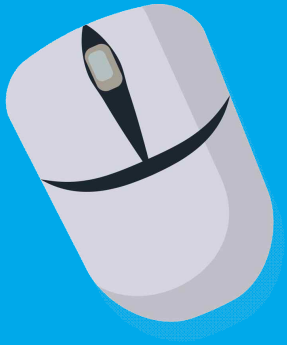
Virtual mouse.

Lee Kyeong Yeong



목차

PyAutoGUI

PART 1.

프로젝트 소개

Virtual Mouse

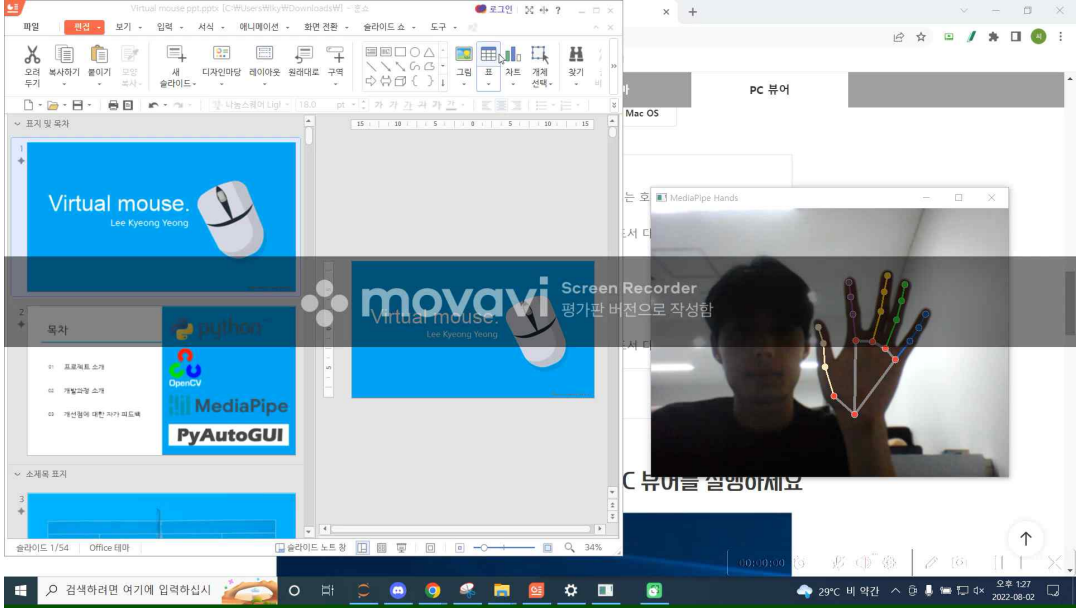
“

야식먹으면서 뭐하세요?

”

PART 2.

개발과정



미디어파이프는 구글에서 주로 인체를 대상으로하는 비전인식기능들을 AI모델 개발과 기계학습까지 마친 상태로 제공하는 서비스이다.

다양한 프로그램언어에서 사용하기 편하게 라이브러리 형태로 모듈화되어 제공되며 사용방법 또한 풍부하게 제공된다.

몇가지 간단한 단계로 미디어파이프에서 제공하는 AI기능을 활용한 응용 프로그램개발이 가능하다.

MediaPipe

MediaPipe Hands는 고성능 손, 손가락 추적 솔루션이다.

머신 러닝(ML)을 사용하여 각 손마마다 21개의 3D 랜드마크를 프레임마다 추론한다.

현재의 최첨단 접근 방식들은 추론을 위해 주로 강력한 하드웨어 성능에 의존하는 반면, MediaPipe의 방법은 휴대폰에서도 실시간 성능을 달성하고 2개 이상의 손을 인식할 수 도 있다.

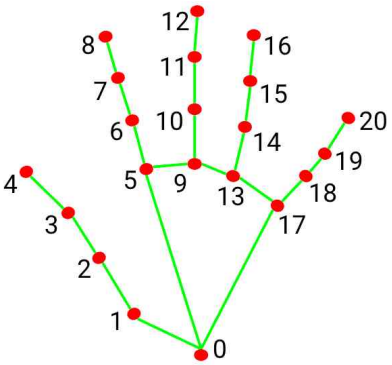
MediaPipe

모델
Palm Detection Model(손바닥 감지 모델)
정확도 95.7 %

실측 자료 데이터를 얻기 위해 아래와 같이 21개의 3D 좌표를 가진 30,000개의 실제 이미지에 수동으로 학습했다(해당 좌표당 Z 값이 존재하는 경우에는 이미지 깊이 맵에서 Z 값을 취한다).

