Human-skeleton based Fall-Detection Method using LSTM for Manufacturing Industries

Sungil Jeong*, Sungjoo Kang**, Ingeol Chun* **
*Department of Information and Communication Technology
University of Science and Technology, KOREA
**Dependable Cyber-Physical Systems Research Group
Electronics and Telecommunications Research Institute, KOREA
sij049@ust.ac.kr, sjkang@etri.re.kr, igchun@etri.re.kr

Abstract

According to the statistics of the Korea Occupational Safety & Health Agency, the incidences of falls in the manufacturing industry are increasing. In this paper, we introduce a fall-detection method based on skeleton data obtained from a 2D RGB CCTV Camera installed on the manufacturing floor. We proposed feature-extraction methods to improve of fall-detection accuracy and the construction of a fall-detection system using LSTM. Experiments were conducted through public datasets (URFD and SDUFall) to find feature-extraction methods that can achieve high classification accuracy. The experimental results showed that the proposed method is more effective in detecting falls than raw skeleton data which are not processed anything.

Keywords: Fall-detection, Machine learning, Smart Factory, LSTM

1. Introduction

In manufacturing industries (ship building, aerospace, automobile factory, etc.), there are many safety-based accidents such as falls, collisions, fires, and explosions, that could cause personal injury. To avoid sudden accidents, most factories install video surveillance systems to increase safety on the manufacturing floor. However, existing video surveillance systems, which rely on the judgement of a small number of administrators, reveal many vulnerabilities due to poor management. To solve this problem, studies have been conducted to apply action recognition to the video surveillance system that recognizes human activity through computer. A fall involves a part of the human body touching or falling on the ground. Some fall cases may result in a simple bruise but may cause serious injury such as slipped disk and fracture. Unlike a general fall accident, if a person experiences a fall due to a heart attack, the initial response is of great importance. Therefore, falls must be accurately and rapidly detected. The existing fall-detection system is divided into sensor- and vision-based systems. The sensor-based system [1,2] has the ability to build low-cost systems by using inexpensive sensors; however, these systems must be attached to the human body. A manufacturing-floor environment requires vision-based systems that do not need to be attached to the body.

2. Related Works

The vision-based fall-detection system detects features in videos and uses them to detect falls. Some methods detect falls by characterizing human shape or silhouette in the videos by subtracting the background [3] or extracting and using human-skeleton data from videos as a feature [4,5]. Recently, studies have focused on fall detection using a method of recognizing human activity based on skeleton data. Skeleton data is obtained either from a 3D depth camera, such as Kinect developed by Microsoft, or from CNN-based techniques such as OpenPose [6] in 2D RGB videos. The extracted skeleton data is spatio-temporal data that changes with time. Recurrent neural network (RNN) is a powerful method for dealing with classification of time series data. However, as learning through RNN utilizes a long time, its results in the vanishing- and exploding-gradient problems. To solve these problems, the use of long short-term memory (LSTM) [7] was introduced. LSTM is not affected by these problems. Generally, CCTV on the manufacturing floor utilizes 2D RGB cameras; thus, in this paper, we extracted the skeleton data from videos using OpenPose. The extracted skeleton data were classified through LSTM, and fall is detected based on the inference results.

3. Manufacturing-Floor Fall-Detection System

3.1 Architecture

The architecture of the manufacturing-floor fall-detection system (MFFDS) constructed in this study is shown in Figure 1. The architecture consists of six steps. STEP 1 includes the data-acquisition process, which collects 2D RGB video data from surveillance cameras on the manufacturing floor.

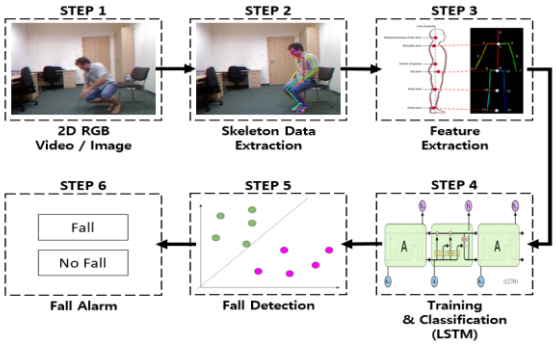

Figure 1: MFFDS Architecture

In STEP 2, the raw skeleton data are extracted from the video through OpenPose, as shown in Figure 2. In STEP 3, the extracted skeleton data are processed through a proposed feature-extraction method before analyzing them through LSTM.

