국민대학교
소프트웨어융합대학
소프트웨어학부

관리번호 : 2018-05

캡스톤 디자인 I
종합설계 프로젝트

프로젝트 명	WiBi
팀 명	MobiCom
문서 제목	최종 보고서

날짜	2018년 05월 29일

팀원	이경재 (조장)
	김태기
	양재영
	박수찬
지도교수	김상철 교수

캡스톤디자인 I (2018-05)

WiBi

국민대학교 소프트웨어학부

제 출 문

본 보고서를 캡스톤 디자인 I 교과목의 종합설계 프로젝트 최종 보고서로 제출합니다.

2018. 05. 29

프로젝트 명 : WiBi

팀 구성원 : 이 경재 (팀장)
김 태기
양 재영
박 수찬

지도교수 : 김상철 교수

목 차

제 1 장

요약

현대의 WiFi는 지하철, 가정집, 심지어는 도로 한복판 등 어디라도 빠르고 간단하게 접할 수 있는 대표적인 매체로 자리매김해 있다. 이렇게 보편화된 인프라에도 불구하고,여태까지의 WiFi의 주된 사용처는 무선 인터넷을 이용하는 것으로만 한정되어있었다.

WiBi는 WiFi의 새로운 사용처를 제시하기 위하여, WiFi의 CSI(Channel State Information, 채널 상태 정보) 를 이용하여 사용자의 행동을 인식하는 CSI 인식 장치를 제작하는 프로젝트이다.

CSI를 이용한 WiBi는 2개의 NIC(Network Interface Card)가 서로 주고받는 전파의 사이에 사용자를 두고, 사용자의 움직임을 카메라나 적외선 센서 등 다른 장비의 도움 없이 WiFi신호의 변화에 대한 인식으로만 사용자의 행동을 구분할 수 있게 만든다.

이로 인해 여태까지의 카메라나 센서가 가진 LoS(Line of Sight) 약점을 극복할 수 있고, WiFi를 사용하기 때문에 뛰어난 접근성과 편리성을 가진다는 차별성을 가지고 있다.

제 2 장

수행 계획서

2.1 수행 계획서

Wifi CSI란

CSI(Channel State Information)란 통신 링크의 채널 속성들을 지칭한다. 이 정보는 어떻게 신호가 전송기로부터 수신기로 전파가 되는지를 설명해주고 합쳐져서 일어나는 효과들 즉, 산란, 페이딩과 거리에 따른 세기의 감소를 나타내준다. CSI는 송신들이 현재 채널 상태들에 적용하게 가능하게 해준다. 이것은 멀티안테나 시스템에서 높은 전송률을 가능하게 해주는 안정적인 통신을 가능하게 해준다.

사람의 움직임은 무선 신호의 반사에 영향을 미쳐 CSI변화를 만들어내게 된다. 그 과정에서 다양한 CSI 결과를 보여준다. 다양한 행동들에서 나타나는 CSI 데이터 스트림을 분석하고 그들을 저장된 모델들과 비교를 함으로써 사람 행동을 분석할 수 있다.

와이파이를 이용한 기술의 장점

이 프로젝트의 주요한 장점은 3가지로 볼 수 있다. 첫번째로, 위에서 언급 한 것과 같이 와이파이를 데이터 전송 용도 이외에 소모되는 와이파이 신호를 활용할 수 있다. 두번째로, 와이파이를 사용한 행동 인식은 LoS(Line of Sight)를 극복 할 수 있다. 마지막으로 카메라나, 레이저 인식 단말과는 다르게 와이파이 라우터는 많은 곳에 이미 설치 되어 있다. 기존의 Home IoT의 input 단말은 카메라나, 레이저 입출력 단말이다. 카메라를 이용한 입력은 시각의 범위에 제한이 생긴다. 카메라 앞에 장애물이 있으면 시스템은 의도한 대로 작동 하지 않을 것이다. 또한 장애물이 없다고 하더라도 설치하는 위치에 따라서 생길 수 있는 문제인 사각지대가 형성된다. 레이저 입출력 또한 빛을 이용하므로 이러한 LoS에서 자유롭지 못하다. 반면 CSI는 파동으로 이러한 기존의 문제점을 극복 할 수 있다. 특히 위에서 해결되지 못했던 사각지대를 없앨 수 있다는 점이다. 따라서 Home IoT 시스템이 기존의 의도대로 작동 할 수 있는 확률을 높일 수 있다. WiFi CSI는 미국에서도 2014년경에 처음 연구가 시작된 비교적 신생 분야이기 때문에, WiFi CSI의 잠재력을 볼 때 잠재적 시장가치는 지속적으로 상승 할 것이라고 생각한다.

WiBi의 장점

1. **기존의 와이파이 인프라를 사용 할 수 있다.**
   기존의 Home IoT는 카메라나, 레이저 인식 장치와 같은 추가적인 장비의 설치가 필요하다. 반면에 각 가정에는 모두 하나씩 와이파이 라우터를 가지고 있다. 이 와이파이 라우터를 이용해서 카메라나 레이저장비가 하는 작업을 대체 할 수 있다.

2. **카메라에 비해 사생활 침해의 가능성이 적다.**
   인터넷에 연결 되어 있는 카메라는 항상 해킹을 통한 사생활 유출의 위험 이 있다. 반면에 CSI를 이용한 행동인식의 경우 영상을 통해 인식을 하는 것이 아니기 때문에 사생활 침해의 요소가 적다.

3. **기존의 광학장비가 작동하지 않는 환경 에서도 사용 가능하다.**
   사각 지대에 대한 커버가 카메라나 레이저 장비에 비해 수월하다. 카메라나 레이저 장비의 경우 빛을 이용하여 인식 하는 장비 이므로 장애물에 영향을 많이 받는다. Line of Sight를 벗어나는 경우는 인식에 어려움이 있다. WiFi CSI는 빛이 아닌 신호를 이용한다. 신호는 파동의 성질을 갖고 있기 때문에 장애물이 있는 경우에도 행동 인식이 가능하다. 따라서 기존의 장비들이 갖는 한계점을 극복 할 수 있다. 또한 광학장비는 낮은 광도의 환경에서는 동작이 제한 되는 것에 비해, CSI는 이러한 환경에 전혀 관계 없이 작동 가능하다.

4. **카메라나 레이저에 비해 단말 설치를 적게 해도 되므로 저렴하다.**
   위의 3번 항목과 연관이 되어 있다. 카메라나 레이저 장비는 LoS를 최소화 하기 위해 최대한 많은 곳에 단말을 설치 하여야 한다. 그러나 Wifi CSI를 이용하면 하나의 라우터로 좀더 넓은 공간을 커버 할 수 있다. 주택 자동화 시스템을 사용하기 위해 여러 개의 카메라나 레이저 인식 단말을 설치 하는 대신 기존에 있던 단 하나의 라우터로 모든 서비스를 제공 할 수 있으니 성능과 비용의 문제를 모두 잡을 수 있다.

5. **좀 더 제한된 환경에서 사용 할 수 있다.**
   WiFi CSI는 사람이 있는가 없는가 뿐만 아니라, 몇명이 있는가 까지 알아 낼 수 있다. 만약 인증되지 않은 사용자가 집 주변에 접근 하는 것을 발견 한다면 등록되어 있는 사용자의 스마트폰 앱으로 푸시 알람을 보내어 현재 상황을 보고 할 수 있다. 이 방법의 장점은 무분별하게 카메라로 수집되는 타인의 신상정보를 WiFi를 이용함으로써 특정한 형태로 제한을 할 수 있다는 것이다. 이 방법은 사생활 침해가 우려되는 상황에서 카메라에 비해 선택의 우위를 가질 수 있을 것이다.

