mmwave 센서를 사용한
위험행동 및 일상행동 인식 장치

소속: 정보컴퓨터공학부

팀명: 뜨거운 아아

구성원: 강수빈, 박준형, 이영인

목차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1.1 요구조건

노인분들의 안전을 보장하기 위해 mmwave를 이용하여 낙상과 같은 위험행동을 인지하고 일상행동과 구분하여 위험행동이 발생했을 때 알림을 울리는 시스템을 개발하는 것이 목표이다.

- 비접촉방식으로 사용자의 낙상과 같은 위험행동과 일상행동을 알 수 있는 정보수집이 요구된다. .
- 사용자의 다양한 행동 데이터가 수집된 후에는 이 수집된 데이터를 사용하여 일상행동과 위험행동을 구분하는 모델학습을 진행한다. 이후 이렇게 학습된 모델을 사용하여 실시간으로 측정된 사용자의 행동 데이터를 위험행동 또는 일상행동으로 분류할 수 있어야 한다.
- 위험행동이 인지되었을 때 위험상황임을 알리기 위해 실시간 모니터링 어플리케이션 제작이 필요하다.

1.2 추가 제약사항 및 수정사항

기존의 제약사항과 비교하였을 때 수정 또는 추가된 제약사항으로는 아래와 같다.

- mmwave센서가 정적인 동작의 경우 움직임을 잘 감지하지 못할 수 있다.

  ⇒ 정적인 동작 뿐만 아니라 격한 동작의 경우에도 사용자의 움직임에 대한 인식이 잘 이루어지지 않을 수 있는 제약사항이 발생하였으나 적절한 거리와 화면 각도를 사용하여 데이터를 수집하면 동적인 동작들도 잘 인식될 것으로 예측된다. 따라서 데이터를 수집하기에 가장 적절한 거리와 화면 각도를 찾는 것이 필요하다.

- 데이터 수집에 부족한 부분은 오픈 데이터를 사용하려고 하였으나 이 프로젝트에 사용할만한 적절한 데이터를 찾기가 쉽지 않다.

  ⇒ 현재 프로젝트에 사용할만한 적절한 오픈 데이터에 대한 조사가 더욱 필요하며, 동시에 데이터 수집 부분에서는 부족한 부분이 없도록 시간을 투자하여 직접 필요한 만큼 많은 데이터를 수집해야 한다.

2. 설계 상세화 및 변경 내역

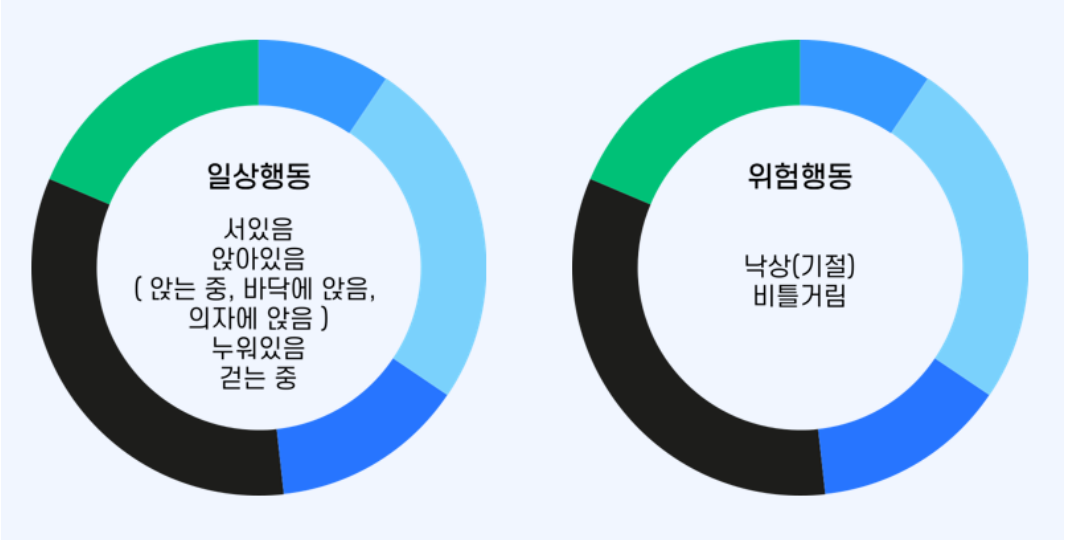
2.1 데이터 수집 장치

이 프로젝트를 계획할 때에는 mmwave센서를 사용하여 사용자의 움직임에 대한 데이터를 수집할 생각이었다. 하지만 depth, sekleton 등의 행동구분을 위한 구체적인 데이터를 얻기 위해 kinect 카메라 또한 사용하기로 하였다.