No	Joint
0	Nose
1	Neck
2	R Shoulder
3	R Elbow
4	R Wrist
5	L Shoulder
6	L Elbow
7	L Wrist
8	Mid Hip
9	R Hip
10	R Knee
11	R Ankle
12	L Hip
13	L Knee
14	L Ankle

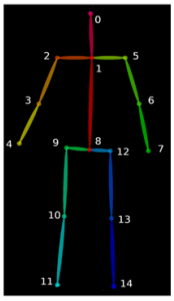

Figure 2: Skeleton-Data Information

In STEP 4, the data processed according to the feature extraction method enter the learned LSTM, which outputs the inference result corresponding to the data. STEP 5 involves the process of determining whether the extracted data represent a fall through the inference results obtained from LSTM. In STEP 6, the alarm is sounded based on the result obtained in STEP 5.

3.2 Feature Extraction

In this study, two feature-extraction methods were used. The first is the human center line coordinate (HCLC) feature. The relationship between the upper and body parts is considered an important factor in detecting falls because the upper and lower body parts cannot be artificially adjusted when they fall. We expect a similar data pattern from most people. The HCLC represents the coordinate corresponding to the center line of the body when the body is viewed from the direction of gravity. HCLC consists of four *X*–*Y* coordinates (i.e., eight data). The coordinates show the center points of the shoulders (2,5), hips (9,12), knees (10,13) and ankles (11,14).

$$\begin{bmatrix} C_{m\_shoulder} & C_{m\_hip} \\ C_{m\_knee} & C_{m\_ankle} \end{bmatrix} = \begin{bmatrix} \frac{C_{l\_shoulder} + C_{r\_shoulder}}{2} & \frac{C_{l\_hip} + C_{r\_hip}}{2} \\ \frac{C_{l\_knee} + C_{r\_knee}}{2} & \frac{C_{l\_ankle} + C_{r\_ankle}}{2} \end{bmatrix}$$

The second feature is the speed of HCLC (SHCLC), which obtains the distance between the coordinates of the next and current frames based on the HCLC by using the Euclidean distance formula. Then, by dividing the obtained distance by the time between frames, four SHCLC data can be obtained.

$$SHCLC = \frac{\sqrt{(C_{next\_x} - C_{cur\_x})^2 + (C_{next\_y} - C_{cur\_y})^2}}{\Delta t}$$

(C_{next} : *next frame HCLC*, C_{cur}: *current frame HCLC*)

4. Experiments and Results

4.1 Dataset

Two datasets were used in this experiment: UR Fall Detection (URFD) Dataset [8] and SDUFall Dataset [9]. Both datasets were recorded with Microsoft Kinect camera, and the features of each dataset are shown in Table 1.

Table 1: Dataset Information

	URFD	SDUFall
Configuration	RGB Video (640×480, 30 fps), Depth Video, Accelerometer Data	RGB Video (640×480, 30 fps), Depth Video, 3D Skeleton Data
Number of falls	30 (30 falls + 40 ADLs)	200 (20 actors, 10 times fall)
Fall direction	forward, backward, left, right	forward, left, right
Scenario	Falls while sitting down, Falls while walking	Falls while walking
Camera viewpoints	One	One
Brightness change	No	Yes

4.2 Subsequence Data-Creating Method

For the learning of the LSTM model, each subsequence datum was constructed using the sliding window method for each video in the dataset. In addition, each subsequence datum obtained through the sliding window method has one label. Figure 3 illustrates how subsequence data are generated. The label of the subsequence data was determined by the entity that occupies the largest percentage of each subsequence datum.

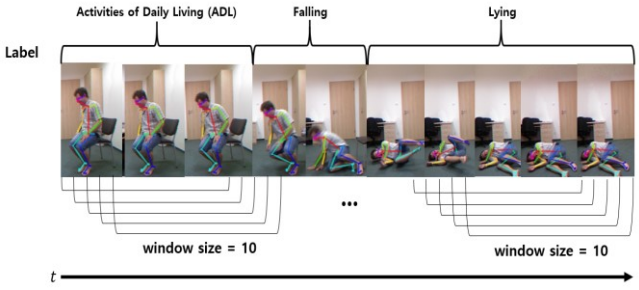

Figure 3: Process of Creating Subsequence Data

For example, in a subsequence of 10 frames, 7 frames are labeled ADL, while the remaining have a falling label; the subsequence will have an ADL label. The subsequence data generated as such are divided into training and test data; they are used for learning.