제 3 장

중간 보고서

3.1 중간 보고서
3.2 수행 내용

3.1 계획서 상의 연구내용

3.1.1 행동 샘플 추출 단계

이 프로젝트에서 사람의 움직임을 파악하기 위하여 가장 먼저 해야할 것이 행동 샘플을 추출하는 단계이다. 그 이유는 먼저 텐서 플로우를 사용한 기계학습을 시키기 위해서는 정확한 행동에 해당하는 샘플을 바탕으로 학습을 시켜야 하기 때문이다. 이에 대한 이유는 정확한 행동이 있어야 그 다음에 사람의 행동이 들어오면 정확한 데이터를 가지고 분석해서 결과를 도출해 낼 수 있기 때문이다. 이 단계에서 추출되는 샘플 데이터 값은 그래프로 나타내질 수 있으며, 그 그래프는 진폭, 위상, SNRdB 등으로 나타낼 수 있다.

3.1.2 샘플을 이용한 학습 단계

정확한 데이터를 가진 샘플을 추출한 이후 텐서 플로우에 그 샘플을 학습 시킨다. 텐서 플로우에 샘플을 학습 시키면 ckpt 파일로 변환되어서 나오게 된다. 다음 단계에서는 이 ckpt 파일을 이용을 하게 된다.

3.1.3 학습을 바탕으로 사람 움직임 검증 단계

검증을 하기 위해서 따로 제한된 공간에서 사람의 움직임을 측정하는 실험을 한다. 현재 진행되고 있는 측정 방법은 두 개의 NIC 사이에서 패킷이 전송되는 과정에서 송신되는 전파의 사이에 있는 장애물에 따른 전파의 변화를 수치화해서 측정하는 방법이다. 이 사람의 움직임에 대한 변화를 적절한 방법을 통해 유의미한 데이터로 진폭, 위상 등으로 변환하고 이를 학습시킨 머신에 적용하여 제스쳐를 인식할 수 있다.

3.1.4 와이파이와의 연동 단계

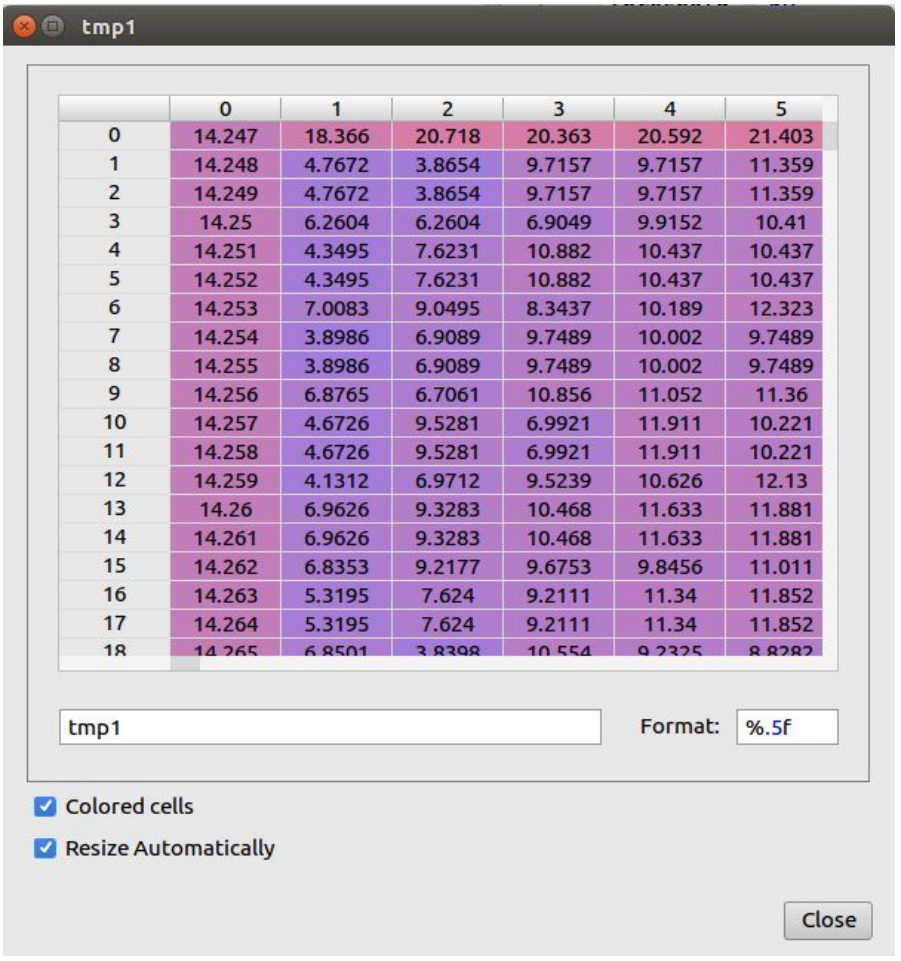
지금까지는 두 개의 NIC 만 상호통신을 하지만 이후 애플리케이션 연동, 홈 IoT 등의 활용을 위해 다른 WiFi 도 받아들여 사용할 수 있게 할 수 있게 연구중이다.

