Fine Grained Group Gesture Detection Using Wearable Devices

Yongjian Zhao
Department of Computer Science
Colorado School of Mines
Golden, CO, USA
yzhao2@mines.edu

Stephen New
Department of Computer Science
Colorado School of Mines
Golden, CO, USA
snew@mines.edu

Kanchana Thilakarathna
School of Information Technologies
The University of Sydney
Sydney, NSW, Australia
kanchana.thilakarathna@sydney.edu.au

Xiaodong Zhang
School of Electrical Engineering and Telecommunication
University of New South Wales
Sydney, NSW, Australia
xiaodong.zhang@student.unsw.edu.au

Qi Han
Department of Computer Science
Colorado School of Mines
Golden, CO, USA
qhan@mines.edu

***Abstract*—People may perform synchronized activities in a group setting. It is helpful to provide notifications to users and also the group leader whether people are in sync. This work aims to provide this support via analyzing motion data collected from wearable devices. We collected experimental data from smart watches worn by people, applied signal processing algorithms in both time and frequency domains for identification of the fine-grained group gesture status. We further developed a prototype system consisting of a smart watch, a smartphone, and a server. Our simulation results and actual system implementation demonstrate the feasibility of our approaches.**

***Index Terms*—Group gesture detection, motion sensor, time domain analysis, frequency domain analysis**

I. Introduction

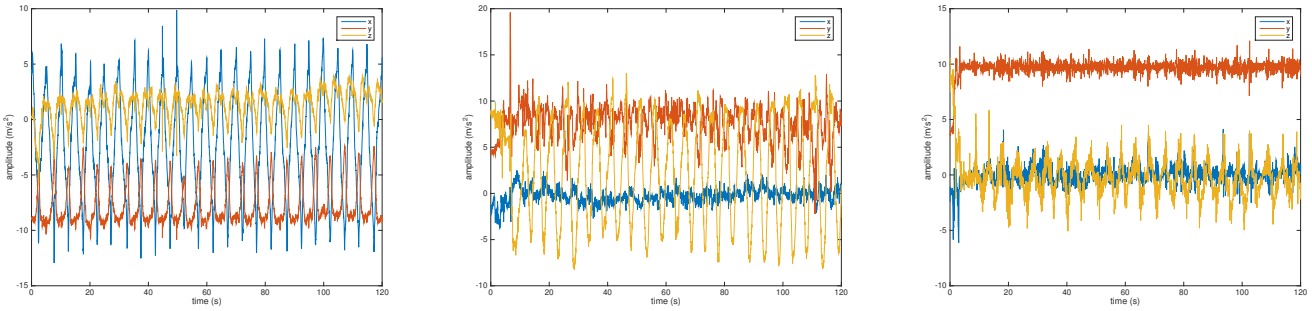
Wearable computing devices show promise in taking human life experience way beyond expectations as a result of the number of attractive services provided by them. Today wearable devices are extremely popular as personal health and fitness tracking devices. However, the true potentials of wearable devices are yet to discover. These smart wearable devices such as smart watches, smart glasses, fitness bands, smart shoes are equipped with a rich set of sensors that can continuously monitor a wide variety of attributes of the human body, the physical surroundings, and the online behaviors of the users that are not available through any other means. In the future, it is highly likely that individuals will wear more than one of these devices in their day-to-day life.

These wearables can be effectively used for the identification of user status in groups that perform the same activity at the same location. This fine grained identification of user status can be helpful in many applications such as emergency response, disaster recovery, and sport activities. For instance, we can help guide people towards emergency exits during a fire evacuation or identifying the group of supporters of a particular team in a sports game. This fine-grained identification of user status can also be used for resource optimization. For instance, the fitness tracking apps on two devices can collaborate with each other to save the energy consumption of the devices if the devices identify that the user is running together with a partner.

The primary aim of this paper is to explore the potential of identifying the user groups with similar activity at the same location. Instead of using traditional methodology of video analysis, we use sensor data extracted from smart watches. We first develop an experimental measurement framework to collect and exchange wearable sensor data. We then apply a signal processing algorithm to accurately identify the group behavior and validate the algorithm with the collected data. We finally develop a prototype smart-watch/smartphone app to demonstrate the practical feasibility of the solution.

