Your WiFi Knows You Fall: A Channel Data-driven Device-free Fall Sensing System

Mengmeng Huang*, Jun Liu*, Yu Gu*,Yifan Zhang*, Fuji Ren†, Xiaoyan Wang§ and Jie Li‡
* School of Computer and Information, Hefei University of Technology, China
† Dept. of Information Science & Intelligent Systems, University of Tokushima, Japan
‡ Dept. of Computer Science, University of Tsukuba, Japan
§ Dept. of Media and Telecommunications Engineering, Ibaraki University, Japan, Japan

***Abstract*—Falls are the second leading cause of injury deaths worldwide, inducing over 0.6 million accidental deaths per year. Among various prevention strategies, fall-related research has been prioritized. However, conventional fall detection solutions rely on computer vision or wearable sensors embody several inherent limitations such as scalability, coverage, and privacy issues. To this end, we present FallSense, a transparent and real-time fall sensing system driven by wireless channel data. FallSense is built on a Dynamic Template Matching (DTM) algorithm, which can start with a light training set and keep updating on usage. FallSense has been realized on commodity WiFi devices and evaluated in real environments. Experimental results show that FallSense outperforms another state-of-the-art approach WiFall in terms of detection precision, false alarm rate and complexity.**

I. Introduction

Fall is the second leading reason of unintentional deaths worldwide. According to the World Health Organization (WHO), each year over $646\,000$ deaths are directly related to accidental events related to fall. Naturally, this issue raises high attention in both industry and academe, where timely and reliable fall detection solutions in various environments are desired. This issue remains an academic hot spot and draws much attention from worldwide researchers for decades [1].

Current mainstream solutions on fall detection can be generally divided into two classes based on the devices used, i.e., devices, i.e., vision-based and sensor-based. In general, the former explores images or video clips [2], [3] captured by vision devices to directly detect a fall event , while the latter mainly replies on wearable or ambient sensors to identify fall by sensing its certain features like acoustic noise or floor vibration [4]–[6]. Also, as people are becoming more and more smartphone-dependent in modern societies, it is natural for researchers to explore these sensor-rich and always-carry-on devices for fall detection [7].

Previous research on vision-based and sensor-based fall detection has achieved solid progress. However, recent studies have revealed several their inherent limitations such as scalability (specialized or costly hardware), coverage (illumination/ line-of-sight constraints), and privacy issues (high level of obtrusiveness) [8]–[12]. Therefore, researchers are striving for revolutionary paradigms over the conventional approaches.

In this paper, we present FallSense, a light-weight fall detection system that automatically and accurately monitors the status of a person and recognizes a fall leverage only WiFi infrastructures. It explores wireless signals for a contactless and noninvasive way of detection falls that robustly works in a wide range of environments including sites, users, illumination conditions. Compared to conventional fall detection solutions, FallSense possesses several significant advantages. Firstly, it is a cost-effective system built on the off-the-shelf WiFi infrastructure. Secondly, it is transparent to users and raises no privacy or security concerns, especially suitable for children and elders. Lastly, it provides better coverage while relying little on the environmental conditions.

FallSense leverages a light-weight Dynamic Template Matching (DTM) algorithm for detecting falls in a real-time manner. The rivals like WiFall [10], RT-Fall [12] rely on the large-volume training data and the sophisticated classifiers such as Hidden Markov Model (HMM) and Support Vector Machine (SVM), leading to two major problems: dependence on the static and optimized training set, and high computational complexity. Therefore, we design this DTM algorithm to start light with a small and arbitrary training set that is updated online. In this way, FallSense not only optimizes its performance on usage, but also significantly reduces the overall complexity.

We prototype FallSense with commodity WiFi devices and evaluate its performance in real environment, where it demonstrates strong stability by achieving better performance than the state-of-the-art WiFall system [9], [10].

