WNFA Final Project Proposal - Team 3

Gesture Recognition Using WiFi

Huan-Ting Chen
National Taiwan University
Taiwan
R10921045@ntu.edu.tw

Pei-Shin Hwang
National Taiwan University
Taiwan
R10921072@ntu.edu.tw

Yo-Cheng Chang
National Taiwan University
Taiwan
R10921102@ntu.edu.tw

Jian-Han Wu
National Taiwan University
Taiwan
R10921065@ntu.edu.tw

KEYWORDS

Gesture recognition, Wi-Fi, Channel State Information (CSI)

1 INTRODUCTION

In the US, about 28 percent (14.7 million) of community-dwelling older adults live alone. Due to their physical limitations, there are several problems like health care, real-time first-aid and inconvenience of daily lifestyle. Based on this issue, We would like to design a program for the elderly based on the concept of smart home. The system can reach human's need at home (eg. on-off light) by recognizing some simple actions like pushing, kicking or drawing a circle in the air, etc.

Human gesture recognition is the core enabler for a wide range of applications such as smart home, security surveillance and virtual reality. By using an adapted Wi-Fi router and a few wireless devices in the living room, users could control their electronics and household appliances from any room in the home with a sample gesture. Nowadays, with the population aging, the healthcare of the elderly cannot be overemphasized, especially those living on their own. Studies have achieved device-free activity recognition by monitoring how human activities affect the channel characteristics (so-called channel state information (CSI)). This leads to a burgeoning research field of wireless sensing[1][2][4]. The authors in [7] propose SignFi to recognize sign language gestures using WiFi based on measured CSI and a Convolutional Neural Network (CNN) as the classification algorithm.

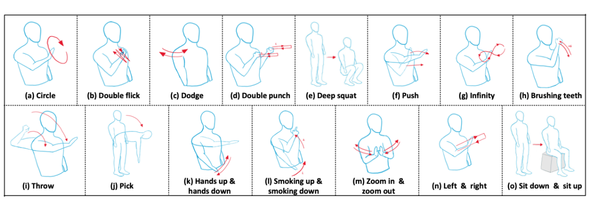


Figure 1: An image of gestures [6]

In this project, we aim to design a program that collects CSI from a Wi-Fi router, and recognizes seven actions by training a CNN model based on the collected data. Here, we are going to implement Wi-Fi gestures according to theses resources [3], [6], [8], [10], [11] and create an intelligent household environment.

2 EQUIPMENT

- Wifi AP
- Antennas

3 BACKGROUND

3.1 Channel State Information

In a wireless communication system, the Channel State Information (CSI) can be obtained from commodity Wi-Fi network interface cards (NICs). CSI describes how a signal propagates from the transmitter to the receiver and represents the combined effect of, for example, scattering, fading, and power decay with distance. CSI entry represents the Channel Frequency Response:

$$H(f;t) = \sum_{n}^{N} a_n(t) e^{-j2\pi f \tau_n(t)} \tag{1}$$

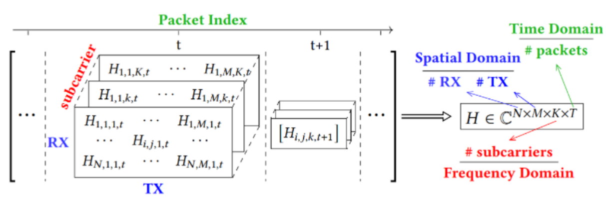


Figure 2: The structure of CSI

For a MIMO-OFDM channel with M transmit antennas, N receive antennas, and K subcarriers, the CSI matrix is a 3D matrix $NXMXK$ representing amplitude attenuation and phase shift of multi-path channels. The structure can be referred to the following figure. For subcarriers in CSI, it complies with 802.11a standards, containing three kinds of subcarriers: data subcarriers, pilot subcarriers and unused subcarriers. Therefore, the detected CSI has some carriers always begin 0, as shown in Fig 3.