II. Related work

Since people tend to form groups in real world activities, a lot of work has been done to recognize



(a) hand waving left and right (b) hand waving up and down (c) hand pushing and pulling

Fig. 1: raw signals of different gestures

groups. One way to detect groups is to first recognize each user's activity and then analyze their cooperative or collaborative relationship [2], this can be classified as the group activity detection problem. Most activity recognition approaches are not generic and they often lead to solutions that are tied to the specific scenarios. Therefore, an algorithm was designed which embeds feature construction into the machine learning process [1]. However, in some cases we may just want to find out whether people belong to the same group rather than identifying what specific activity they are performing. This leads to the problem called group affiliation detection.

Groups can be identified based on interactions, proximity, mobility, or activity. Most existing work relies on mobility for group detection, in which the individuals who have the similar trajectories are considered in the same group. For example, GruMon [5] determines a group of individuals in a specific location who are traveling together in crowded urban environments. Spatial and WiFi information [4] are utilized for indoor group detection. Location is the key factor to determine whether users belong to the same group when relying on mobility.

Another kind of group based on similarities of activities /gestures performed by users, a divergence-based affiliation detection (DBAD) approach [3] is proposed which provides a framework to identify group affiliation based on a sensing modality. Since the modality has to be manually selected, a two-stage process [6] is designed where sensing modality selection given a high level activity is performed, followed by multimodal clustering to build a general platform to identify sub-groups. In our work, rather than identifying specific activities or determining group affiliation, we determine whether group members are performing same activities in synchronicity using wearable sensors.

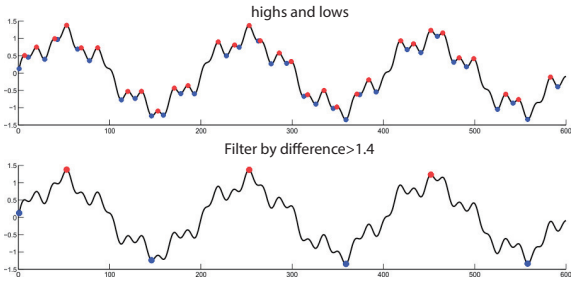


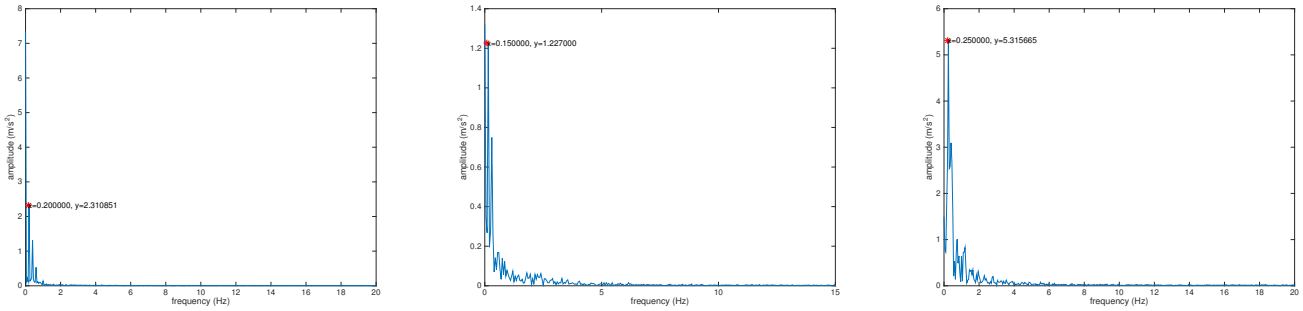
Fig. 2: Peak detection

III. Group Gesture Detection

We assume each user wears a wearable device in the same position of the body. Accelerometer data is collected from each user and sent to a central server.

There are various gestures and each can be represented by an index (e.g, waving hands from left to right, moving hands up and down). We define a set of gestures I as $I = \{I_1, I_2, ..., I_n\}$. Each gesture is associated with a starting time stamp S and a completion time stamp C. i.e., it takes a user $T = C - S$ (s/round) to complete a gesture, implying the frequency $f = 1/T$ Hz. The gestures may be repetitive, i.e., users can repeat a particular gesture k times. Therefore, the total completion time for the user would be kT. For each user in a group, we define his activities as a series of gestures $I_1, I_2, ..., I_i$ with starting time stamp $S_1, S_2, ..., S_i$ and completion time stamp $C_1, C_2, ..., C_i$.

