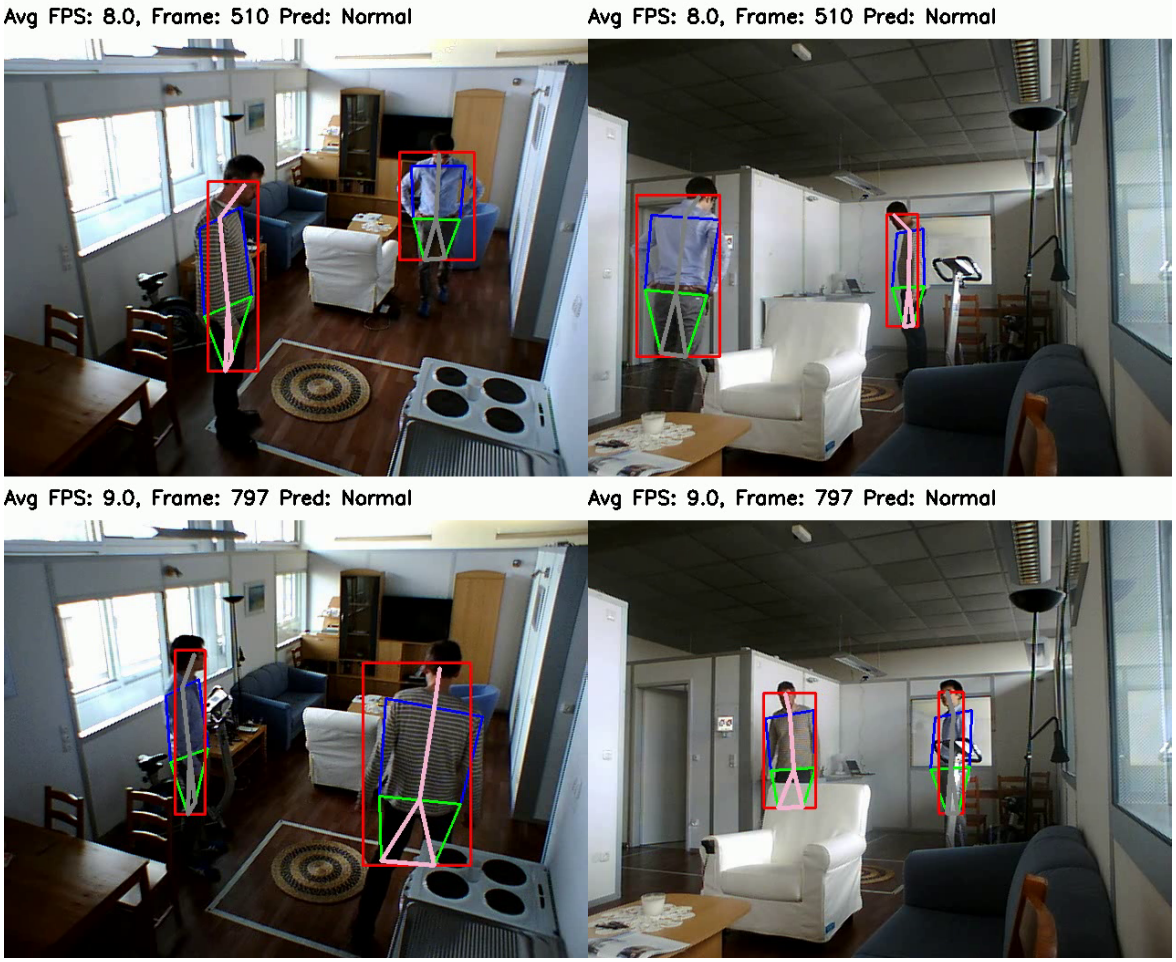
4. CONCLUSION

In this work, we proposed a new methodology for video-based fall detection using videos recorded with uncalibrated cameras. We extended recent works[10,11] by adding multi-camera capabilities, subject identification and tracking. In a proof of concept, we obtained good results on the UP Fall Detection Dataset. By using LSTM we expect to improve the performance of the LR classifier proposed in[11] but unfortunately we could not verify that as the authors did not respond to our requests in providing their dataset. Using the UP Fall Detection Dataset[16] instead, we observed the problem of the imbalanced datasets: 36% of videos (5/11 activities) show falls which is not realistic. Therefore, significantly larger datasets containing a realistic share of videos showing a fall are required. Due to different labelling and frames-per-seconds we could not train the LSTM on other datasets[17,18] which is an avenue for the future. As we used the efficient OpenPifPaf algorithm[12] for pose estimation, our proposed algorithm runs in real-time on off-the-shelf computers with rather low-processing power and does not require a GPU. At times, we observed that OpenPifPaf was unable to detect keypoints correctly once the person had fallen on the ground. Replacing the pose detection algorithm with a more accurate algorithm could increase performance (Tbl. 2) at the cost of higher processing power being required.