- IWR6843ISK + DCA1000EVM 보드를 이용하여 데이터 수집을 위한 환경 구성
- Kinect 카메라를 사용하여 데이터 수집을 위한 환경 구성

2.2 데이터 수집

단순히 위험행동(낙상)과 일상행동으로 구분하여 데이터를 수집하는 것이 아닌 위험행동과 일상행동을 아래 이미지와 같이 세분화하여 행동 데이터를 수집한다.



- Pykinect로 원하는 동작을 인식하고 해당 동작에 대해 스켈레톤 형식으로 수집
- mmwave센서로 동작에 대한 정보를 cloud point와 raw data형식으로 수집

2.3 데이터 분석

행동 구분 모델학습을 단순히 두가지 종류로 구분하는 것이 아닌 낙상, 비틀거림, 앉아있음, 서있음, 누워있음과 같이 세분화된 상태로 분류하도록 학습시켜야 한다.

	Tag_ID	x	y	z	activity
0	010-000-024-033	4.062931	1.892434	0.507425	walking
1	020-000-033-111	4.291954	1.781140	1.344495	walking
2	020-000-032-221	4.359101	1.826456	0.968821	walking
3	010-000-024-033	4.087835	1.879999	0.466983	walking
4	010-000-030-096	4.324462	2.072460	0.488065	walking
...	...	...	...	...	...
164855	010-000-030-096	3.209474	2.044571	0.062902	walking
164856	010-000-024-033	3.386878	2.004729	0.395161	walking
164857	020-000-033-111	3.188895	1.915717	1.353087	walking
164858	010-000-030-096	3.150169	1.931164	0.055037	walking
164859	010-000-024-033	3.209994	1.939577	0.364777	walking

데이터수집 전 오픈데이터를 활용하여 다양한 행동을 구분하는 모델 학습을 간단히 진행해본 결과 정확도가 40% 또는 70%정도로 낮게 나왔다.

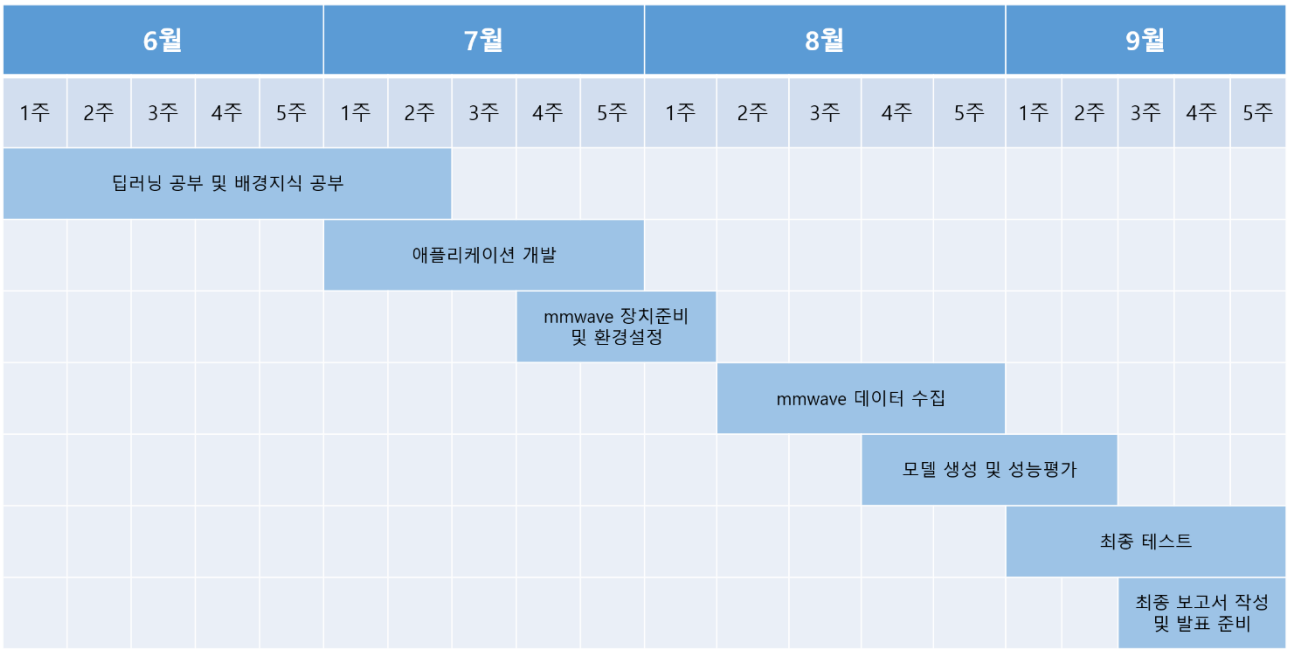
따라서 이전에 사용하려고 했던 Decision tree model뿐만 아니라 Gradient Boosting Classifier model, RandomForest model, SVM model 그리고 DNN model 등의 다양한 모델과 하이파라미터들을 조정하며 학습에 가장 적절한 model을 찾는 것이 필요하다.