4.3 Video-based Fall-Detection Method

Figure 4 shows the method of detecting falls in a video through inferred results learned by LSTM. The subsequence data are fed into the learned LSTM, which outputs the inference results for each subsequence datum.

```
Detect falls through inference results
Input: predictions(from LSTM, Inference Results, list), window_size
Output: f_index (start frame of fall judgment)
1.   GET predictions
2.   SET win_size ← window_size
3.   FOR i = 0 to len(predictions) DO
4.     IF i + win_size >= len(predictions) THEN
5.       end_index ← len(predictions)
6.     ELSE
7.       end_index ← i + win_size
8.     ENDIF
9.     IF predictions[ i : end_index ].sum() == 1
10.                              and predictions[ i ] == 1 THEN
11.      IF i != end_index -1 THEN
12.        PRINT "Fall Detection"
13.        SET f_index ← i + 1
14.      ENDIF
15.    ENDIF
16.  ENDFOR
```

Figure 4: Detect Falls through Inference Results

The inferred results are entered into the input of Figure 4. In this study, it is judged that the fall occurred if the result of the inference is “falling” and then “lying” on the floor for a fixed time. In contrast, if “falling” or “ADL” occurred within a fixed time frame, it is not judged a fall. The inference results were analyzed using the sliding window method. (In the video with 30fps) The window size in Figure 4 changes according to the duration of “lying.” If the time of 3 seconds for “lying” is judged as a fall, the window size is 91 (“falling” is 1 and “lying” is 90). We assigned different weights to the inferred results in the window (ADL: 2, falling: 1, lying: 0). As a result, if the sum of weights in the window is 1, it is judged as fall.

4.4 Experimental Configuration

We implemented the LSTM through TensorFlow and used the GeForce GTX 1080 GPU. The experiment results were obtained from the training of 500 epochs and learning rate of 0.0001. The LSTM used in the classification has 2 stacked layers and 256 hidden-layer features. As a result of experiment with increasing 1~4 stacks, classification accuracy of 96.94%, 98.89%, 98.64% and 98.64% was obtained. (Increase by 40s for each additional stack) Considering the execution time and classification accuracy, the 2 stacks are most appropriate. The number of hidden-layer features was also obtained through experiments. (32, 64, 128, 256, and 512 sizes) The accuracy of classification was 87.18%, 91.21%, 97.12%, 98.89% and 98.70% respectively. The execution time increased by about 10s, 10s, 50s, and 150s, respectively, as the size increased. When the number of features increased to 256, the accuracy increased significantly. However, when the sized was increased to 512, the accuracy was similar, and the time was much longer than before, so we chose 256. The Dataset used to determine the two hyper-parameters is URFD, and the feature extraction method is “RAW + SHCLC”. We used the Adam [10] optimizer, which can be reliably optimized. For the multi-classification, Softmax function was used. In raw skeleton data extracted using OpenPose, some missing parts were filled by linear interpolation. And the extracted skeleton joint coordinates were normalized by referring to resolution of each dataset.

4.5 Experimental Results

Experiments were conducted using the two public datasets introduced in Section 4.1. The URFD dataset was used to test which feature extraction methods have high classification accuracy. Table 2 shows the classification accuracy for the URFD dataset. The length of the subsequence data is 10 (frames), and the number of subsequence data per class is as follows: (URFD Total: 10749, lying: 2123, falling: 1642, ADL: 6984). Because Deep Learning is a black box method, classification accuracy according to feature-extraction method is analyzed through experimental results. The “RAW” that used all raw skeleton data without processing showed an accuracy of 95.3%. The results of “RAW + HCLC” and “RAW” show that “HCLC” is a significant feature of class classification. “SHCLC” is a speed related feature, and “HCLC” is a coordinate related feature. The result of “RAW + SHCLC + HCLC” shows that the correlation between speed and coordinate is less. So the accuracy was less than when each was used separately. In the case of “SHCLC”, it is the most changeable feature when falling in relation to speed. Therefore, “RAW + SHCLC” feature showed high accuracy in “falling” class classification.

Table 2: Classification Result using URFD Dataset

Method	Data_size /frame	Classification Accuracy (%)			
		Avg	Lying	Falling	ADL
RAW + SHCLC	34	**98.83**	98.59	98.18	99.71
RAW + HCLC	38	97.7	98.12	95.15	99.85
RAW + SHCLC + HCLC	42	97.72	99.06	94.54	99.57
RAW	30	95.3	99.06	87.27	99.57

We expected that feature extraction methods with high average classification accuracy would show high fall-detection accuracy in Videos. To verify this, we experimented with the SDUFall dataset. The length of the subsequence data is 10 (frames), and the number of subsequence data per class is as follows: (SDUFall Total: 15279, lying: 5886, falling: 2835, ADL: 6558). The fall-detection method is the same as that described in Section 4.3. From the results of "Learned dataset: URFD", we found that the average classification accuracy and fall-detection accuracy are not proportional. It was found that the average accuracy of "lying" and "falling" was related to fall-detection accuracy. “L&F AVG” means the average of the classification accuracy of "lying" and "falling". We found that the fall-detection accuracy was proportional to “L&F AVG”.