In order to identify different status of users, the server first selects a user as reference, each user is then compared with the reference and the status is identified and reported to both the individual user and server within each time window.



(a) reference (b) slow (c) fast

Fig. 3: User status detection using frequency domain features

1) Select the group "reference" user - We define a user as the reference if the majority of group users (i.e., over 50%) are doing the same gesture with respect to the the selected reference.
2) Identify each user's status at different time windows. A user's gesture status within a group can fall into the following categories:
   - Different gestures - the users are performing different gestures.
   - Same gestures, which can be divided into several synchronization status:
     - Synchronized - the gesture period of the user under test (UUT) is the same as the reference, and the difference in starting times tamp is less than 0.5s, that is, $S_{UUT} - S_{ref} \leqslant 0.5$s.
     - Lagging - the gesture period of the user under test (UUT) is the same as the reference, and the difference in starting times tamp is greater than 0.5s, that is, $S_{UUT} - S_{ref}$ ¿ 0.5s.
     - Fast - the gesture period is smaller than that of the reference, that is, $T_{UUT} \leqslant T_{ref}$.
     - Slow - the gesture period is larger than that of the reference, that is, $T_{UUT} > T_{ref}$.

We next describe the details of each of the two steps.

A. Selection of Reference User

The reference user can be either pre-assigned or selected using the following steps. First, we choose the gesture performed by most users (at least half of the group) as the reference gesture, this will exclude the users who are doing different gestures. Second, we choose the most common gesture completion time among all users. Last, we determine the proper time lag. Here we assume all the time lags form a normal distribution, then the user with the lag closest to the expected value should be the reference.

B. Identification of User Status

We divide the total completion time for each user into several fixed length time windows. For example, if the total completion time is 2 minutes, and the time window size is selected to be 5 seconds, then for each user, we have 24 time windows.

1) Different gestures: In a group setting, people may perform different gestures and we would like to distinguish them. Fig. 1 shows the raw acceleration signal of three different gestures. We can observe that the acceleration data varies among three axes, namely, x, y, z: data in x-axis varies the most on the gesture which the user waves the hand left and right, whereas data in z axis varies the most on the gesture which the user waves the hand up and down.

Since our primary goal is to determine if the users in the group are doing the same gesture with the reference rather than to specifically identify which gesture each individual user performs, we simply extract the amplitude of the signals at each axis and. We calculate the average amplitude of peak values from each axis for each user. if the difference between the values of the two users exceeds the predefined threshold θ, the two users are performing two different gestures.

2) Same gestures: If two users are performing the same gesture, we then determine whether the user is synchronized with the reference. We apply two approaches to identify each user's status: time domain analysis and frequency domain analysis.

a) Time Domain Analysis: We apply extrema detection on the raw signal, here the extrema refers to peak (local maximal) and trough (local minimal). For two signals,

we detect the peak and the trough in each time window. As peak and trough appear in pairs, we pair them up and filter out the pairs that have the smallest difference. The difference is computed by subtracting the trough value from the peak value. This algorithm is described in Algorithm 1.

Algorithm 1 :Extrema detection

Require: time series data $s_1, s_2, s_3, ..., s_n$
Ensure: peaks and troughs with certain difference values
1: Sort the time series acceleration data in ascending order. return array S'
2: **for** $i \in [1, n]$ **do**
3: **if** S'(i) is a local minimal **then**
4: Create a span, a range with left and right bound that covers the local minimal
5: **end if**
6: **if** S'(i) is neither a local minimal nor a local maximal **then**
7: Expand the span to the left or right
8: **end if**
9: **if** S'(i) is a local maximal **then**
10: Generate the paired extrema. Calculate the difference
11: **end if**
12: **end for**
13: Filter out the extrema pairs with small differences

To generate the results shown in Fig. 2, we use half of the difference of the highest and lowest as the threshold. If we detect more peaks and troughs within one time window than the reference, the user is considered being 'fast'. In contrast, if we detect fewer peaks and troughs than the reference user, the user is considered being 'slow'. We then calculate the time difference between the starting point of the window and the nearest peak/trough. If the difference is larger than 0.5s, the user is considered lagging behind; otherwise, the user is synchronized.