2.4 모바일 애플리케이션

- 낙상 감지 시 휴대폰 앱 푸쉬 알림으로 위험행동 감지를 알림
- 낙상 감지 시 앱에 등록된 번호로 위험행동 발생 알림 메시지 전송
- 낙상이 발생한 시간을 앱으로 확인가능

3. 갱신된 과제 추진 계획

기존에 계획한 일정으로는 데이터 수집 후 모델 학습 및 앱 개발을 진행하려고 하였으나 데이터 수집을 위한 장치 준비 및 환경설정으로 인해 아직까지 데이터 수집이 제대로 진행되지 못하였다. 따라서 기존 일정에서 앱 개발 및 오픈데이터를 사용한 모델 학습 공부를 우선으로 하는 일정을 진행하였다. 현재 데이터 수집을 위한 장치 준비 및 환경설정이 완료되었고 다음주부터 바로 데이터 수집을 시작하여 최대한의 정확도를 이끌어 내는 모델을 생성하는 것이 목표이다. 수정된 추진체계 및 일정은 아래와 같다.



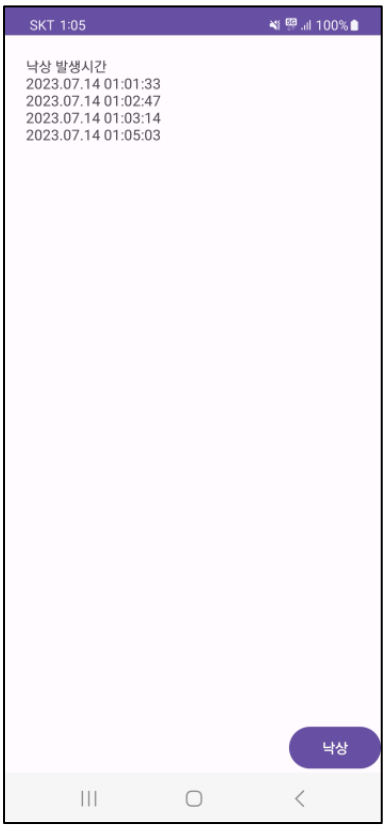
4. 구성원별 진척도

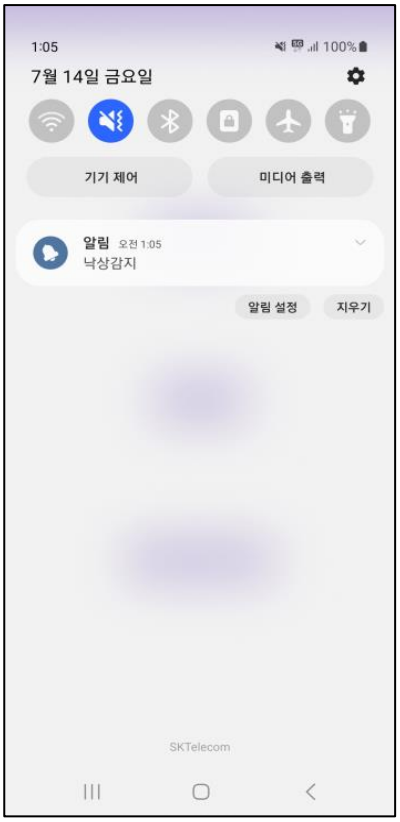
이름	진척도
강수빈	- Mmwave 장치 준비 및 환경설정 - 행동인식 머신러닝 학습 (오픈 데이터 사용)
박준형	- Kinect 장치 준비 및 환경설정 - 알림 어플리케이션 개발
이영인	- Mmwave 장치 준비 및 환경설정 - 행동인식 머신러닝 학습 (오픈 데이터 사용)