3.2 Beacon Frame

Beacon frame is one of the management frames defined in IEEE 802.11., containing information about the network. Beacon frames are transmitted periodically, they serve to announce the presence of a wireless LAN and to synchronise the members of the service set.

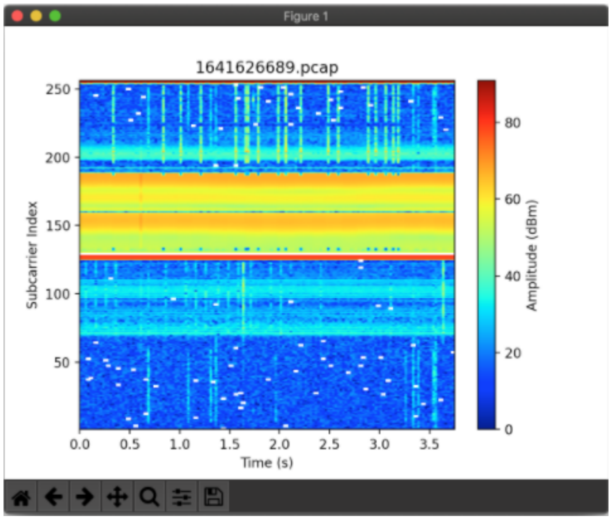


Figure 3: The data of CSI

We take advantage of the periodical characteristic of the beacon so that we can send requests with certain periods repeatedly.

One special thing to be noted is the frame control. In 802.11, frames starting with 0x80 are beacon frames, and will always be 20 MHz. They announce the presence of the router. Hence, we set our beacon as 0x80. The patterns in beacon frames can be referred to Fig 4. The usage of the control will be noted later in our environment settings.

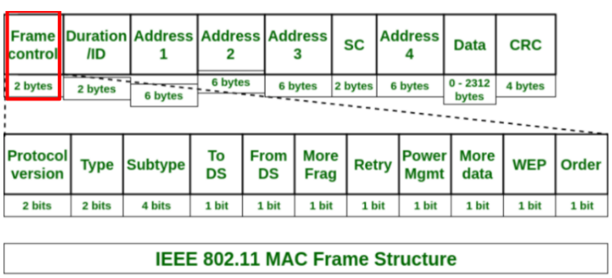


Figure 4: MAC structure.

4 SYSTEM DESIGN

Our gesture recognition system is composed of three components: transmitter (Tx), receiver (Rx), and edge server. The overall system architecture is shown on figure 5. We adopt Asus RT-N66U Wireless router as our Tx, Raspberry Pi 4B as our Rx, and a Laptop as the edge server. Since the limited computing resource of Raspberry Pi and the operating system is only 32-bit on Raspberry Pi, we cannot perform gesture recognition algorithms on the Raspberry Pi.

The system control flow can separate into four stages:

(1) Tx broadcasts the Wi-Fi beacons to the Rx.
(2) Rx only captures the Wi-Fi beacon packets.
(3) When capturing some amounts of the packets, save these packets to a file.
(4) Transmit this file to the edge server and get the gesture recognition result.

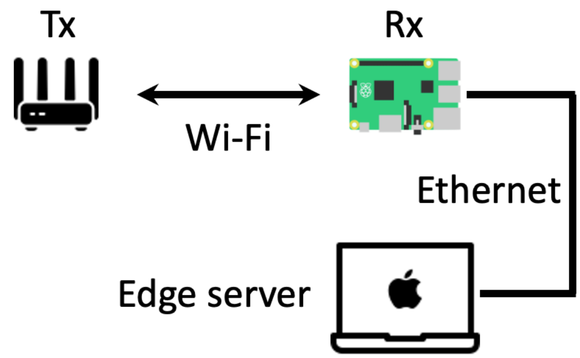


Figure 5: System architecture

5 SYSTEM SETUP

We have to setup three components in our system individually.

5.1 Transmitter Setup

(1) Select the control channel to 157 and set the channel bandwidth to 40 MHz.
(2) Set the Wi-Fi beacon interval to 20 ms.