b.) Frequency Domain Analysis. We first apply Fast Fourier Transformation (FFT) to the original time series data. After getting the frequency of the signal, we choose the frequency with the highest amplitude value as the movement frequency. For instance, in Fig. 3, we use the frequency f as 0.15Hz, 0.2Hz and 0.25Hz for the status of slow, synchronized and fast respectively. We then compare these frequencies with the reference. If the measured frequency is larger than the reference, we consider this user as moving fast (Fig. 3 (c)), otherwise, the user is moving slower than the reference (Fig. 3(b)). If the user is moving with the same frequency as the reference, we calculate the phase difference between the user and the reference, if it is bigger than 0.7s, the user is lagging behind; otherwise, the user is synchronized.

IV. Performance Evaluation

We first use simulation to test our approaches and then implement the system on actual hardware platforms.

A. Simulation Setup and Results

To evaluate system scalability, we test the performance under different number of users. Due to the limited devices available, we only compared the result of one smart watch with the ground truth. We consider four scenarios: different gestures, fast or slow, lagging, and synchronized. Each scenario has a percentage associated: $\alpha, \beta, \gamma, \delta$. To match our assumption of selecting a reference user, δ should be larger than 0.5. Here we set it to be 0.6. $\alpha = 0$, $\beta = 0.2$, $\gamma = 0.2$. We can generate the lagging and synchronized cases using a normal distribution. As 0.5s will be our threshold to distinguish between the status of synchronized and lagging, we expect around 75% of the time difference to fall into (-0.5s, 0.5s) region and the expectation of the time difference with the reference should be 0. We then generate 40 random users status of lagged using normal distribution N (0,0.5). Finally we add $50 \times 0.1 = 5$ users as slow cases and 5 users as fast cases.

To keep scenarios simple, we make the following assumptions: (1) each user performs the same gesture: waving hand from left to right. This generates raw signals in Sinusoid-like waveform. (2) all users move periodically. (3) minimum window size should be larger than the completion time of the reference gesture. We choose one reference user and generate other data by shifting the signals. For reference case, a user waves hand from right to left for 5 seconds. We test five basic cases:

1: synchronized
2: waving hand from left to right with 0.5s lagging
3: waving hand from left to right with 1s lagging
4: waving hand from left to right with the period of 4s
5: waving hand from left to right with the period of 6s

We use precision and recall to measure the overall performance of gesture detection. Precision is the ratio of correctly identified windows to the total number of identified time windows, recall is the ratio of correctly identified windows to the total number of time windows that should have been identified. In addition, we are interested in (a) whether we can correctly find the reference user; and (b) precision in different time window.

Fig. 4 shows the result using frequency domain analysis averaged over 10 simulations. We observe that

FFT achieves overall good performance under difference scenarios except for the fast scenario. This is because faster movement results in small frequency values and may not be distinguished when the frequency of the reference user is small enough.

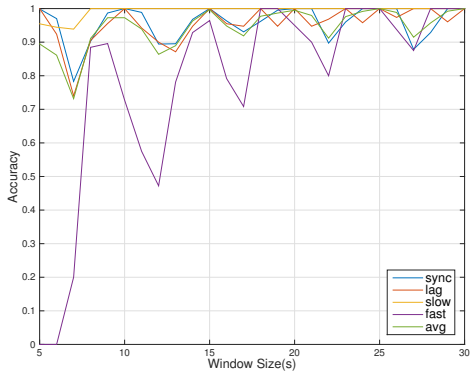


Fig. 4: Frequency domain analysis (50 users, $\alpha = 0, \beta = 0.2, \gamma = 0.2$)

We also test the performance on five different datasets with similar basic cases mentioned above.

- left: waving hand from left to right
- push: push hand forward and pull hand backward
- up: waving hand up to down
- walk: attach a smart watch to the ankle and walk
- twist: twist hand clockwise and anticlockwise alternatively

Fig. 5 shows that the walk dataset performs the worst since we attach a smart watch on the ankles and the data variation on each axis is not as significant as the data collected when the watch is worn on the wrist.