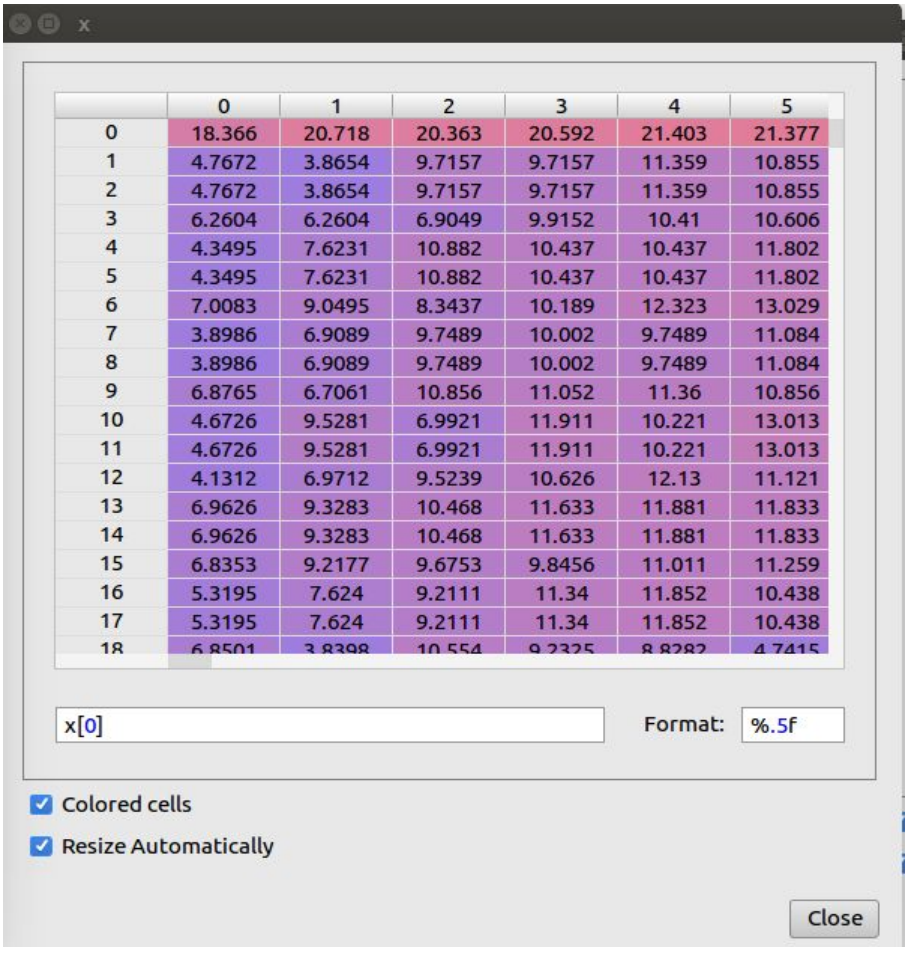
3.2 수행 내용
3.2.1 개요

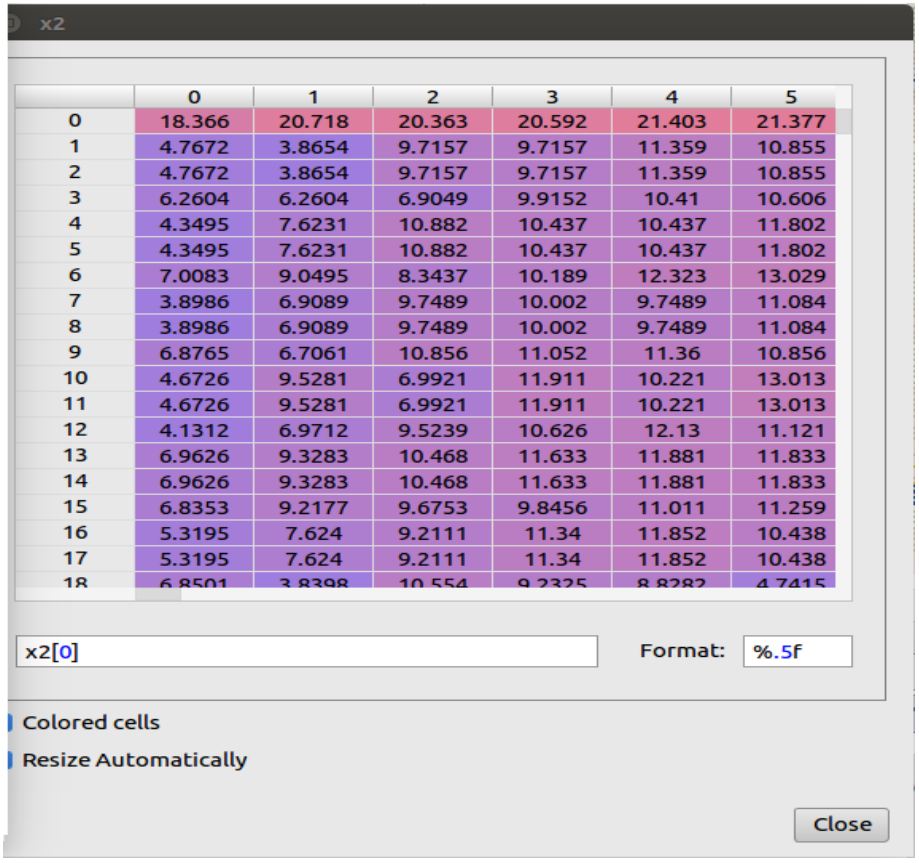
```
12          xx = np.empty([0,window_size,90],float)
13          yy = np.empty([0,8],float)
14
15          ###Input data###
16          #data import from csv
17          input_csv_files = sorted(glob.glob(path1))
18          for f in input_csv_files:
19                  print("input_file_name=",f)
20                  data = [[ float(elm) for elm in v] for v in csv.reader(open(f, "r"))]
21                  tmp1 = np.array(data)
22                  x2 =np.empty([0,window_size,90],float)
23
24                  #data import by slide window
25                  k = 0
26                  while k <= (len(tmp1) + 1 - 2 * window_size):
27                          x = np.dstack(np.array(tmp1[k:k+window_size, 1:91]).T) #매번 시작점을
28                          x2 = np.concatenate((x2, x),axis=0)
29                          k += slide_size
```

먼저 이 부분에서는 정확한 정보를 바탕으로 한 데이터를 전처리하는 과정이다.

전처리 하는 과정은 데이터의 정보를 보면 어떤 것을 제거하고 어떤 것을 추가하는지를 알 수 있다.


tmp1
0 1 2 3 4 5
0 14.247 18.366 20.718 20.363 20.592 21.403
1 14.248 4.7672 3.8654 9.7157 9.7157 11.359
2 14.249 4.7672 3.8654 9.7157 9.7157 11.359
3 14.25 6.2604 6.2604 6.9049 9.9152 10.41
4 14.251 4.3495 7.6231 10.882 10.437 10.437
5 14.252 4.3495 7.6231 10.882 10.437 10.437
6 14.253 7.0083 9.0495 8.3437 10.189 12.323
7 14.254 3.8986 6.9089 9.7489 10.002 9.7489
8 14.255 3.8986 6.9089 9.7489 10.002 9.7489
9 14.256 6.8765 6.7061 10.856 11.052 11.36
10 14.257 4.6726 9.5281 6.9921 11.911 10.221
11 14.258 4.6726 9.5281 6.9921 11.911 10.221
12 14.259 4.1312 6.9712 9.5239 10.626 12.13
13 14.26 6.9626 9.3283 10.468 11.633 11.881
14 14.261 6.9626 9.3283 10.468 11.633 11.881
15 14.262 6.8353 9.2177 9.6753 9.8456 11.011
16 14.263 5.3195 7.624 9.2111 11.34 11.852
17 14.264 5.3195 7.624 9.2111 11.34 11.852
18 14.265 6.8501 3.8398 10.554 9.2325 8.8282
tmp1
Format: %.5f
Colored cells
Resize Automatically
Close


x
0 1 2 3 4 5
0 18.366 20.718 20.363 20.592 21.403 21.377
1 4.7672 3.8654 9.7157 9.7157 11.359 10.855
2 4.7672 3.8654 9.7157 9.7157 11.359 10.855
3 6.2604 6.2604 6.9049 9.9152 10.41 10.606
4 4.3495 7.6231 10.882 10.437 10.437 11.802
5 4.3495 7.6231 10.882 10.437 10.437 11.802
6 7.0083 9.0495 8.3437 10.189 12.323 13.029
7 3.8986 6.9089 9.7489 10.002 9.7489 11.084
8 3.8986 6.9089 9.7489 10.002 9.7489 11.084
9 6.8765 6.7061 10.856 11.052 11.36 10.856
10 4.6726 9.5281 6.9921 11.911 10.221 13.013
11 4.6726 9.5281 6.9921 11.911 10.221 13.013
12 4.1312 6.9712 9.5239 10.626 12.13 11.121
13 6.9626 9.3283 10.468 11.633 11.881 11.833
14 6.9626 9.3283 10.468 11.633 11.881 11.833
15 6.8353 9.2177 9.6753 9.8456 11.011 11.259
16 5.3195 7.624 9.2111 11.34 11.852 10.438
17 5.3195 7.624 9.2111 11.34 11.852 10.438
18 6.8501 3.8398 10.554 9.2325 8.8282 4.7415
x[0]
Format: %.5f
Colored cells
Resize Automatically
Close



위 그래프는 tmp1, x, x2에 대한 시각적인 자료를 보여준다. 처음 첫 열은 timestamp에 해당한다.