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

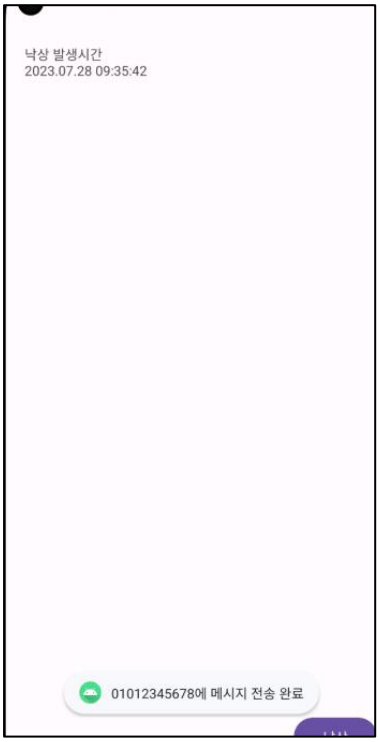
5.1 애플리케이션 개발

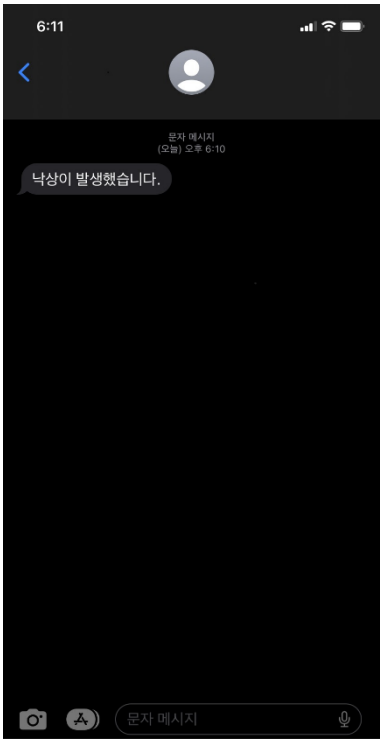
- 낙상 발생시 시간 기록 및 푸시 알림 발생





- 등록된 전화번호로 낙상 메시지 전송





5.2 오픈데이터를 사용한 모델학습 연습

가슴, 허리, 양쪽 발목에 대한 x, y, z 좌표 정보를 가지고 있는 오픈 데이터 셋으로 걷기, 낙상, 눕기, 앉기, 일어나기 등 행동을 구분하는 모델학습을 진행하였다. 각 Tag_ID는 가슴, 허리, 오른쪽 발목, 왼쪽 발목을 구분하는 ID이다.

	Tag_ID	x	y	z	activity
0	010-000-024-033	4.062931	1.892434	0.507425	walking
1	020-000-033-111	4.291954	1.781140	1.344495	walking
2	020-000-032-221	4.359101	1.826456	0.968821	walking
3	010-000-024-033	4.087835	1.879999	0.466983	walking
4	010-000-030-096	4.324462	2.072460	0.488065	walking
...	...	...	...	...	...
164855	010-000-030-096	3.209474	2.044571	0.062902	walking
164856	010-000-024-033	3.386878	2.004729	0.395161	walking
164857	020-000-033-111	3.188895	1.915717	1.353087	walking
164858	010-000-030-096	3.150169	1.931164	0.055037	walking
164859	010-000-024-033	3.209994	1.939577	0.364777	walking

- 가슴에 대한 x, y, z좌표를 가지고 Gradient Boosting Classifier와 DNN model 학습을 진행한 경우

```
[7]  # GradientBoostingClassifier model
     from sklearn.ensemble import GradientBoostingClassifier
     from sklearn.metrics import accuracy_score

     # Gradient Boosting 모델 초기화
     gb_model = GradientBoostingClassifier()

     # 모델 학습
     gb_model.fit(X_train2, Y_train2)

     # 검증 데이터에 대한 예측
     val_predictions = gb_model.predict(X_val2)

     # 검증 데이터에 대한 정확도 평가
     val_accuracy = accuracy_score(Y_val2, val_predictions)
     print("Validation Accuracy:", val_accuracy)

     # 테스트 데이터에 대한 예측
     test_predictions = gb_model.predict(X_test2)

     # 테스트 데이터에 대한 정확도 평가
     test_accuracy = accuracy_score(Y_test2, test_predictions)
     print("Test Accuracy:", test_accuracy)

     Validation Accuracy: 0.7412739965095986
     Test Accuracy: 0.7385689354275742
```