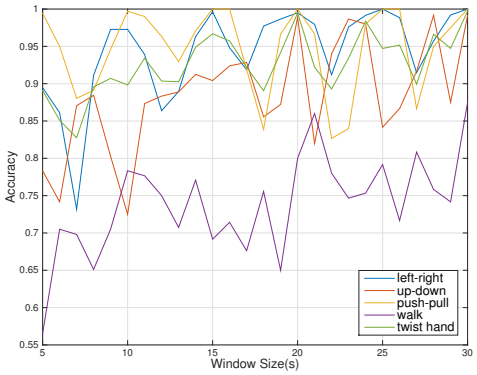


Fig. 5: Frequency domain analysis on different datasets

Table I and II shows the precision and recall results on two datasets: left and push. The confusion matrix

Status	Sync	Lag	Fast	Slow	Precision	Recall
Sync	360	0	0	0	0.986	1
Lag	0	120	0	0	0.857	1
Fast	0	0	60	0	1	1
Slow	5	20	0	35	1	0.583

TABLE I: Confusion Matrix on left dataset

Status	Sync	Lag	Fast	Slow	Precision	Recall
Sync	240	78	40	2	0.872	0.667
Lag	10	93	17	0	0.419	0.775
Fast	11	17	25	8	0.298	0.410
Slow	14	34	2	10	0.5	0.167

TABLE II: Confusion Matrix on push dataset

provides the multi-class precision and recall information. The tables reveal that the accuracy drops on push dataset since the sensor data only vary significantly on one axis.

Fig. 6 shows the sample output when applying time domain analysis and frequency domain analysis under five cases, where a small block represents a time window and different colors of the block represents different user status. The figure shows a total length of two-minute data analysis with the window size set to 5 seconds, so for each case, there are 24 time windows analyzed, the precision values on the right indicate the ratio of the correctly identified time windows.

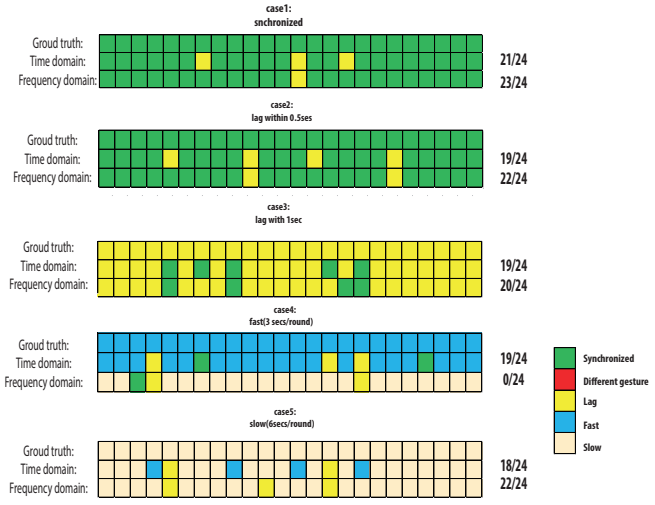


Fig. 6: Five cases

B. System Implementation

A Nexus 5 with Android 6.0.1, API 23 and a LG Watch Urbane with Android 7.1.1, API 25 were used in the implementation. The PC server was hosted on a 2017