Table 3: Classification and Fall Detection Accuracy using URFD and SDUFall Dataset

Method	Data_size /frame	Learned Dataset	Classification Accuracy (%)					Fall Detection Accuracy (%)
			AVG	L&F AVG	Lying	Falling	ADL	
RAW + SHCLC	34	URFD	65.91	**82.36**	97.80	66.91	33.04	**97.38**
		URFD + SDUFall	**99.15**	**98.78**	99.93	97.63	99.90	100
RAW + HCLC	38	URFD	67.81	78.65	99.52	57.77	46.15	94.76
		URFD + SDUFall	98.85	98.49	99.83	97.14	99.58	100
RAW + SHCLC + HCLC	42	URFD	72.56	83.15	95.93	70.37	51.38	95.28
		URFD + SDUFall	98.91	98.76	99.84	97.68	99.22	100
RAW	30	URFD	61.51	75.82	99.38	52.27	32.90	94.76
		URFD + SDUFall	98.64	98.11	99.86	96.36	99.71	100

We could get the following conclusions from this experiment. All three classes are important in detecting falls, but the classification accuracy of "falling" and "lying" is especially important. Although "RAW + SHCLC" did not have the highest average classification accuracy, it showed relatively high accuracy in average accuracy of "lying" and "falling" than other methods. This led to high fall-detection accuracy, and we concluded that "RAW + SHCLC" is the most suitable method for detecting falls. SDUFall dataset has similar composition and features to the URFD dataset. However, the classification accuracy is low when considering LSTM, which only learns the URFD dataset. For a person, the two datasets are similar but from a data point-of-view, a difference is observed. The result of learning both datasets (Learned dataset: URFD + SDUFall) showed that the classification and fall-detection accuracies were increased in every method; all methods showed better performance. The experimental results showed that the more the data learned, the higher is the performance. Figure 5 shows the executed result of the implemented system.

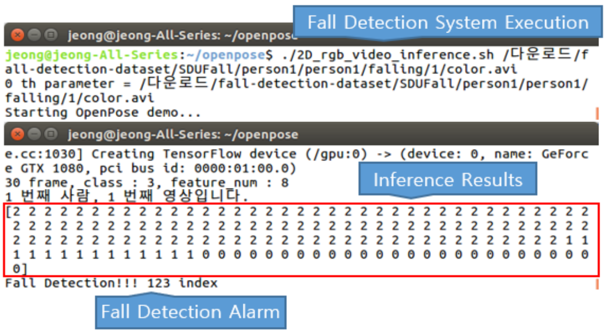

Figure 5: MFFDS execution result

5. Conclusion

This paper presented a 2D RGB video-based fall-detection system. The LSTM was adopted to detect falls from continuous human activity. We proposed feature-extraction methods by using human-skeleton data. We experimented with several combinations of the proposed feature-extraction methods, and determined that the classification accuracy of “falling” and “lying” is important for fall-detection accuracy. Experimental results have shown that “RAW + SHCLC” is the most suitable feature-extraction method for detecting falls. In the future, we will apply our MFFDS at an actual manufacturing floor and conduct research to obtain better fall-detection accuracy.

Acknowledgment

This work was supported by National IT Industry Promotion Agency(NIPA) grant funded by the Korea government(MSIT)(No.S0249-19-1062, Development of Software Development Kit(SDK) and Core Libraries for Shipbuilding & Marine Engineering Industry)

References

[1] AK Bourke, JV O’brien, and GM Lyons, “Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm”, *Gait & Posture*, 2007.

[2] WC Cheng, and DM Jhan, “Triaxial accelerometer-based fall detection method using a self-constructing cascade-AdaBoost-SVM classifier”, *IEEE Journal of Biomedical and Health Informatics*, 2013.

[3] A. Abobakr, M. Hossny, and S. Nahavandi, “A Skeleton-Free Fall Detection system From Depth Images Using Random Decision Forest”, *IEEE Systems Journal*, vol. 12, Sept. 2018.

[4] P. T. Hai, and H. H. Kha, "An efficient star skeleton extraction for human action recognition using hidden markov models", *IEEE Sixth International Conference on Communications and Electronics*, 2016.

[5] Wen-Nung Lie, Anh Tu Le, and Guan-Han Lin, “Human Fall-down Event Detection Based on 2D Skeletons and Deep Learning Approach”, *International Workshop on Advanced Image Technology*, 2018.

[6] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”, *Computer Vision and Pattern Recognition*, 2017.

[7] Sepp Hochreiter, and J. Schmidhber, “Long short-term memory”, *Neural Computation*, 1997.

[8] Bogdan Kwolek, and Michal Kepski, “Human fall detection on embedded platform using depth maps and wireless accelerometer”, *Computer Methods and Programs in Biomedicine*, 2014.

[9] X. Ma, H.Wang, B. Xue, M. Zhou, B. Ji and Y. Li, “Depth-Based Human Fall Detection via Shape Features and Improved Extreme Learning Machine”, *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 6, pp.1915-1922, Nov. 2014.

[10] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization”, *ICLR*, 2015.