먼저 어떠한 데이터를 추가하고 제거하는지 설명을 하면, 이 데이터는 timestamp, phase, amplitude의 세가지로 구성되어있다. 그 중에 timestamp 부분은 잘려 나가게 된다. 학습을 시키는 과정에서 필요하지 않은 정보이기 때문이다. timestamp를 자르는 동시에 amplitude에 해당하는 데이터를 x2라는 배열을 이용하여 차례대로 저장을 시킨다.

현재 이 부분에서 실행하는 과정은 한 가지의 행동에서 여러개의 샘플에 대한 각각의 amplitude만을 모으는 과정이다.

```
31                  xx = np.concatenate((xx,x2),axis=0)
32          xx = xx.reshape(len(xx),-1)
```

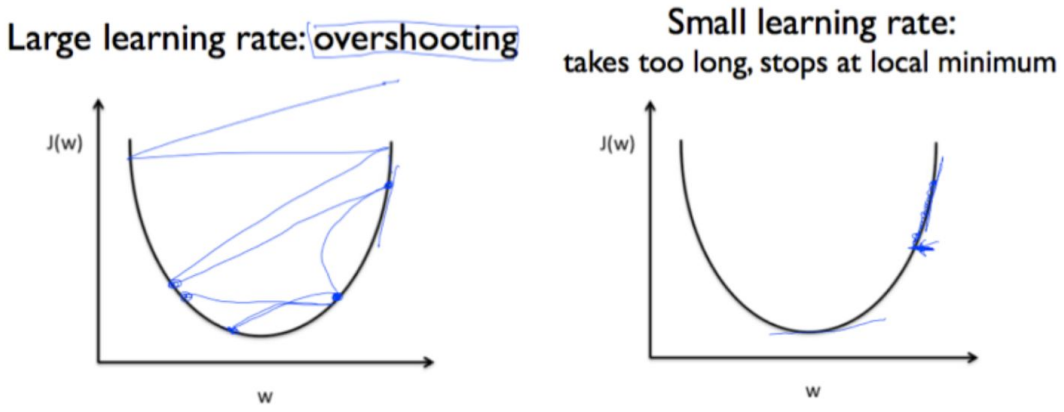
다음에 실행하는 부분은 이 부분인데, 이 부분에서는 지금까지 각각의 샘플에 대한 amplitude를 하나의 배열로 만들어 주는 부분이다. 즉 하나의 행동에 대하여 관하여 모아놓은 amplitude들을 하나의 파일로 만들기 위하여 하는 과정이라고 볼 수 있다.

```
for i, label in enumerate (["bed", "fall", "pickup", "run", "sitdown", "standup", "walk"]):
        filepath1 = "./Dataset/input_*" + str(label) + "*.csv"
        filepath2 = "./Dataset/annotation_*" + str(label) + "*.csv"
        outputfilename1 = "./input_files/xx_" + str(window_size) + "_" + str(threshold) + "_" + label + ".csv"
        outputfilename2 = "./input_files/yy_" + str(window_size) + "_" + str(threshold) + "_" + label + ".csv"

        x, y = dataimport(filepath1, filepath2)
        with open(outputfilename1, "w") as f:
                writer = csv.writer(f, lineterminator="\n")
                writer.writerows(x)
        with open(outputfilename2, "w") as f:
                writer = csv.writer(f, lineterminator="\n")
                writer.writerows(y)
        print(label + "finish!")
```

마지막 메인 부분에서 각각의 파일에 대하여 이름을 만들어 주는 과정을 하고 있다. 현재 사용하는 정확한 샘플은 침대에 누워있을 때, 사람이 넘어질 때, 무언가를 줍는 행동을 할 때, 달리는 행동, 앉는 행동, 일어서는 행동과 마지막으로 걷는 행동을 사용하고 있다.

다음은 샘플을 이용하여 텐서 플로우에 학습을 시키는 과정이다.



이 그래프는 Gradient decent로 불리는 학습을 시키는 과정에서 사용하는 알고리즘을 시각적으로 보여준다. 이 안에서 learning rate 가 너무 커지게 되면 이 알고리즘의 궁극적인 목표인 최저점을 찾는 과정을 하지 못할 수도 있고 그래프의 밖으로 빠져나가는 현상도 벌어질 수 있다. 하지만 반대로 learning rate가 너무 작게 되면 이 그래프의 최저점을 찾아가는 과정에서 시간이 너무 많이 걸릴 수가 있다. 그 현상을 오른쪽 그래프에서 보여진다. 파란점의 움직임이 너무 조금씩 움직이는 바람에 언제 이 알고리즘이 언제 끝날지 모르는 것이다. 그래서 적절한 learning rate를 찾아서 입력해 주는것이 중요하다.

```
# Keep training until reach max iterations
while step < training_iters:
    batch_x, batch_y = wifi_train.next_batch(batch_size) #wifi_train에 저장된 x와 y를 배
    x_vali = wifi_validation.images[:]
    y_vali = wifi_validation.labels[:]
    # Reshape data to get 28 seq of 28 elements
    batch_x = batch_x.reshape((batch_size, n_steps, n_input)) #batch_x를 500행 90열 ba
    x_vali = x_vali.reshape((-1, n_steps, n_input)) #x_vail은 500행, 90열로 만들고 남은걸
    # Run optimization op (backprop)
    sess.run(optimizer, feed_dict={x: batch_x, y: batch_y})

    # Calculate batch accuracy
    acc = sess.run(accuracy, feed_dict={x: batch_x, y: batch_y})
    #x에 batch_x, y에 batch_y를 입력하여 accuracy 실행하여 결과값을 acc에 저장
    acc_vali = sess.run(accuracy, feed_dict={x: x_vali, y: y_vali})
    # Calculate batch loss
    loss = sess.run(cost, feed_dict={x: batch_x, y: batch_y})
    loss_vali = sess.run(cost, feed_dict={x: x_vali, y: y_vali})

    # Store the accuracy and loss
    train_acc.append(acc)
    train_loss.append(loss)
    validation_acc.append(acc_vali)
    validation_loss.append(loss_vali)

    if step % display_step == 0:
        print("Iter " + str(step) + ", Minibatch Training  Loss= " + \
            "{:.6f}".format(loss) + ", Training Accuracy= " + \
            "{:.5f}".format(acc) + ", Minibatch Validation  Loss= " + \
            "{:.6f}".format(loss_vali) + ", Validation Accuracy= " + \
            "{:.5f}".format(acc_vali) )
    step += 1
```

이 과정은 텐서 플로우에 학습을 시키는 과정이다.

제 4 장

결과 보고서

4.1 결과 보고서
4.2 결과 보고서 발표자료

4.1.1 목표

본 프로젝트에서는 현재 사용되고 있는 카메라나 빛을 이용한 센서의 대안책을 제시하고자 한다. 그 이유는 현재 사용되고 있는 기술에서는 여러 가지 문제점이 발생될 수 있기 때문이다. 예를 들어 카메라 설치 위치에 따라 생겨날 수 있는 사각지대가 있다. 이러한 문제점을 극복하고자 WiFi CSI 기술을 이용하여 사람의 움직임을 감지하고 그것을 그래프로 변환¨ 그래프를 바탕으로 텐서플로우를 활용한 머신러닝 기술을 이용하여 학습을 시키고 학습된 데이터를 이용하여 사람의 움직임이 들어왔을 때 어떤 움직임에 해당하는지 알아내는 것을 목표로 한다. 사람의 움직임을 와이파이를 통해서 파악하는 기술이 더 발전이 되면 홈 IoT 에 까지 적용이 될 수도 있을 것이고 그것을 위하여 따로 장비를 설치하지 않고 이미 설치되어 있는 와이파이를 통해서 기능을 제어할 수 있다면 비용절감이나 사생활 관리 편의성 향상에도 많은 도움이 될 것이다.