5.2 Receiver Setup

(1) Install the operating system with kernel version 5.10
(2) Compile and install the firmware patches provide by seemon-lab/nexmon-csi [9]
(3) Kill the `wpasupplicant` process which is related to wireless control.
(4) Generate channel infomation parameters by the command: `makecsiparams -c 157/80 -C 1 -N 1 -m <Tx MAC address> -b 0x80`
(5) Configure the packet extractor by the command: `nexutil -Iwlan0 -s500 -b -l34 <parameters>`
(6) Enable monitor mode on wlan0
(7) Collect the packets by `tcpdump`
(8) Connect to Ethernet

5.3 Edge Server Setup

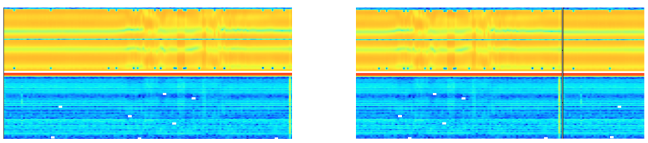
(1) Clone the source code from our GitHub repository dingyiyi0226/gesture-recognition-csi
(2) Install requirements
(3) Start the server by executing the script `server.sh`

6 GESTURE RECOGNITION FLOW

6.1 Data preprocessing and data augmentation

We adopt two method to do data preprocessing. First, we have to extract the CSI data from the packet. We implement these function by utilizing the toolkit built by Gi-z/CSIKit [5]. Secondly, removing the extreme values is an important step in data preprocessing. Since we have to calculate the channel information in log scale, we clip the minimum value to -20 dBm.

The time that detect the gesture is not fixed. Therefore, we implement a random horizontal shift to our CSI data in data augmentation step as shown in figure 6. The maximum horizontal shift is 0.3 times of the signal length.



(a) Original **(b) After horizontal shift**

Figure 6: Data augmentation

6.2 Model Training

The CNN model that we build for our gesture recognition includes four convolution layers, three linear layers, three pooling layers, and four dropout layers. The detailed model parameters is shown in figure 7.

We split the original data into training set, validation set, and testing set with the ratio of 6:2:2. After finding the best configurations of the model, we merge the training set and validation set to train the model again.

```
CNN2(
  (cnn): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU(inplace=True)
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (11): Dropout(p=0.2, inplace=False)
    (12): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU(inplace=True)
    (15): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (16): Dropout(p=0.2, inplace=False)
  )
  (fc): Sequential(
    (0): Linear(in_features=36864, out_features=256, bias=True)
    (1): Dropout(p=0.3, inplace=False)
    (2): Linear(in_features=256, out_features=64, bias=True)
    (3): Dropout(p=0.25, inplace=False)
    (4): Linear(in_features=64, out_features=7, bias=True)
  )
)
```

Figure 7: CNN model

6.3 Edge server

We build the gesture recognition service by the Flask package, which is the widely-used web server framework. The server listens to a POST `/gesture` request after it starts the service. Once the client send the request with the file containing the packets, the server would response with the inference result

7 EXPERIMENT

7.1 Setup

The place we perform our experiment is a sparse room, with Rx (Raspberry Pi 4B), Tx (wireless router) being placed on the desk, and Rx is 4 meters away from Tx. Subjects are seated on a chair between Rx and Tx while performing different gestures. We define seven gestures: *Circle, Clap, Slide, Kick, Push, Stand up,* and *Sit*, where *Kick* means subject kicking with single leg, *Sit* equals to no action at all, and the other actions are performed with single hand.

7.2 Dataset

We collect our dataset from four subject, including 3 males and 1 female. Each participant performs each gesture 50 times. All gestures are performed in a 3-second period, with 50 Hz sampling rate. The sampling rate comes the limitation of router, which can only broadcast beacon frames with minimum interval of 20 ms. We obtain 1400 CSI measurements in total (4 users $\times$ 7 gestures $\times$ 50 instances).