The rest of this paper is organised as follows: in the next section we set up a prototype and present some preliminary experimental results, followed by the system design of Sleepy in Section III. Then, we evaluate its performance in Section IV. Finally, we conclude our work and outline some future directions in Section V.

II. Preliminaries

Understanding how signals are affected by falls and other effects In everyday activities, we built prototypes on previous WiFi-based motion recognition systems [13], [14] to get some preliminary results.

A. An Overview of Channel Response

In general, recent Wi-Fi standards leverage either RSS representing the received power level, or CSI indicating signal

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

attenuation, as channel response.

RSS characterizes the total received power of all paths,

$$RSS = 10\log_2(\|H\|^2), \tag{1}$$

where $H=\sum_{k=1}^{N}\|H_k\|e^{j\theta_k}$ with $\|H_k\|$ and θ_k being the amplitude and phase of the k-th multipath, respectively. Equation (1) indicates that RSS is coarse-grained so that it is inherently incapable of capturing the multi-path effect. Therefore, recently CSI on the PHY layer which can better capture multi-path channel features emerges as a more effective alternative.

Specifically, in Orthogonal Frequency Division Multiplexing system (OFDM), $H(f,t)$ represents the complex value channel frequency response (CFR) in the format of CSI, which characterizes the channel performance with the amplitude and phase for sub-carrier frequency f measured at time t.

$$H(f,t) = \sum_{k=1}^{N} h_k(f,t)e^{-j\theta_k(f,t)}, \tag{2}$$

where h_k is the amplitude that characterizes the attenuation, $e^{\theta_k(f,t)}$ is the phase shift on the k-th path caused by a propagation delay.

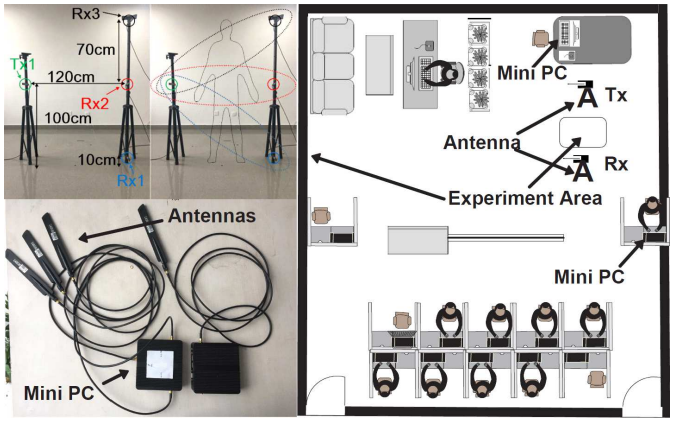

Fig. 1: Prototype and layout of laboratory.

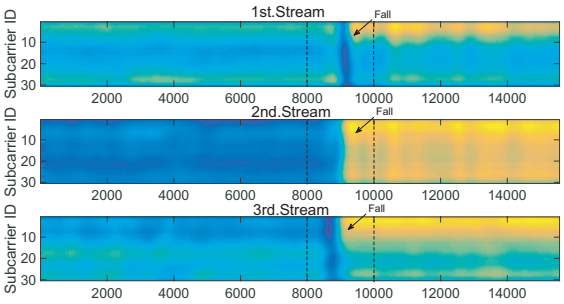

Fig. 2: The impact of fall on channel data.

B. Preliminary Experiments

[Prototype]. Fig. 1 shows our prototype, which consists of two MiniPCs equipped with Intel network interface to form a controller (NIC) 5300 (running at 5 GHz). One miniPC is equipped with an external antenna as a transmitter, while the other is connected to three antennas as a receiver. Atennas are fixed on the tripods. The sampling rate is 1000 Hz.

[Participant]. Considering the potential danger of this experiment, a young man participated in the preliminary experiment.

[Environment]. The experiments were carried out in a $7 \times 10m^2$ office room (Fig. 1). It includes office furniture, including chairs, sofas, and computer desks. Other students are present in the same room during the experiment.