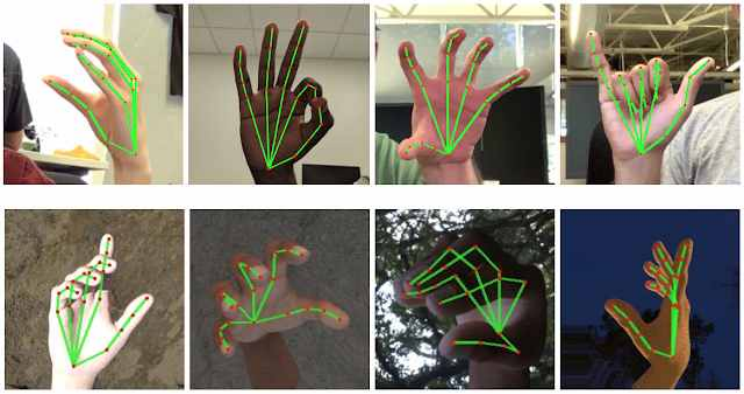
손 포즈를 더 잘 다루고 손 기하학의 특성을 여러분이 사용하기 위해 다양한 배경에 고품질 모델을 렌더링하고 해당 3D 좌표점에 매핑한다.

MediaPipe



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

MediaPipe



기본 동작 원리

```
# 필요에 따라 성능 향상을 위해 이미지 작성을 불가능함으로 기본 설정합니다.
        image.flags.writeable = False
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image_blur = cv2.GaussianBlur(image, (7,7), 0)
        results = hands.process(image_blur)


    # 이미지에 손 주석을 그립니다.
        image_blur.flags.writeable = False
        image_blur = cv2.cvtColor(image_blur, cv2.COLOR_RGB2BGR)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
```

```
landmark {
  x: 0.3118162155151367
  y: 0.890053702705383
  z: 0.0014742190251126885
}
```

0~20번 까지의 번호 별 x, y, z 까지의 좌표 반환

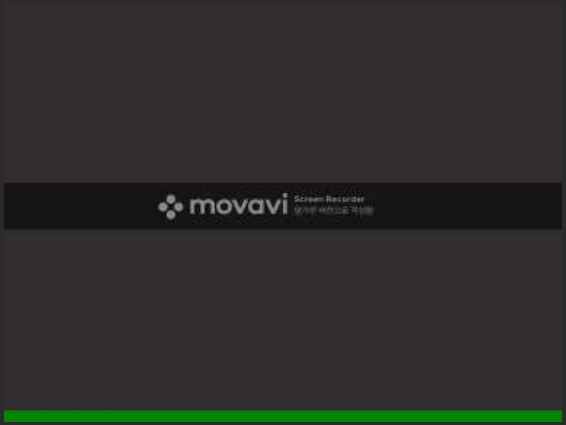
기본 동작 원리

```
#(bx,by -> 이전위치 저장변수) (nx, ny -> 프레임별 마우스 위치 저장변수)
bx, by = 0, 0
nx, ny = 0, 0
moniter_x, moniter_y = pyautogui.size() #사용자 모니터 크기

if bx==0 and by==0:
    pyautogui.moveTo(moniter_x//2, moniter_y//2)

else:
    pyautogui.moveTo(cx+(ax*moniter_x)*(-2), cy+(ay*moniter_y)*2)
```

pyautogui : 마우스, 키보드 제어용 라이브러리

ax, ay : 변화율
bx, by : 이전 커서위치
cx, cy : 현재 커서위치

문제점1 : 떨림문제



Part 2

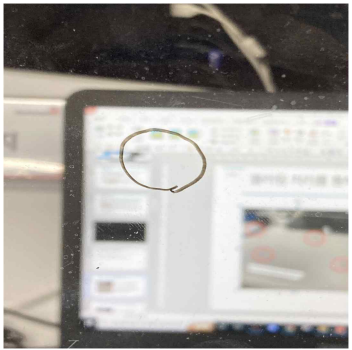
떨림문제 개선을 위한 방안

1. 블러링 처리를 통해 떨림보정 시도

2. N개의 frame별 이동평균을 구하여 떨림보정 시도

3. 정규화된 좌표의 반올림을 통해 떨림보정 시도

블러링 처리를 통해 떨림보정 시도



1. 아크릴판에 동그라미를 그려 놓는다.