4.1.2 연구/개발 내용 및 결과물

4.1.2.1 연구/개발 내용

A. CSI 검출 단계

전파를 전송할 Injection PC와 전파를 받는 Receiver PC에 CSI를 측정하기 위해 Intel WiFi 5300 wireless NIC를 설치한다. NIC의 3개의 안테나를 이용하여 1kHz의 패킷을 WiFi 신호의 형태로 전송한다. Receiver의 3개의 안테나로 서로 다른 1kHz의 패킷을 받는다. CSI 펌웨어는 input을 받아, TimeStamp, Ntx, SNR, Subcarrier 등의 CSI 정보들을 데이터 파일로 만들어 준다.

B. Amplitude 분리/가공 단계

CSI 펌웨어로부터 받은 CSI 데이터 파일에는 여러 정보들이 담겨 있는데, 여기서 30개의 부반송파만을 추출한다. 3개의 안테나를 통하여 만들어진 30개의 벡터의 부반송파를 추출하여 90개의 CSI에서 [n x 90] 형태의 데이터를 얻는다. 이 프로젝트에서는 동작의 구분을 위해 Amplitude만을 사용하기 때문에 Matlab의 유틸리티 코드를 사용하여 [n x 90]개의 CSI 데이터를 절대값을 취함으로써 Amplitude로 변환한다. 만들어진 90개의 Amplitude들은 csv 파일로 저장되어 input File이 된다..

C. 기계학습 단계

최종적으로 가공되어 각 동작에 해당하는 [90 x 1000] 형태의 배열들의 데이터를 기반으로 딥러닝 방법 중 하나인 RNN(순환신경망)과, LSTM(Long-Short term Memory) 기법을 사용하여 학습하는데, 이를 위해 Google의 Tensorflow를 사용한다. RNN은 200개의 Hidden Layer와 Cell을 사용하여 총 2000번 반복하는 학습을 10번 반복하여 오차를 최소화하였다. 이러한 학습은 데이터셋이 많을수록 소요되는 시간이 급격히 늘어나기 때문에, 기계학습 시에 Nvidia의 CUDA[1]를 이용한 GPGPU[2]를 지원하는 TensorFlow-GPU를 이용하여 학습 과정의 시간 소모를 1/20 이하로 단축

[1] CUDA : Nvidia에서 지원하는, GPU의 병렬처리를 이용할 수 있게 하는 도구
[2] GPGPU: General-Purpose computing on Graphics Processing Units

시켰다.

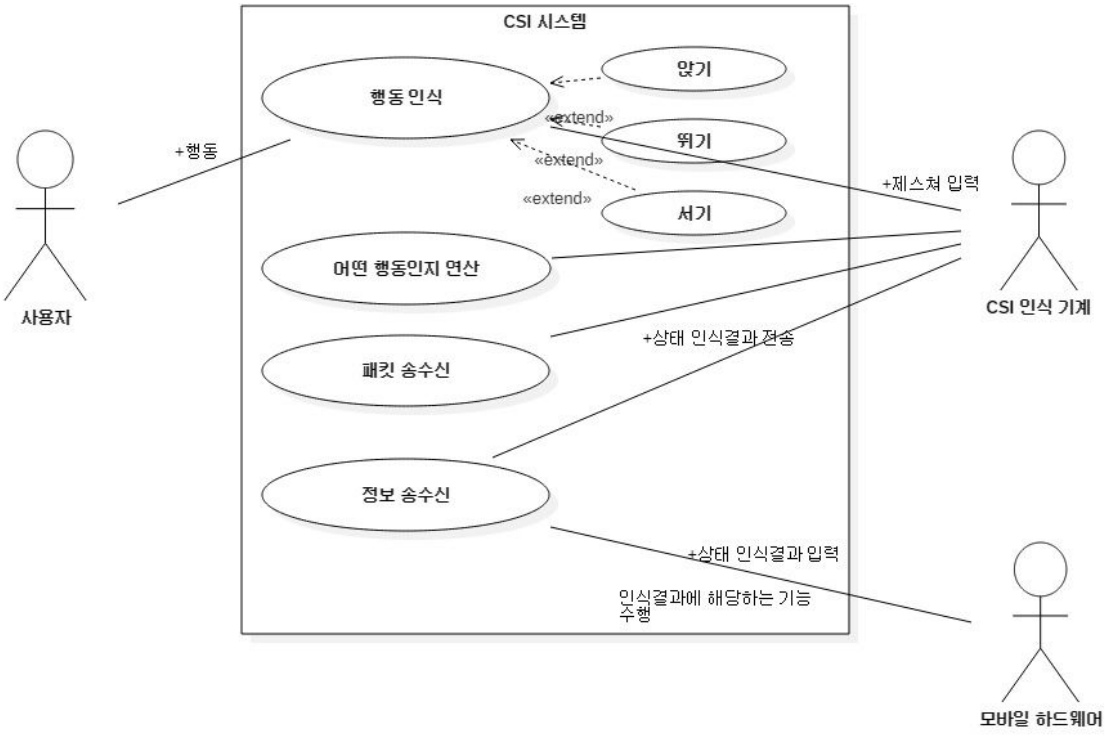
D. 예측 단계

이전 단계에서 생성된 모델을 Tensorflow의 체크포인트를 이용하여 로드하여 가중치와 편향, 그래프를 로딩 한다. 패킷을 전송하여 그 데이터를 [90 x 500]의 형태로 바꾸어, 예측을 하는 프로그램을 적용해 어떤 동작을 수행하였는지 예측되는 값을 출력한다.

E. Home IoT(사물인터넷) 적용 단계

이전 단계에서 받은 정보를 이용하여, 여러가지 실생활의 Home IoT에 적용할 수 있으나, 시간 등의 제약으로 간단한 전구 켜기 등의 기능을 수행한다.

4.1.2.2 시스템 기능 요구 사항



예측 정확도 조절 문제

예측의 정확도를 올리려면 더 많은 샘플 데이터를 생성하여 학습 시키면 된다. 그러나 제한된 시간내에 충분한 양의 샘플 데이터를 확보 할 수 없었다.

피실험자가 두 대의 NIC 사이에서 약속된 동작을 취하면 그 것을 인식 할 수 있어야 한다.

- 완료

인식된 동작을 토대로 전구의 스위치를 켤 수 있어야 한다.

- 미완료

4.1.2.3 시스템 비기능(품질) 요구 사항

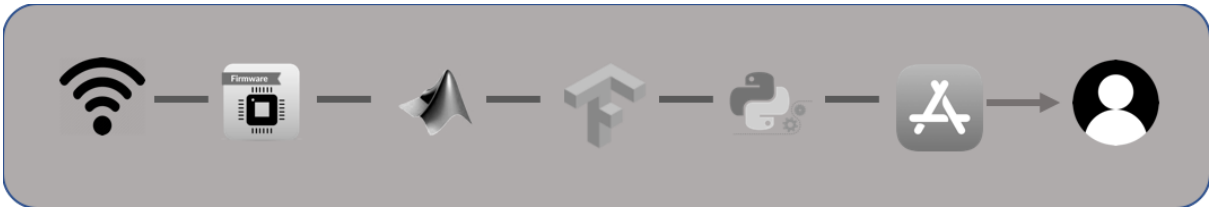
학습 모델 생성과 예측 프로그램의 수행 시간을 줄이기 위해 Nvidia GPU score 3.0이상의 그래픽 카드가 필요하다.