Our results raise several avenues for future work. To overcome the limitations of existing datasets recorded in laboratory environments, we will use the smart home (living lab) environment located at the Peter L. Reichertz Institute (Fig. 5b) to generate a large-scale, multi-camera video dataset representing realistic fall conditions and more realistic probabilities of fall occurrence. Furthermore, including information from the surrounding (e.g. furniture) acquired from the video via object detection could increase the value of the proposed method.



(a) Example frames of a video from the dataset processed by the proposed method. The second frame was classified by the LSTM as "Fall" and changed to "Fall Warning" by the post-processing stage (chapter 2.1.5).



(b) Example frames of a video recorded in the smart home (living lab) environment. Subject identification and tracking in two cameras is shown in two frames several hundred frames apart. Each person is shown with different colors (gray/pink).

Figure 5: Demonstration of video data processed by the proposed algorithm.

REFERENCES

[1] Tinetti ME, Leon CFMD, Doucette JT, Baker DI. Fear of falling and fall-related efficacy in relationship to functioning among community-living elders. Journal of Gerontology. 1994;49(3):M140–M147.

[2] Wang J, Spicher N, Warnecke JM, Haghi M, Schwartze J, Deserno TM. Unobtrusive health monitoring in private spaces: The smart home. Sensors. 2021 jan;21(3):864.

[3] Mubashir M, Shao L, Seed L. A survey on fall detection: Principles and approaches. Neurocomputing. 2013;100:144–152.

[4] Ren L, Peng Y. Research of fall detection and fall prevention technologies: A systematic review. IEEE Access. 2019;7:77702–77722.

[5] Albawendi S, Lotfi A, Powell H, Appiah K. Video based fall detection using features of motion, shape and histogram. In: Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference; 2018. p. 529–536.

[6] Hazelhoff L, Han J, de With PHN. Video-based fall detection in the home using principal component analysis. In: Advanced Concepts for Intelligent Vision Systems. Springer Berlin Heidelberg; 2008. p. 298–309.

[7] de Miguel K, Brunete A, Hernando M, Gambao E. Home camera-based fall detection system for the elderly. Sensors. 2017;17(12):2864.

[8] Núñez-Marcos A, Azkune G, Arganda-Carreras I. Vision-based fall detection with convolutional neural networks. Wireless Communications and Mobile Computing. 2017;2017:1–16.

[9] Gong W, Zhang X, Gonzàlez J, Sobral A, Bouwmans T, Tu C, et al. Human pose estimation from monocular images: A comprehensive survey. Sensors. 2016;16(12):1966.

[10] Adhikari K, Bouchachia H, Nait-Charif H. Long short-term memory networks based fall detection using unified pose estimation. In: Osten W, Nikolaev DP, editors. Twelfth International Conference on Machine Vision (ICMV 2019). SPIE; 2020. p. 114330H.

[11] Zhang J, Wu C, Wang Y. Human fall detection based on body posture spatio-temporal evolution. Sensors. 2020;20(3):946.

[12] Kreiss S, Bertoni L, Alahi A. PifPaf: Composite fields for human pose estimation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2019. p. 11978–11986.

[13] Gale D, Shapley LS. College admissions and the stability of marriage. The American Mathematical Monthly. 1962;69(1):9.

[14] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32; 2019. p. 8024–8035.

[15] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: 3rd International Conference for Learning Representations, San Diego, 2015; 2015. Available from: http://arxiv.org/abs/1412.6980.

[16] Martínez-Villaseñor L, Ponce H, Brieva J, Moya-Albor E, Núñez-Martínez J, Peñafort-Asturiano C. UP-fall detection dataset: A multimodal approach. Sensors. 2019;19(9):1988.

[17] Auvinet E, Rougier Cea. Multiple cameras fall dataset. Université de Montréal; 2010. 1350.

[18] Kwolek B, Kepski M. Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer Methods and Programs in Biomedicine. 2014;117(3):489–501.