2. 동일 거리를 이격하여 10초간 손을 가만히 놓고 openCV에서 제공하는 Gaussian 필터를 사용하여 블러링 처리를 한다.

3. 본실험에서 gaussian 필터의 행렬의 크기는 n=[1, 11, 21, 31] n x n 행렬을 사용하였다.

4. gaussian 필터 크기에 따라 나온 x, y값의 분산을 통해 얼마나 퍼져있는지를 비교해 보았다.

Part 2

블러링 처리를 통해 떨림보정 시도

In [206]:
```
varx_list=[]
vary_list=[]

for i in range(4):
    varx_list.append(sum(gvarx[1+i*10:4+i*10])/3)
    vary_list.append(sum(gvary[1+i*10:4+i*10])/3)
print(varx_list)
print(vary_list)
```

실험은 n값에 따라
1회당 10초씩 총 3회 진행
3회의 평균값을 사용

```
[3.8139084095779494, 1.2315559335070287, 2.991354578014004, 2.483744909472883]
[1.1800355131070044, 1.1298727560017527, 3.4705827381682557, 2.114306043743788]
```

n(gaussian)	1	11	21	31
x	3.81	1.23	2.99	2.48
y	1.18	1.12	3.47	2.11

블러링 처리를 통해 떨림보정 시도 : 결과분석

3회 반복 평균의 결과 값에서 n=11 일때 분산이 가장 작았으므로
손의 떨림이 가장 작았다고 가정

Gaussian filter에 사용 할 n값을 11이라고 도출 하였다.

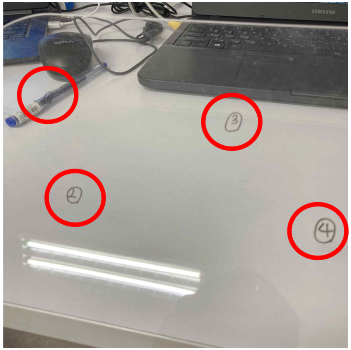
블러링 처리를 통해 떨림보정 시도 : 결과분석

차후 개선 점) 실험마다 집중하여 손을 떨지 않으면 n=1 일때와 n=31일때 결과의 큰 차이를 얻기 힘들었다.

실험 결과의 신뢰도를 높이기 위해서는 실험 반복횟수를 늘리고 매번 동일한 떨림을 주는 손모양 장치를 구현하여 실험하여야 보다 정확한 실험이 가능 할 것으로 보인다.

Part 2

N개의 frame별 이동평균을 구하여 떨림보정 시도



1. 1~4번을 그려놓은 아크릴판을 카메라와 동일한 거리에 이격시켜놓는다.

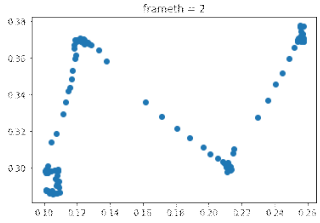
2. 손가락을 1~4번까지 차례대로 이동

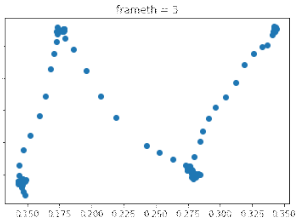
3.반환받는 x, y에 대한 좌표를 산점도로 나타내보고 손의 떨림이 얼마나 보정되었는가를 비교