- Nvidia GTX 1070 사용

충분한 WiFi 신호의 생성과 검출을 위해, 송신,수신측 컴퓨터에 장착될 네트워크 카드가 각각 하나씩 필요하다.

- Intel Ultimate WiFi Link 5300 wireless NIC 사용

4.1.2.4 시스템 구조 및 설계도



Access Point -> CSI tool -> Matlab -> Tensorflow(model) -> python -> app -> user

두 개의 컴퓨터 네트워크 포인트 간의 MIMO상황에서의 WiFi CSI의 변화를 수신부 컴퓨터의 Intel 5300 CSI 펌웨어, 툴을 통해 해석을 하고, 이미 만들어 놓은 모델에 입력하여 행동 예측 결과를 출력하는 프로그램을 파이썬으로 만들었다.

4.1.2.5 활용/개발된 기술

WiFi신호를 사용해 행동을 인식하기 위해 Channel State Information(CSI, 채널 상태 정보) 기술을 이용하였다.

CSI 전파의 Amplitude(진폭)변화를 학습하여 행동을 인식하기 위해 RNN(신경회로망)이 사용되었고, RNN을 보조하기위해 LSTM(Long-short term memory)를 적용하였다.

학습과 예측에서 계산되는 행렬 계산을 빠르게 하여 연산속도를 높이기 위해 GPGPU를 이용하였으며, 이를 위해 CUDA와 Tensorflow-GPU가 사용되었다.

4.1.2.5.1 기술적 요구사항

프로젝트의 개발 환경

Wibi는 Linux Ubuntu 14.04(Trusty)버전에서 개발되며 프로젝트 결과 또한 동 버전의 리눅스에서 확인가능하다.

네트워크 펌웨어의 작성 및 사용에는 C, C++을 사용하며, 해당 언어에 사용하는 컴파일러는 g++ 7.2.0을 사용한다.

WiBi에서 사용하는 WiFi 신호를 이용한 인물 및 제스처 인식을 위해서 사용하는 기계학습을 구현하기 위해 파이썬 3.6버전과 Tensorflow를 이용한다.

WiBi에서 사용되는 와이파이 신호 가시화와 기계 학습을 위한 그래프의 사용을 위해 Matplotlib 라이브러리를 사용한다.

Home IOT를 표현하기 위한 견본으로 Lego Mindstorm EV3을 이용한 스위치를 제작하였다.

4.1.2.5.2 프로젝트의 결과물 확인 환경

프로젝트의 결과물은 Linux Ubuntu 14.04(Trusty)버전의 운영체제에서 동작한다.

4.1.2.6 현실적 제한 요소 및 그 해결 방안

장애물,사람 등 외부요인, 통제되지 않은 환경 외에서의 현실적 한계
인식률을 위한 알고리즘(찾아볼것).

4.1.2.6.1 하드웨어

A. 기능을 IoT로 확장하기 위해, 모바일 애플리케이션을 개발할 경우, 현재 인물 및 제스처 인식을 위해 사용중인 NIC로 인해 다른 WIFI에 연결할 수 없어서 어플리케이션과 통신할 수 없다. 펌웨어를 개조하거나 네트워크가 아닌 다른 경로를 통하여 어플리케이션과 통신을 하여야 할 것 같다.

B. Intel Ultimate N 5300 NIC는 인텔 측에서 정식으로 펌웨어 및 관리 소프트웨어를 제공하지 않으므로, 개인이 제작한 오픈소스 펌웨어를 사용하여야 한다. 이는 상용화 프로그램 제작이나 대형 프로젝트를 제작 할 때, NIC 제작사인 인텔에서 적절한 안정성이나 피드백을 보장할 수 없다는 뜻이다.

C. 수신받은 패킷을 이용하여 제스쳐를 인식할 때, 어떤 제스쳐인지 구별하기 위해 상당히 많은 양의 패킷을 필요로 한다. 패킷의 양이 일반 데스크탑의 성능 처리 속도로는 짧은 시간 내에 처리할 수 없는 양이므로 실시간으로 제스쳐를 인식해야 하는 IOT 환경을 구현하는 데에 문제가 생길 수 있다.

4.1.2.6.2 소프트웨어

A. IoT 환경을 위해 애플리케이션을 제작하려면, 안드로이드 개발 환경 특성상 Java를 사용해서 구현하여야 한다. Wibi 프로젝트에서는 기계 학습 구현을 위해 Python과 Tensorflow를, Intel 5300 NIC를 이용한 와이파이 신호 생성 및 관리를 위해 C와 C++을 이용하므로 이를 애플리케이션과 연동할 때 언어간의 접합과 관리가 어려울 수 있다.