[Gradient Boosting Classifier model]

```
[13]  1 # test set을 사용하여 모델 평가
      2 loss, categorical_accuracy = model.evaluate(X_test2, tf.keras.utils.to_categorical(Y_test2, num_classes=11), verbose=0)
      3 print("loss: ",loss)
      4 print("categorical_accuracy:",categorical_accuracy)
      5
      6 # 모델 결과(구조) 출력
      7 model.summary()
```

```
loss:  0.8625609278678894
categorical_accuracy: 0.722745761764526
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                256

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 batch_normalization_1 (Batc  (None, 32)               128
 hNormalization)

 dense_2 (Dense)             (None, 32)                1056

 batch_normalization_2 (Batc  (None, 32)               128
 hNormalization)

 dropout_1 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 16)                528

 batch_normalization_3 (Batc  (None, 16)               64
 hNormalization)

 dense_4 (Dense)             (None, 11)                187

=================================================================
Total params: 4,683
Trainable params: 4,395
Non-trainable params: 288
_________________________________________________________________
```

[DNN model]

- 약 70%의 정확도를 가진다.

- 발목에 대한 x, y, z좌표를 가지고 Gradient Boosting Classifier와 DNN model 학습을 진행한 경우

```python
# GradientBoostingClassifier model
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

# Gradient Boosting 모델 초기화
gb_model = GradientBoostingClassifier()

# 모델 학습
gb_model.fit(X_train2, Y_train2)

# 검증 데이터에 대한 예측
val_predictions = gb_model.predict(X_val2)

# 검증 데이터에 대한 정확도 평가
val_accuracy = accuracy_score(Y_val2, val_predictions)
print("Validation Accuracy:", val_accuracy)

# 테스트 데이터에 대한 예측
test_predictions = gb_model.predict(X_test2)

# 테스트 데이터에 대한 정확도 평가
test_accuracy = accuracy_score(Y_test2, test_predictions)
print("Test Accuracy:", test_accuracy)
```

```
Validation Accuracy: 0.5515508328546812
Test Accuracy: 0.545485871812543
```

[Gradient Boosting Classifier model]

```
# test set을 사용하여 모델 평가
loss, categorical_accuracy = model.evaluate(X_test2, tf.keras.utils.to_categorical(Y_test2, num_classes=11), verbose=0)
print("loss: ",loss)
print("categorical_accuracy:",categorical_accuracy)

# 모델 결과(구조) 출력
model.summary()
```

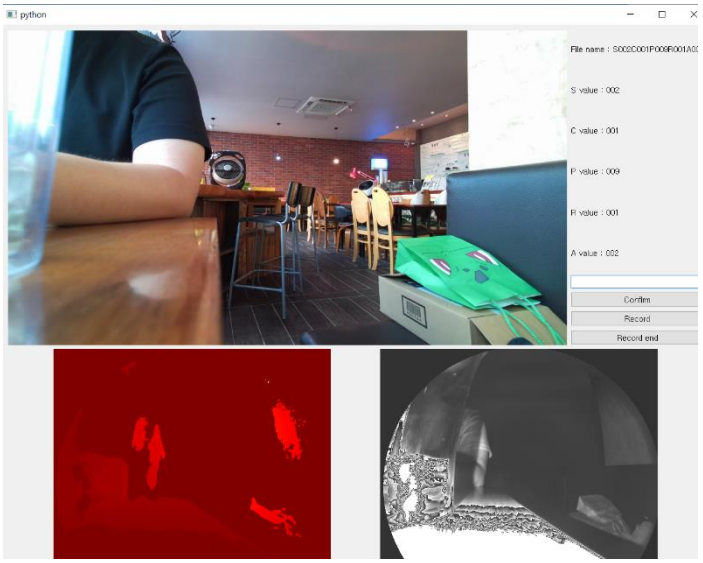
```
loss:  1.3682559728622437
categorical_accuracy: 0.5176889300346375
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 64)                256

batch_normalization_8 (Batc  (None, 64)                256
hNormalization)

dropout_4 (Dropout)          (None, 64)                0

dense_11 (Dense)             (None, 32)                2080

batch_normalization_9 (Batc  (None, 32)                128
hNormalization)

dense_12 (Dense)             (None, 32)                1056

batch_normalization_10 (Bat  (None, 32)                128
chNormalization)

dropout_5 (Dropout)          (None, 32)                0

dense_13 (Dense)             (None, 16)                528

batch_normalization_11 (Bat  (None, 16)                64
chNormalization)

dense_14 (Dense)             (None, 11)                187

=================================================================
Total params: 4,683
Trainable params: 4,395
Non-trainable params: 288
_________________________________________________________________
```