7.3 Experiment Steps

We repeat the following steps in our experiment:

(1) Collect data by the command:
```
sudo tcpdump -i wlan0 dst port 5500
-c 150 -w <output file name>
```
(2) Send data to server by the command:
```
curl -X POST <server_ip>:5000/gesture
-F 'csi=@"<file name>"'
```
(3) Get the result from server

Each time we capture data using 3 second. The capured packets are visualized and shown in figure 8.

8 RESULT

In figure 9, One can see that after almost 20 to 30 epoch of training, the CNN model has decreased the error rate to lower than 0.5 percent. Our gesture recognition system derives the gesture prediction from 7 possible gesture candidates within at most 0.5 seconds regardless of the person and reaches high precision, 0.943. From the confusion matrix of the gesture recognition model, shown in figure 10, we can see that the performance is stunning.

9 DISCUSSION

We find that if we reboot the Rpi, the data we catch from the same gesture, shown in figure 11, varies entirely. We infer it is because the situation of the antenna in Rpi depends on the factor in environments like temperature. Getting an external antenna, we may minimize the proportion of the situation of the antenna that changed by reboot.

10 FUTURE WORK

(1) more gestures to predict
(2) make the prediction without position and orientation dependence

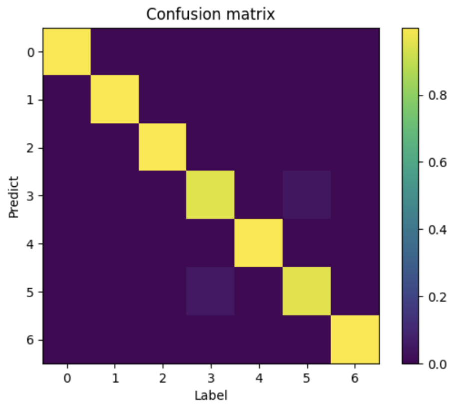


Figure 10: Confusion Matrix

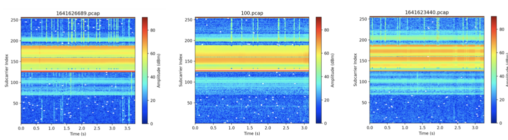
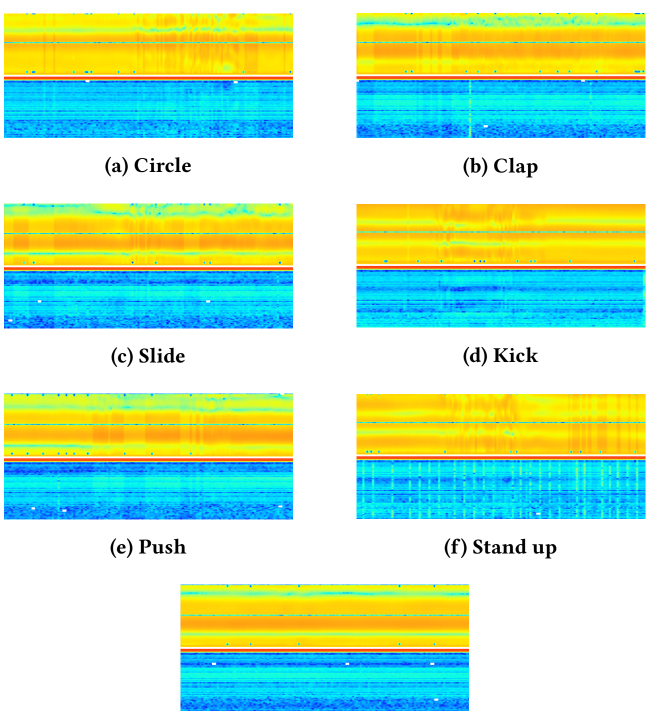