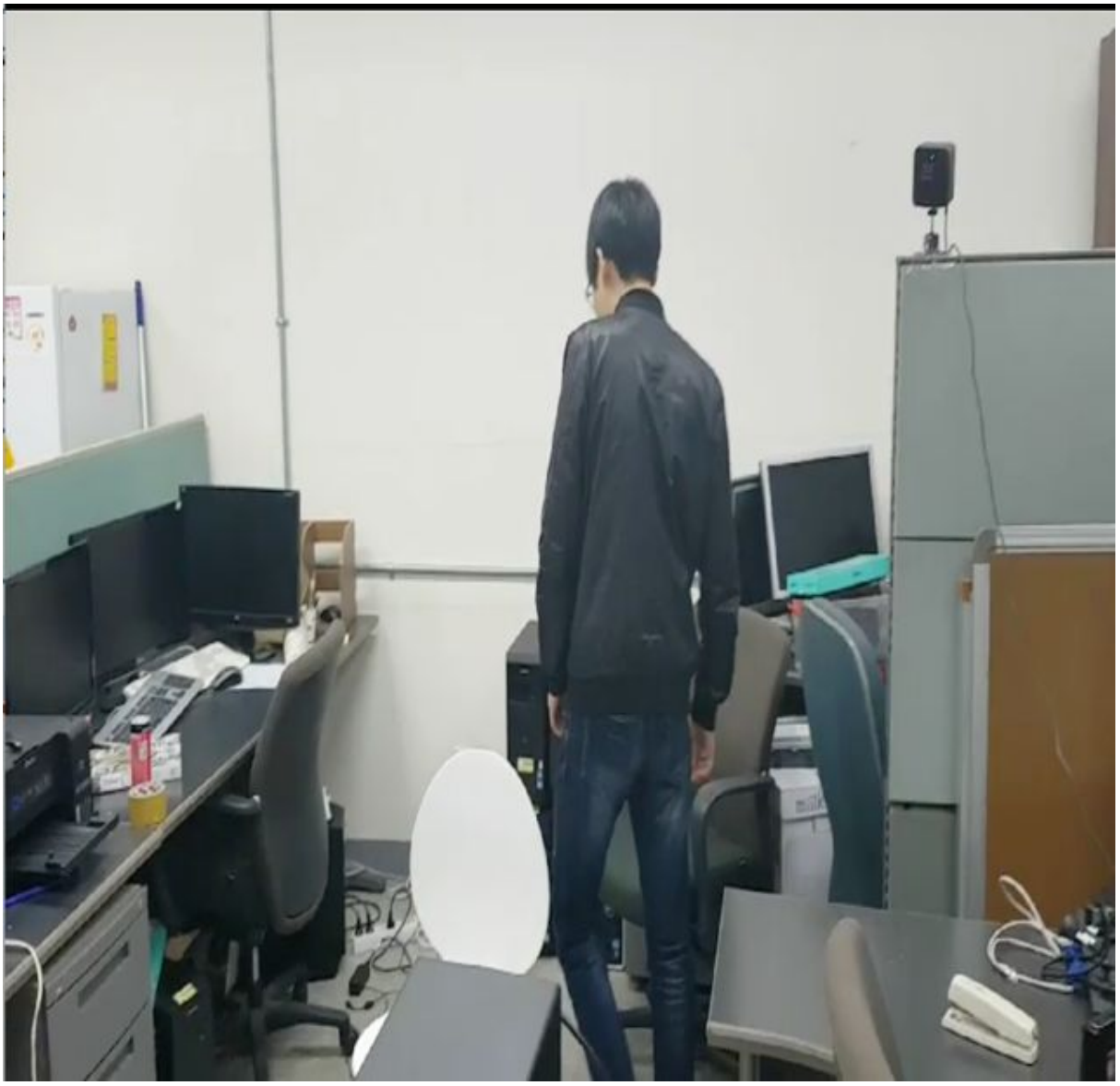
4.1.2.6.3 기타

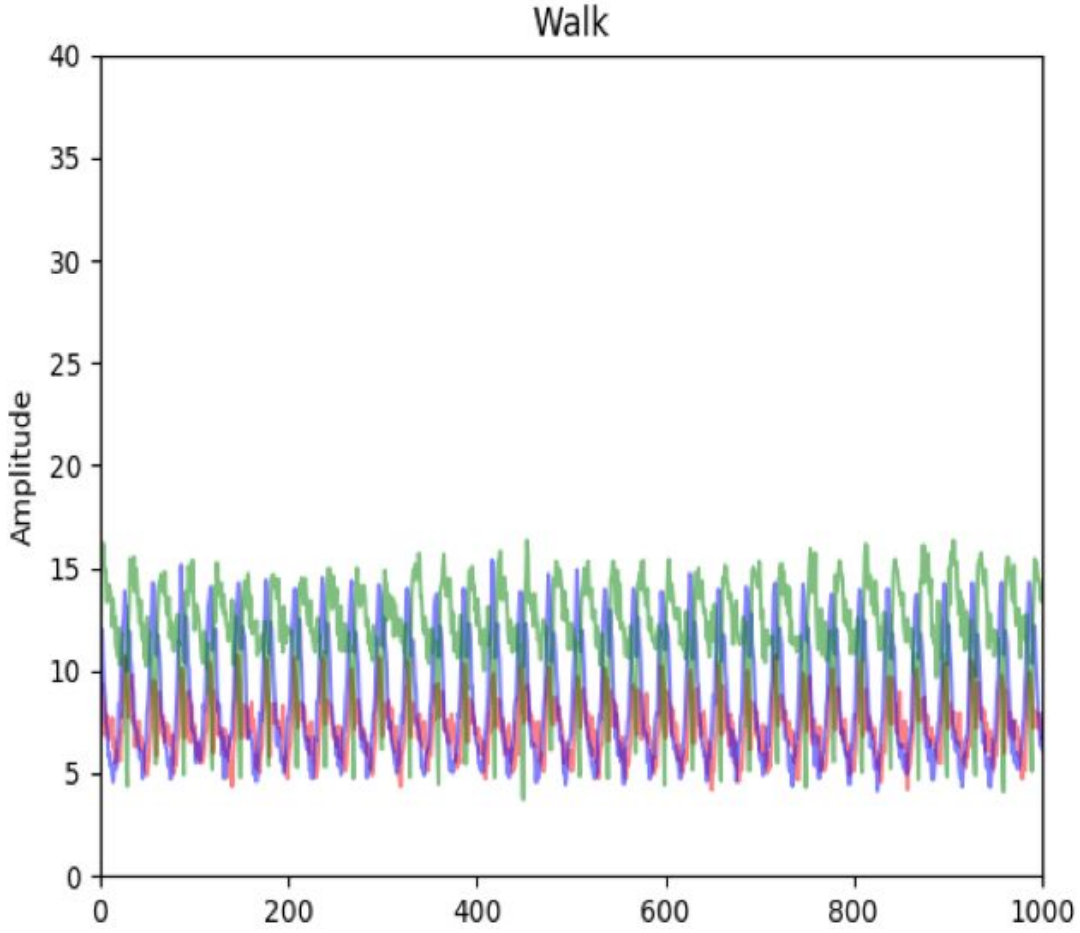
A. 전파로 송출되는 WIFI 특성상 다른 곳에서 송출되는 WIFI나 송/수신 장치 사이의 장애물 등 여러가지 방해 요소로 인해 일정하지 않은 값을 얻게 될 수 있다.

B. 매우 통제된 환경에서 만들어진 데이터를 토대로 학습 모델을 만든다. 통제된 환경은 데이터 측정시 1명만 있는 공간을 말한다. 하지만 실제 시연환경은 주변에 사람이 많다. 따라서 기존에 학습된 모델로는 일정한 예측 결과를 얻기 힘들 수 있다.