[Motions]. As in WiFall [9], we choose four similar falls (sitting, standing, lying down and squatting), which may be confused with falls. Empty serves as a benchmark.

[Experiment]. The experiment setting is shown in Fig. 1. The three elliptic regions present the sensing regions of each pair of antennas. Participants are asked to perform motions in the region.

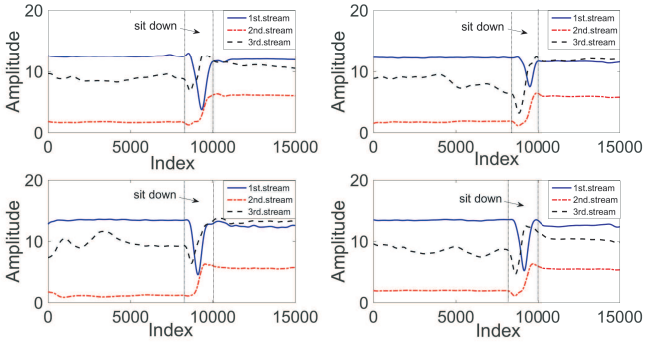

Fig. 3: A pattern exists for the activity "sitting down".

We ask the participant to perform every motion ten times, where two key observations have been derived:

1) Human motion indeed affects channel response and such effect follows certain patterns: Fig.2 visualizes the CSI amplitude profiles of one selected case, all three data streams vividly record the impact of falls on channel data. As shown in Fig.3, it is clear that for the same person there exists a pattern for a given activity.

2) Such effect on channel data is activity-dependent: In Fig.4, the amplitudes of the three data streams are obtained by averaging 30 subcarriers. You can see that different activities indeed have different patterns. For instance, the signal distortion caused by "lie down" significantly differs with "fall" on channel data.

In summary, we confirm that human motion indeed affects the channel data and there exist patterns for different activities through preliminary real-world experiments. These observations inspire us to develop a template-based fall detection system, i.e., FallSense. The next section details its design.

III. System Design

In this section, we present the system design of FallSense.

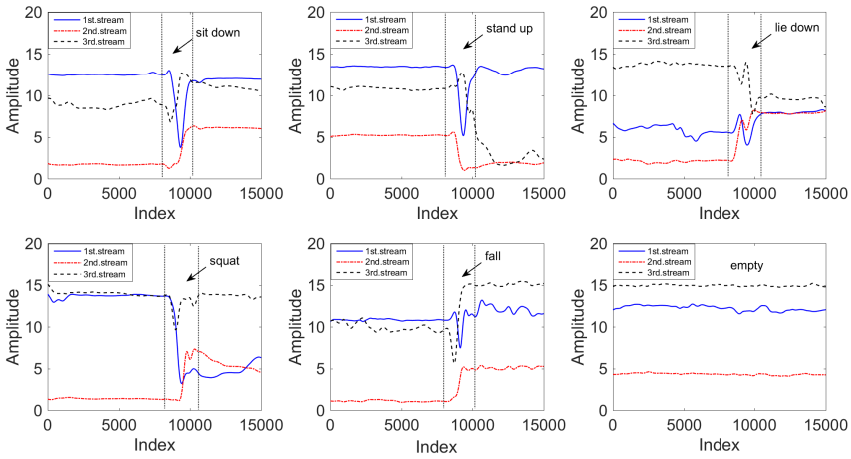

Fig. 4: The impact on channel data is activity-dependent.

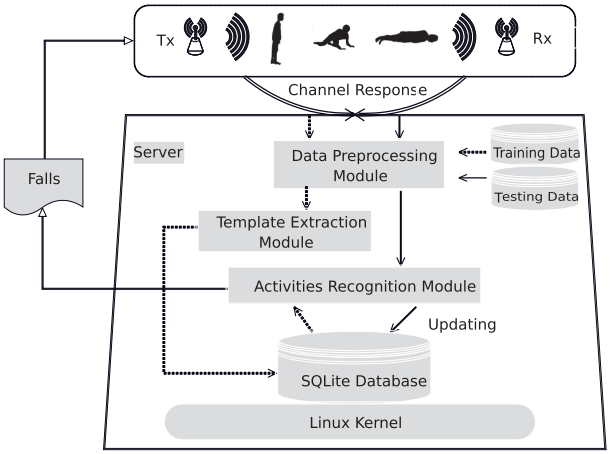

Fig. 5: System architecture of FallSense.