[DNN model]

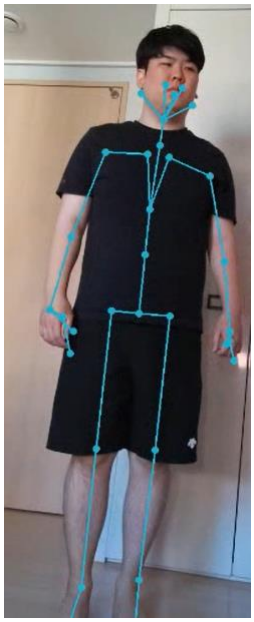
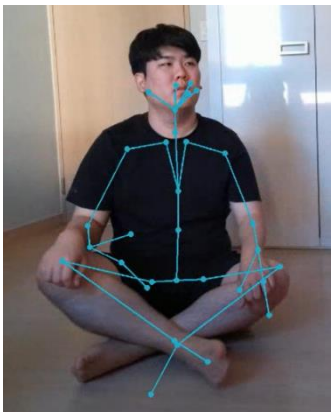
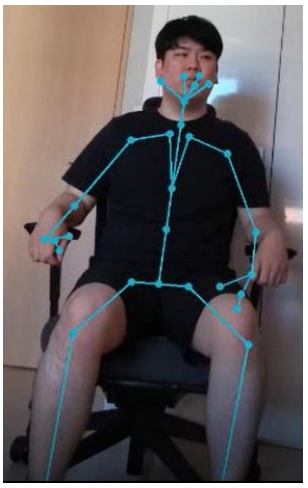
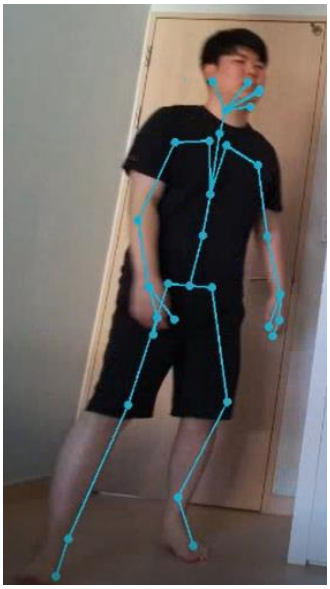
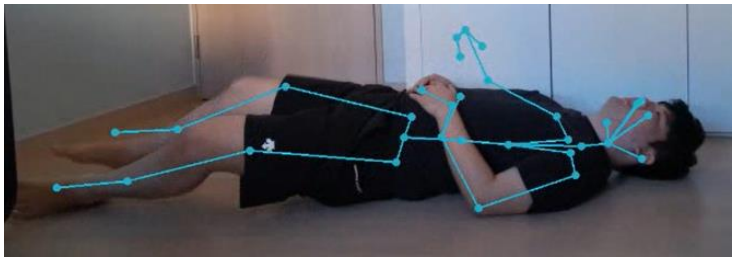
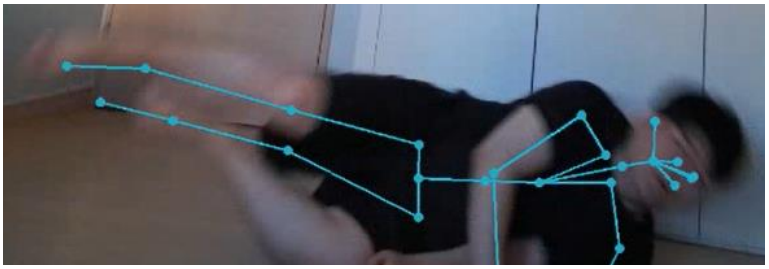
- x, y, z좌표의 변화량이 가슴에 비해 크지않아 model의 정확도도 가슴보다 낮은 약 50%이다.