4.1.2.6.4 결과물 목록

전체모습, 실행중인 프로그램 사진, 실제예측상황, 제스쳐





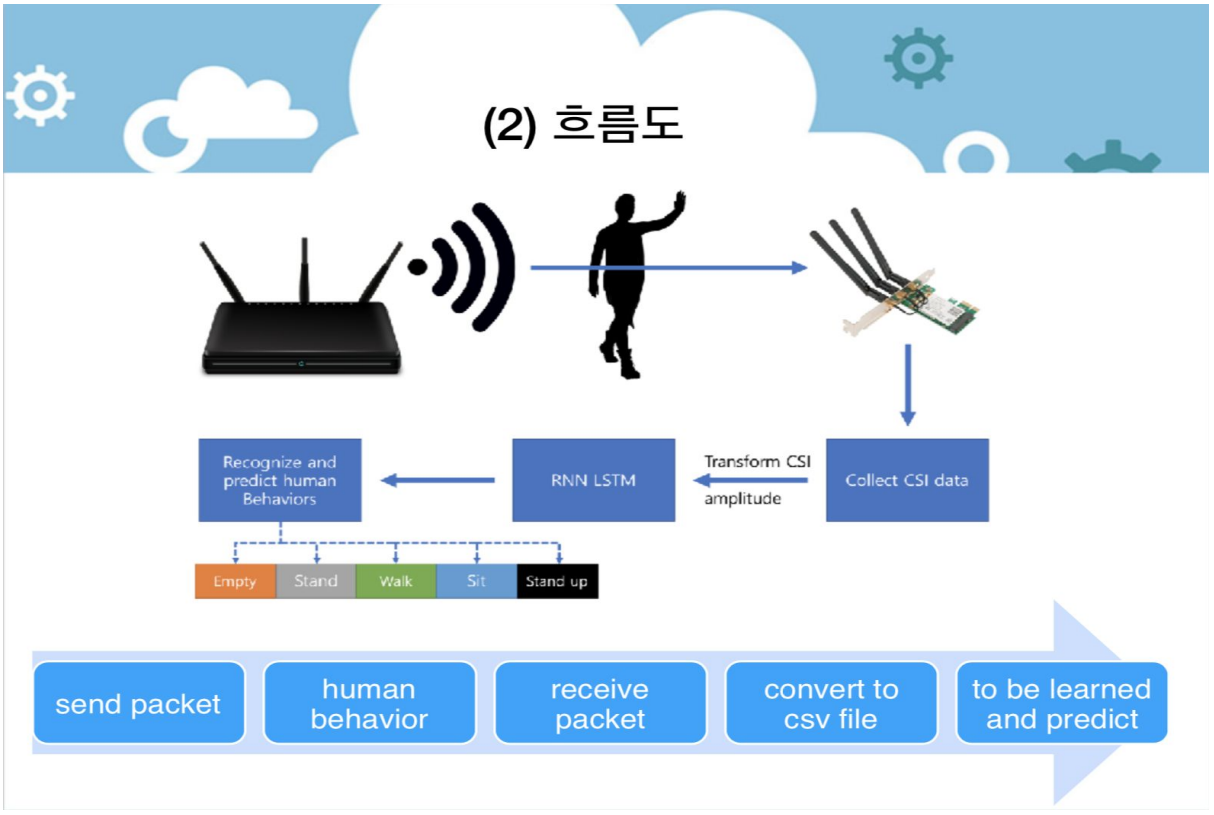
4.1.2.6.5 기대효과 및 활용 방안

기존의 상용된 제스쳐/인물 인식 방법은 카메라나 레이저 센서를 사용하였기에 인식 범위에 한계가 있었고, 사각지대가 존재한다는 문제점이나 낮은 조도에서 사용이 제한된다는 문제 등 여러 가지 제한사항이 존재하였다. 그러나 CSI기술을 이용한 와이파이 신호를 사용한다면 전파를 이용하기 때문에 장애물, 사각지대, 사생활 보호에 카메라에 비해 상당한 우위를 지니게 된다.. 그렇기 때문에 사생활이 중요시될 만한 가정집 등의 장소에서 폭넓게 사용될 수 있다. 또한 WiFi 신호를 이용한 CSI 기술은 사람의 호흡까지 감지할 수 있을 정도로 예민하게 측정할 수 있고, 카메라로 보이지 않는 곳도 빈틈없이 동작을 인식 할 수 있기 때문에, 중요 보안 시설이나 제한구역 등에서도 카메라를 보조하거나 대체할 수 있을 것으로 기대된다.

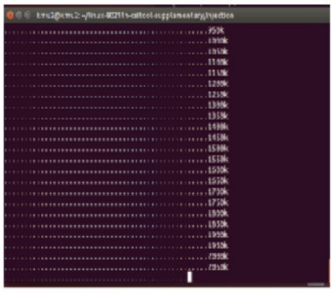
이러한 장점으로 인해 WiBi는 IoT 기술 분야에서 여러 가지 장점을 지닐 수 있는데, WiFi 신호를 이용하면 제스처를 인식 할 수 있기에 기존의 IoT에 사용되던 말을 통한 명령이나 스마트폰 애플리케이션을 통한 명령에 제스쳐를 통한 명령이라는 새로운 방법을 도입할 수 있게 된다. 그로 인해 기존 IoT 기기들의 편의성이 더욱 늘어날 것이며, 주 기능 이외에도 외출 시에 측정되는

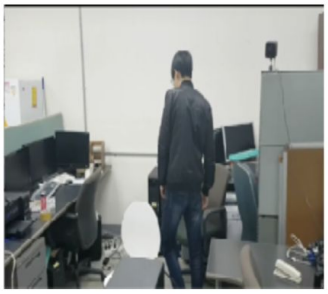
CSI 정보를 이용한 보안 등 WiFi를 이용한 여러 가지 부가적 활용이 가능할 것으로 예상된다.

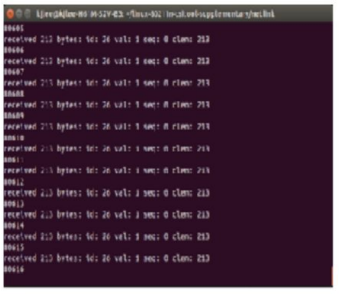
4.2. 결과 보고서 발표자료



(1) 데이터 셋 생성







send packet → human behavior → Receive packet

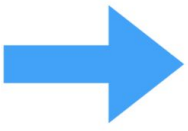
(2) 학습 단계

```
# Keep training until reach max iterations
while step < training_iters:
    batch_x, batch_y = wifi_train.next_batch(batch_size) #wifi_train에 저장된 x와 y를 배
    x_vali = wifi_validation.images[:]
    y_vali = wifi_validation.labels[:]
    # Reshape data to get 28 seq of 28 elements
    batch_x = batch_x.reshape((batch_size, n_steps, n_input)) #batch_x를 500행 90열 ba
    x_vali = x_vali.reshape([-1, n_steps, n_input]) #x_vali은 500행, 90열로 만들고 난뒤에
    # Run optimization op (backprop)
    sess.run(optimizer, feed_dict={x: batch_x, y: batch_y})

    # Calculate batch accuracy
    acc = sess.run(accuracy, feed_dict={x: batch_x, y: batch_y})
    #x에 batch_x, y에 batch_y를 넣어야며 accuracy 실행해야 리스값을 acc에 저장
    acc_vali = sess.run(accuracy, feed_dict={x: x_vali, y: y_vali})
    # Calculate batch loss
    loss = sess.run(cost, feed_dict={x: batch_x, y: batch_y})
    loss_vali = sess.run(cost, feed_dict={x: x_vali, y: y_vali})

    # Store the accuracy and loss
    train_acc.append(acc)
    train_loss.append(loss)
    validation_acc.append(acc_vali)
    validation_loss.append(loss_vali)

    if step % display_step == 0:
        print("Iter " + str(step) + ", Minibatch Training  Loss= " + \
            "{:.6f}".format(loss) + ", Training Accuracy= " + \
            "{:.5f}".format(acc) + ", Minibatch Validation  Loss= " + \
            "{:.6f}".format(loss_vali) + ", Validation Accuracy= " + \
            "{:.5f}".format(acc_vali) )
    step += 1
```



model.ckpt.data-00000-of-00001

model.ckpt.index

model.ckpt.meta

제 5 장

프로젝트 수행
자체 평가

이 프로젝트는 김상철 교수님께 아이디어를 받아 지난 겨울 방학부터 준비한 것으로 본래 Home IoT를 구상 하던중 기존의 카메라와 같은 센서를 대체할 새로운 센서의 개념을 실증해본 것이다. 기존의 Home IoT의 센서의 한계인 LoS(Line of Sight), 사생활에 취약함 등을 극복 할 수 있는 기술 이라고 생각된다. 또한 기존에 NIC, Router를 인터넷에 연결 하는데에만 사용했다. 즉 그 외에 라우터, NIC에서 나오는 패킷은 쓸모 없이 버려지는 것이다. 이러한 버려지는 패킷을 사용하여 카메라가 불가능한 영역을 충족 시켜줄 수 있다는 점에서 기존의 카메라 센서를 보완 할 수 있는 새로운 센서 기술이라고 생각된다.

주 의 문

2018년 캡스톤 디자인 I 종합설계 프로젝트 최종보고서
WiBi

발행인 : **이경재, 양재영, 김태기, 박수찬**
발행일 : 2018년 5월 29일
발행처 : 국민대학교 전자정보통신대학 컴퓨터공학부
주소 : (136-702) 서울시 성북구 정릉동 861-1