Algorithm 1: Training Date Flow.

Input: $\mathbf{CSI_{train}}$; C_{thr}(Cluster thredhold);
Output: SQL(SQLite Database);
begin
  $\mathbf{STemplate}$=Cluster($\mathbf{CSI_{train}}$,C_{thr});
  D_Maintenance=D_Calculate($\mathbf{STemplate}$,$\mathbf{CSI_{train}}$);
  return SQL.$\mathbf{STemplate}$;
  return SQL.D_Maintenance;
  return SQL.$\mathbf{CSI_{train}}$;
  end
Function *Template Extraction*($\mathbf{CSI_{train}}$,C_{thr})
  Collect the CSI_{1-30} (30 subcarriers) from $\mathbf{CSI_{train}}$;
  $\overline{CSI_{train}} = \sum_1^{30} CSI_{train(i)}$;
  while $C_{thr} \leq \widehat{D}$ **do**
    Divide and extracting processed CSI data into several core templates;
    Calculate the Eulic distance $\widehat{d_k}$ between every $\overline{CSI_{train}}$ and it's nearest core templates;
    $\widehat{D}$=$\sum_1^n \widehat{d_k}$;
  return $\mathbf{STemplate}$;
Function *D_Calculate*($\mathbf{STemplate}$,$\overline{CSI_{train}}$)
  Calculate the Eulic distance $\widehat{d_k}$ between every fragment and it's nearest $\mathbf{STemplate}$;
  Record the farthest $\widehat{d_k}$ and the fragment's label into D_Maintenance;

A. System Overview

Fig. 5 presents an overview of the system design. FallSense is built on top of the DTM algorithm and has two streams of data, training and testing. As shown in Alg. 1, the former denoises through the data preprocessing module, and the latter extracts standard templates of different activities from the training set through the template extraction module. Both the template and the training dataset are stored in a SQLite database installed on a Linux server. As shown in Alg. 2, he latter also first passes the data preprocessing module, then identifies the activity and updates the template through the activity recognition module.

In the next parts, we will introduce each module in details.

B. Data Preprocessing Module

The raw channel data may contain anomalous samples caused by background noise or hardware failure, so it should be filtered first. In the preprocessing module, we select the Butterworth filter. As we sample CSI values at a rate of F_s = 1000 samples/s, we set the cut-off frequency ω_c of the Butterworth filter at $\frac{2\pi \cdot f}{F_s} = 0.0942$ rad/s.

The reason we choose the Butterworth filter lies in that the frequency of variations caused due to human motions lie at the low end of the spectrum while the frequency of the

Algorithm 2: Testing Data FLow

Input: $\mathbf{CSI_{test}}$; SQL(SQLite Database);
Output: Classify Result; SQL(SQLite Database);