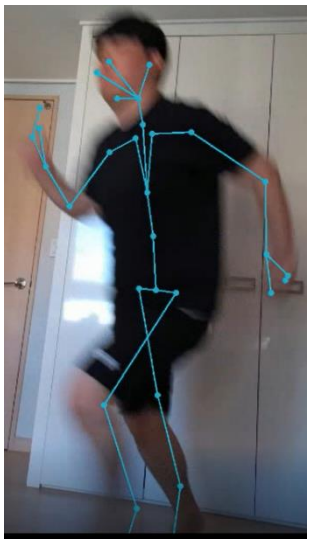
5.3 mmwave 센서와 Kinect 카메라의 리눅스 환경과 윈도우 환경 준비

- Kinect 카메라



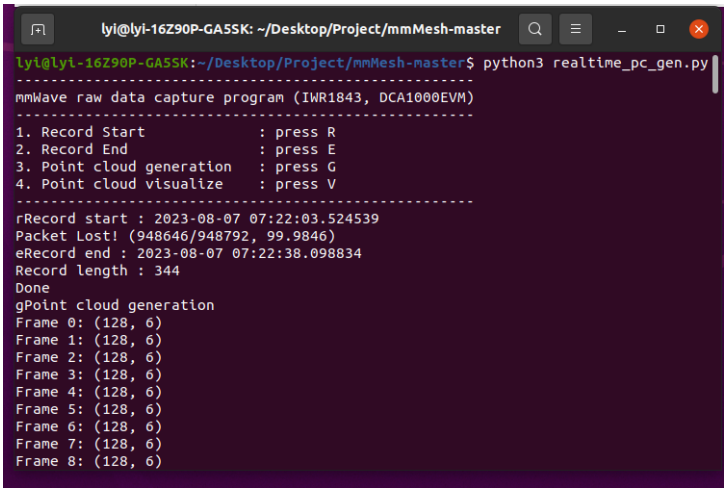
[Kinect 카메라 동작 모습]

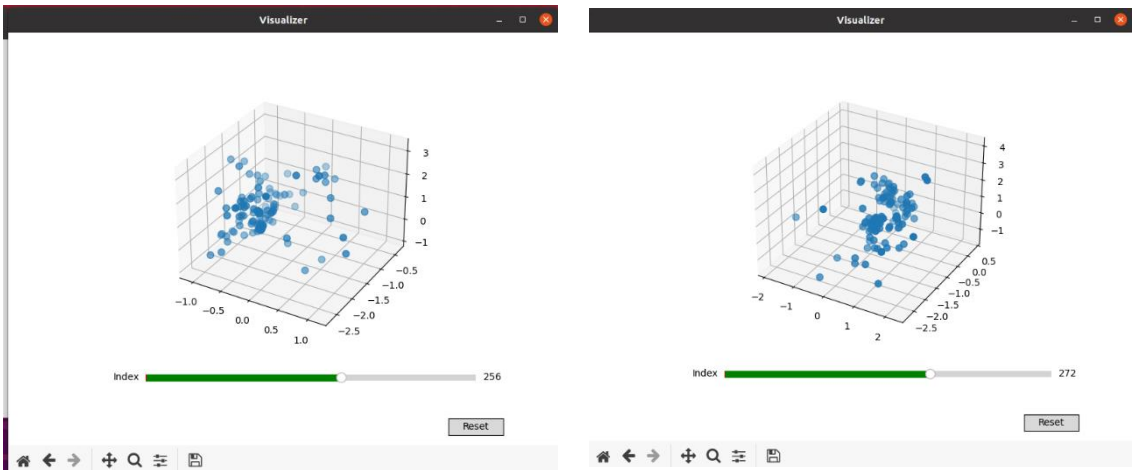
<table>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td>서있음</td><td>바닥에 앉아있음</td><td>의자에 앉아있음</td></tr>
<tr><td rowspan="3">  </td><td colspan="2">  </td></tr>
<tr><td colspan="2">누워있음</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td>비틀거림</td><td colspan="2">낙상</td></tr>
</table>



달리는 중

- mmwave 센서 (IWR6843ISK + DCA1000EVM 보드)





[좌우로 움직였을 때]