Figure 11: Inconsistent Data



(a) Circle **(b) Clap**

(c) Slide **(d) Kick**

(e) Push **(f) Stand up**

(g) Sit

Figure 8: Visualized CSI packets of gestures

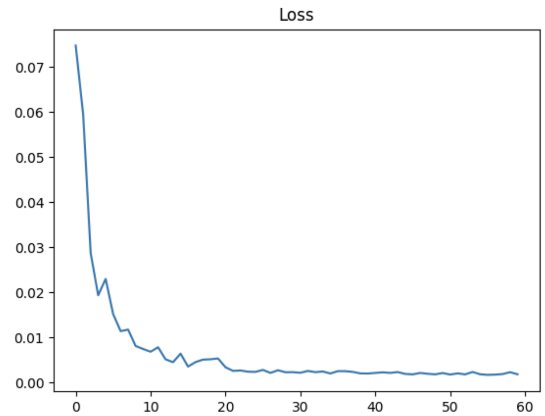


Figure 9: The Error Rate Versus Epoch

(3) multiple transmission and receive antennas to increase communication diversity thus further improving the accuracy of the prediction
(4) receive and analyze consecutively to make the whole gesture prediction system more useful

REFERENCES

[1] Heba Abdelnasser, Moustafa Youssef, and Khaled A. Harras. 2015. WiGest: A ubiquitous WiFi-based gesture recognition system. *2015 IEEE Conference on Computer Communications (INFOCOM)* (2015), 1472–1480.

[2] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. 2017. Recognizing Keystrokes Using WiFi Devices. *IEEE Journal on Selected Areas in Communications* 35, 5 (2017), 1175–1190. https://doi.org/10.1109/JSAC.2017.2680998

[3] Jen-Yin Chang, Kuan-Ying Lee, Yu-Lin Wei, Kate Ching-Ju Lin, and Winston Hsu. 2016. Location-Independent WiFi Action Recognition via Vision-Based Methods. In *Proceedings of the 24th ACM International Conference on Multimedia* (Amsterdam, The Netherlands) *(MM '16)*. Association for Computing Machinery, New York, NY, USA, 162–166. https://doi.org/10.1145/2964284.2967203

[4] Liqiong Chang, Jiaqi Lu, Ju Wang, Xiaojiang Chen, Dingyi Fang, Zhanyong Tang, Petteri Nurmi, and Zheng Wang. 2018. SleepGuard: Capturing Rich Sleep Information Using Smartwatch Sensing Data. *Proceedings of ACM on interactive, mobile, wearable and ubiquitous technologies* 2, 3 (Sept. 2018). https://doi.org/10.1145/3264908

[5] Glenn Forbes. 2021. CSIKit: Python CSI processing and visualisation tools for commercial off-the-shelf hardware. https://github.com/Gi-z/CSIKit

[6] Xinyi Li, Liqiong Chang, Fangfang Song, Ju Wang, Xiaojiang Chen, Zhanyong Tang, and Zheng Wang. 2021. CrossGR: Accurate and Low-Cost Cross-Target Gesture Recognition Using Wi-Fi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 1, Article 21 (mar 2021), 23 pages. https://doi.org/10.1145/3448100

[7] Yongsen Ma, Gang Zhou, Shuangquan Wang, Hongyang Zhao, and Woosub Jung. 2018. SignFi: Sign Language Recognition Using WiFi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 23 (mar 2018), 21 pages. https://doi.org/10.1145/3191755

[8] Yongsen Ma, Gang Zhou, Shuangquan Wang, Hongyang Zhao, and Woosub Jung. 2018. SignFi: Sign Language Recognition Using WiFi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 23 (mar 2018), 21 pages. https://doi.org/10.1145/3191755

[9] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. 2017. Nexmon: The C-based Firmware Patching Framework. https://nexmon.org

[10] Aditya Virmani and Muhammad Shahzad. 2017. Position and Orientation Agnostic Gesture Recognition Using WiFi. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services* (Niagara Falls, New York, USA) *(MobiSys '17)*. Association for Computing Machinery, New York, NY, USA, 252–264. https://doi.org/10.1145/3081333.3081340

[11] Yue Zheng, Yi Zhang, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. 2019. Zero-Effort Cross-Domain Gesture Recognition with Wi-Fi. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 313–325. https://doi.org/10.1145/3307334.3326081