begin
  while *TRUE* **do**
    Classify Result=Classify(SQLite Database,$\mathbf{CSI_{test}}$);
    SQL=Update(SQLite Database,$\mathbf{CSI_{test}}$,Classify Result);
    return Classify Result;
    return SQL;
    end
  end
Function *Acitivity Recognition(SQL,*$\mathbf{CSI_{test}}$*)*
  Collect the CSI_{1-30} (30 subcarriers) from $\mathbf{CSI_{test}}$;
  $\overline{CSI_{test}} = \sum_1^{30} CSI_{test(i)}$;
  Calculate Eulic distance between the $\overline{CSI_{test}}$ and every SQL.$\mathbf{STemplate}$, finds the closest $\mathbf{STemplate}$, outputs the corresponding activity;
  return $Classify\ Result$;
Function *Update(SQL,*$\mathbf{CSI_{test}}$*,Classify Result)*
  Calculate the Eulic distance $\widehat{d}$ between $\overline{CSI_{test}}$ and the SQL.$\mathbf{STemplate}$ form Classify Result;
  Read farthest Eulic distance $\mathbf{d_k}$ from SQL.D_Maintenance in corresponding $\mathbf{STemplate}$;
  if $\widehat{d} \leq d_k$ **then**
    replace the SQL.$\mathbf{CSI_{train}}$ with $\mathbf{CSI_{test}}$;
    Update SQL.$\mathbf{STemplate}$;
    Update SQL.D_Maintenance ;
    return SQL;
  else
    return ;

noise lies at the high end of the spectrum. To remove noise in such a situation, Butterworth low-pass filter is a natural choice which does not significantly distort the phase information in the signal and has a maximally flat amplitude response in the passband and thus does not distort the motion signal much.

C. Template Extraction Module

After preprocessing, the training data is fed into the template extraction module, and the standard template libraries of different actions are extracted from the training data of the module. Although the fingerprints of CSI signals of different actions are quite different, the challenge is that different people perform the same action differently, because different body shapes and personal habits lead to the diversity of physical expression. Obviously, the best way is to maintain a separate template for each individual action, but the complexity of the algorithm will also increase. Therefore, we set a parameter C_{thr} to achieve a tradeoff between complexity and performance.

Assuming there are K participants, the kth participant performs the activity i ($i \in [1, M]$) n_i times. We average those n_i entries into a template. So we have K templates of each activity. If all K templates are used for classification, then performance should be the best. Complexity is also the highest. Therefore, we further divide K templates into specific groups. We average templates in each group as a core template and then calculate the Euclidean distance d_k between the kth templates and the core template as follows,

$$d_k = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}, \quad (3)$$

where $\vec{X} = \{x_1, \cdots, x_n\}$ denote the kth templates, and $\vec{Y} = \{y_1, \cdots, y_n\}$ denote the core template, and d_k is the Euclidean distance between them. The n on the right side of the equation is not n_i, which is the n-dimensional space in the Euclidean distance formula.

Then we sum all d_k to get D, which is used to evaluate the diversity of physical expressions of different people in the same movement:

$$D = \sum_{k=1}^{K} d_k \ , \quad (4)$$

We enumerate all possible grouping methods until D is less than the tradeoff value C_{thr}. In general, the smaller C_{thr} is, the larger the size of the template library, resulting in higher accuracy and higher complexity.

D. Activity Recognition Module

This module is used for motion recognition. Calculates the Euclidean distance between CSI_{test} and the template library, find the closest standard template, and then output the corresponding action type. If the test set data stream is better than one of its data streams in the standard corpus, the test set data stream will replace the worst template to update the standard corpus.

E. Complexity Analysis

We present the complexity analysis for the proposed algorithm.

Proposition 1. The computational complexity of FallSense is $O(N)$ for training and $O(t \cdot N)$ for testing, where N and t represent the size of the training set and the standard corpus, respectively.

Proof. For the template extraction module, we process N samples directly. So, its complexity is $O(N)$. For the activity recognition module, we compare N samples to t standard templates, so the computational complexity is $O(t \cdot N)$. Therefore, the overall complexity is $O(t \cdot N)$.

IV. Evaluation

We realize Fallsense on the prototype system and evaluate its perform in the real environment.