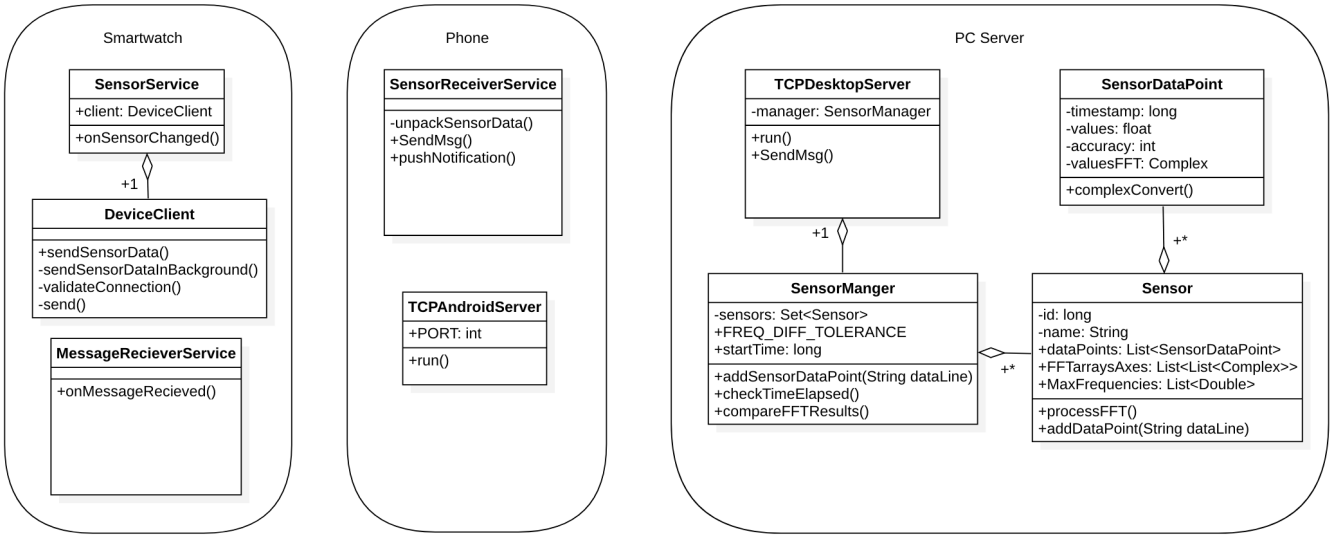


Fig. 7: Overview of system implementation

Macbook Pro. The application was based off the open-source project SensorDashboard by GitHub user pocmo under the Apache License 2.0. The project provided the structure to send sensor data from a smart watch to a smart phone. To better fit the desired prototype, we added communication between the phone and PC server and between the phone and watch. Fig, 7 shows the overview of the system implementation.

As only one watch was used, previously recorded sensor data simulated another watch. Two types of periodic motions were examined: an upwards-facing stationary watch and a waving motion restricted to the $y-z$ plane with a period of 5 seconds. All of the processing is handled by the PC server. After collecting 10 seconds of sensor data from the watch, the SensorManager preforms the FFT on all the collected accelerometer data. If the most occurring frequency for a particular watch was not within a range (± 0.005Hz) of the designated leader, then the PC server would send back either Fast or Slow. Otherwise the server would send back Sync. Once the phone receives the message, it notifies the current wearer. The latency for the round trip time between the watch and the server is about 0.1 second on average.

V. Conclusion

In this paper, we target a specific application scenario where multiple users try to perform same gestures at the same time such as a group fitness lesson, group dance lesson, orchestra. We use data collected from wearable devices and apply signal processing algorithms to determine whether a user is in in sync with the reference user such as a coach or an instructor. We evaluated our approaches using data generated based on real data collected from smart watches. We also developed a prototype system on Android smartwatches and smartphones. Our simulation and actual system results demonstrate that group gesture synchronicy status can be accurately identified.

References

[1] R. Cachucho, M. Meeng, U. Vespier, S. Nijssen, and A. Knobbe. Mining multivariate time series with mixed sampling rates. In *Proceedings of the 2014 ACM Conference on Ubiquitous Computing (UbiComp '14)*, pages 413–423, 2014.

[2] D. Gordon, J. Hanne, M. Berchtold, A. A. N. Shirehjini, and M. Beigl. Towards collaborative group activity recognition using mobile devices. *Mobile Networks and Applications*, pages 326–340, 2013.

[3] Dawud Gordon, Martin Wirz, Daniel Roggen, Gerhard Tröster, and Michael Beigl. Group affiliation detection using model divergence for wearable devices. In *Proceedings of the 2014 ACM International Symposium on Wearable Computers (ISWC'14)*, pages 19–26, 2014.

[4] M. B. Kjargaard, M. Wirz, D. Roggen, and G. Troster. Mobile sensing of pedestrian flocks in indoor environments using wifi signals. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'12)*, pages 95–102, 2012.

[5] R. Sen, Y. Lee, K. Jayarajah, A. Misra, and R. K. Balan. Grumon: Fast and accurate group monitoring for heterogeneous urban spaces. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems(SenSys'14)*, pages 46–60, 2014.

[6] Na Yu, Yongjian Zhao, Qi Han, Weiping Zhu, and Hejun Wu. Identification of partitions in a homogeneous activity group using mobile devices. *Mobile Information Systems*, 2016:3545327:1–3545327:14, 2016.