A. Evaluation Setup

The prototype, environment and experimental settings are the same as the preliminaries. We have recruited 8 participants (2 females), whose age ranks from 21 to 26. As in WiFall [9], we select four fall-like activities (i.e., sit down, stand up, lie down, and squat) that could be potentially confused with fall. As in WiFall [9], we use two metrics, i.e., *precision (P)* and *false negative rate (FNR)*, to characterize the overall system performance as follows,

$$\mathbf{P} = \frac{\textit{\# of True positives}}{\textit{\# of True postives + \# of False postives}}. \quad (5)$$

$$\mathbf{FNR} = \frac{\textit{\# of False negatives}}{\textit{\# of True postives +\# of False negatives}}. \quad (6)$$

For **P**, we further define P_5 and P_2 to denote the detection precision for all five activities as well as fall/fall-like activities, respectively.

$$\mathbf{P}_5 = \frac{\textit{\# of correct prediction of 5 actvities}}{\textit{\# of testing entries}}. \quad (7)$$

$$\mathbf{P}_2 = \frac{\textit{\# of correct prediction of fall or not fall}}{\textit{\# of testing entries}}. \quad (8)$$

Each participant was asked to perform 10 non-falls, 20 falls. Therefore, the data set is composed of $8\times4\times10+8\times20 = 480$ data entries. Each entry is a 3 (streams) $\times$ 30 (subcarriers) $\times$ 3000 (packages, 3 seconds $\times$ 1000 Hz) matrix.

B. The impact of parameters on Performance

TABLE I

#Standard templates for each activity	Updating=0	Updating=1
1	60%	60.42%
8	94.17%	94.58%

In this part, we study the impact of different system parameters (listed in Tab. I) on the performance.

TABLE II: Updating=0 & #Standard templates=1

	Sit down	Stand up	Lie down	Squat	Fall
Sit down	16(40%)	3(7.5%)	3(7.5%)	4(10%)	14(35%)
Stand up	3(7.5%)	31(77.5%)	3(7.5%)	2(5%)	1(2.5%)
Lie down	3(7.5%)	5(12.5%)	26(65%)	0(0%)	6(15%)
Squat	2(5%)	0(0%)	5(12.5%)	31(77.5%)	2(5%)
Fall	0(0%)	15(18.75%)	8(10%)	17(21.25%)	40(50%)

[Updating=0 & #Standard templates=1].

Under this setting, we don't update the corpus and keep only one template for each activity. The corresponding confusion matrix is shown in Table II. As we mentioned before, there exist certain variations on channel data for the same activity performed by different participants. Therefore, FallSense only achieves 60% $\mathbf{P}_5$, and 73.75% $\mathbf{P}_2$.

[Updating=1 & #Standard templates=1].

Under this setting, we keep updating the corpus but only keep one template for each activity. The corresponding confusion matrix is shown in Table III. FallSense achieves 60.42% $\mathbf{P}_5$, and 76.67% $\mathbf{P}_2$. There exists improvements over the first setting, caused by corpus updating.

TABLE III: Updating=1 & #Standard templates=1

	Sit down	Stand up	Lie down	Squat	Fall
Sit down	16(40%)	4(10%)	3(7.5%)	6(15%)	11(27.5%)
Stand up	1(2.5%)	32(80%)	3(7.5%)	3(7.5%)	1(2.5%)
Lie down	0(0%)	7(7.5%)	18(45%)	3(7.5%)	12(30%)
Squat	2(5%)	4(10%)	4(10%)	28(70%)	2(5%)
Fall	2(2.5%)	11(13.75%)	6(7.5%)	10(12.5%)	51(63.75%)

TABLE IV: Updating=0 & #Standard templates=8

	Sit down	Stand up	Lie down	Squat	Fall
Sit down	39(97.5%)	1(2.5%)	0(0%)	0(0%)	0(0%)
Stand up	1(2.5%)	37(92.5%)	1(2.5%)	1(2.5%)	0(0%)
Lie down	0(0%)	1(2.5%)	37(92.5%)	0(0%)	2(5%)
Squat	0(0%)	0(0%)	3(7.5%)	37(92.5%)	0(0%)
l Fall	0(0%)	1(1.25%)	3(3.75%)	0(0%)	76(95%)

[Updating=0 & #Standard templates=8].

Under this setting, we don't update the corpus but keep 8 templates (8 participants) for each activity. The corresponding confusion matrix is shown in Table IV. FallSense achieves 94.17% $\mathbf{P}_5$, and 97.5% $\mathbf{P}_2$. Compared with previous two settings, the performance of FallSense has been improved significantly. This observation implies that the bigger the standard corpus is, the better the performance will be.

TABLE V: Updating=1 & #Standard templates=8

	Sit down	Stand up	Lie down	Squat	Fall
Sit down	39(97.5%)	1(2.5%)	0(0%)	0(0%)	0(0%)
Stand up	1(2.5%)	37(92.5%)	1(2.5%)	1(2.5%)	0(0%)
Lie down	0(0%)	1(2.5%)	37(92.5%)	0(0%)	2(5%)
Squat	0(0%)	0(0%)	2(5%)	38(95%)	0(0%)
Fall	0(0%)	1(1.25%)	3(3.75%)	0(0%)	76(95%)

[Updating=1 & #Standard templates=8].

Under this setting, we keep updating the corpus and use 8 templates (8 participants) for each activity. The corresponding confusion matrix is shown in Table V. FallSense achieves 94.58% $\mathbf{P}_5$, and 97.5% $\mathbf{P}_2$. Compared with the previous setting, there exists certain performance improvement. Like our previous analysis, it is indeed the best setting for FallSense system from the aspect of system performance.

C. Performance Comparison

To further verify our proposed system, we implement WiFalls [WiFall-2014 uisng SVM and WiFall-2017 using Random Forest (RF)] [9], [10] and compared their performance with FallSense under the same experimental conditions. As shown in Fig.6, FallSense achieves 95% P and 2.44% FNR, which are significantly better than both WiFalls. As for why Fallsense's FNR is greater than Wifall-2014, Fallsense needs to set parameters to achieve a trade-off between complexity and performance. Interestingly, although the accuracy of the 2017 WiFall has improved compared to its 2014 version (92.31% vs. 91.86%), its false positive rate (FNR) has also risen sharply. (from 1.39% to 7.1%).

Furthermore, Table VI compares FallSense and WiFalls in term of complexity, where m, n and t represent number

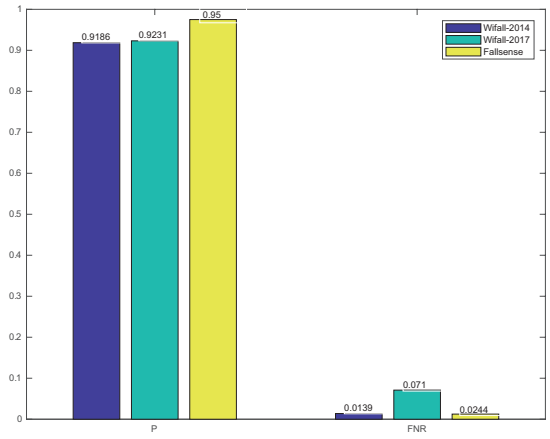

Fig. 6: Comparison of Wifall and FallSense.

TABLE VI: Comparison of Wifall and FallSense

	Complex(train)	Complex(test)
Wifall-2014	$O(m^2N^2)$	$O(m^2N)$
Wifall-2017	$O(m^2NlogN)$	$O(mNlogN)$
FallSense	$\mathbf{O(N)}$	$\mathbf{O(tN)}$

of features , number of samples, and number of templates, respectively. It is straightforward to see that FallSense is superior than WiFalls in all three metrics.

Summary. Our real-world empirical research not only demonstrates the feasibility of fall detecting via template matching method, but also shows that FallSense can work in a real-time manner. In short, Fallsense is a low-overhead but highly efficient fall monitoring system, so it has potential application value in the real-life environments.

V. Conclusion and Future Work

This paper proposes a WiFi-based fall monitoring system, i.e., FallSense. Compared with its competitors in computer vision and wearable sensors, this algorithm has three significant advantages. First, it is a low-cost system built on top of a ready-made WiFi infrastructure; second, it is transparent to users and does not cause privacy or security issues; in the end, it provides better coverage, and rarely Depends on environmental conditions. Experimental results show that the performance of this algorithm is better than WiFall. It has great advantages in detection accuracy, false alarm rate and complexity.

There are several points of Fallsense that require further attention. For example, the existing template-based fall monitoring algorithm is coarse-grained because it treats a fall as a single motion rather than a composite motion. In addition, system evaluation requires the expansion of system parameters and evaluation scenarios.

Acknowledgments

This research is funded by the NSFC Grant No. 61432004 and 61772169. The authors gratefully acknowledges the support of K.C.Wong Education Foundation.

References

[1] Z. Zhang, C. Conly, and V. Athitsos, "A survey on vision-based fall detection," in *ACM International Conference on Pervasive Technologies Related To Assistive Environments*, Corfu, Greece, July 2015, p. 46.

[2] C. Rita, P. Andrea, and V. Roberto, "A multi-camera vision system for fall detection and alarm generation," *Expert Systems*, vol. 24, no. 5, pp. 334–345, 2010.

[3] A. Nunez-Marcos, G. Azkune, and I. Arganda-Carreras, "Vision-based fall detection with convolutional neural networks," *Wireless Communications & Mobile Computing*, vol. 2017, no. 1, pp. 1–16, 2017.

[4] D. Litvak, Y. Zigel, and I. Gannot, "Fall detection of elderly through floor vibrations and sound," in *IEEE Engineering in Medicine and Biology Society*, Vancouver, Canada, Oct 2008, pp. 4632–4635.

[5] M. Alwan, P. J. Rajendran, S. Kell, and D. Mack, "A smart and passive floor-vibration based fall detector for elderly," in *Information and Communication Technologies, 2006. Ictta '06.*, Damascus, Syria, April 2008, pp. 1003–1007.

[6] Y. Zigel, D. Litvak, and I. Gannot, "A method for automatic fall detection of elderly people using floor vibrations and sound–proof of concept on human mimicking doll falls," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 12, pp. 2858–67, 2009.

[7] S. Zhang, P. McCullagh, H. Zheng, and C. Nugent, "Situation awareness inferred from posture transition and location: Derived from smartphone and smart home sensors," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 6, pp. 814–821, Dec 2017.

[8] S. Kianoush, S. Savazzi, F. Vicentini, V. Rampa, and M. Giussani, "Leveraging rf signals for human sensing: Fall detection and localization in human-machine shared workspaces," in *Proc. of 2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, Cambridge, UK, July 2015, pp. 1456–1462.

[9] C. Han, K. Wu, Y. Wang, and L. Ni, "Wifall: Device-free fall detection by wireless networks," in *Proc. of IEEE INFOCOM 2014*, Toronto, Canada, April 2014, pp. 271–279.

[10] Y. Wang, K. Wu, and L. M. Ni, "Wifall: Device-free fall detection by wireless networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 581–594, Feb 2017.

[11] D. Zhang, H. Wang, Y. Wang, and J. Ma, "Anti-fall: A non-intrusive and real-time fall detector leveraging csi from commodity wifi devices," in *Inclusive Smart Cities and e-Health*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 9102, pp. 181–193.

[12] H. Wang, D. Zhang, Y. Wang, J. Ma, Y. Wang, and S. Li, "Rt-fall: A real-time and contactless fall detection system with commodity wifi devices," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 511–526, Feb 2017.

[13] Y. Gu, F. Ren, and J. Li, "Paws: Passive human activity recognition based on wifi ambient signals," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 796–805, Oct 2016.

[14] Y. Gu, J. Zhan, Y. Ji, J. Li, F. Ren, and S. Gao, "Mosense: An rf-based motion detection system via off-the-shelf wifi devices," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2326–2341, Dec 2017.