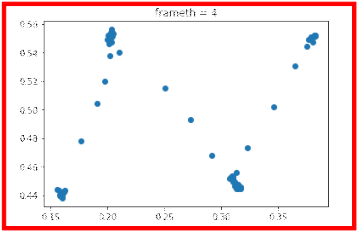
4. frame threshold값은 2,3,4,5,6,10을 사용

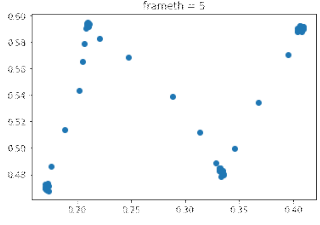
Part 2

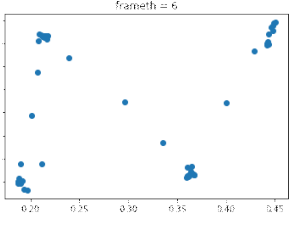
N개의 frame별 이동평균을 구하여 떨림보정 시도

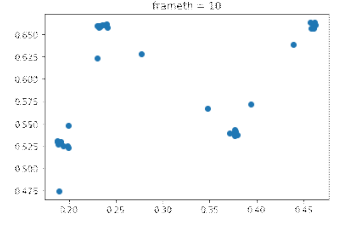












N개의 frame별 이동평균 : 결과분석

n frame 저장하여 평균을 내어 산점도를 그려본 결과
n값이 증가 할 수록 손의 떨림이 보정되었다.

하지만 frame값이 커지면 커질수록 마우스 이동이 거칠어지고 부드럽지 못하였기 때문에

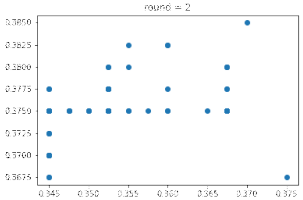
frame threshold 값은 4를 사용하였다.

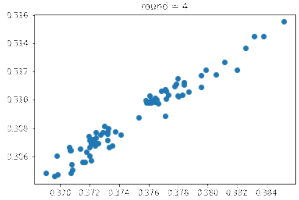
좌표값 반올림을 통한 떨림보정 시도

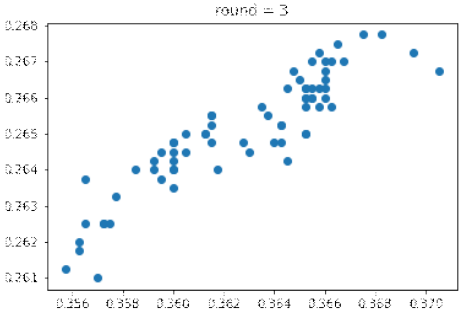
```
#frame_th 프레임의 평균값을 저장
if framecount<frame_th:
    nx += round(hp[8][0],roundvalue)
    ny += round(hp[8][1],roundvalue)

else:
    framecount=0
    ax, ay = nx/frame_th - bx, ny/frame_th - by
    cx, cy = pyautogui.position()
```

Part 2

좌표값 반올림을 통한 떨림보정 시도







문제점2 : 마우스 이동 문제

검지의 위치를 반환받아 화면상에서 움직이도록 하였더니
화면 끝으로 이동이 힘들었고

손과 카메라의 거리가 멀면
마우스 이동속도가 현저히 떨어졌다.

Part 2

이동 및 이동속도 개선을 위한 방안

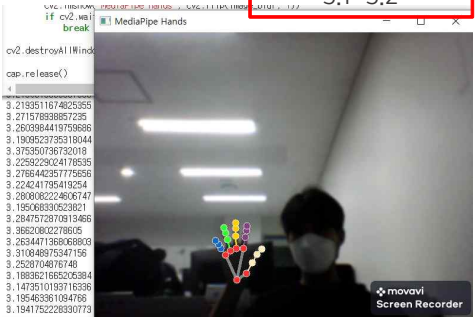
1. 사용자에 맞춰 ***화면크기를 반환***

2. 이전 검지위치와 현재 검지위치의 ***변화율*** 을 구해 ***마우스를 이동***

3. 손의 크기에 따라 ***가중치를 부여하여 가속***

손의 크기에 따라 가중치를 부여하여 가속하도록 함

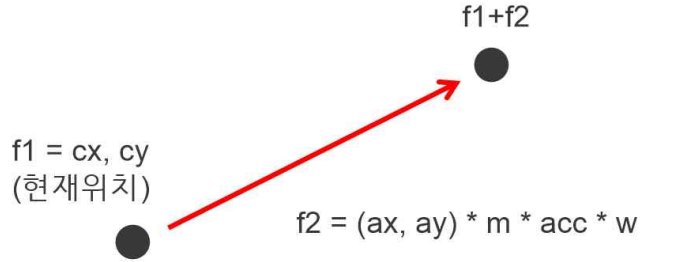
3.1~3.2



9.4~9.7

3~12 정도의 손크기 측정가능
(손크기는 검지, 약지의 시작부분을 기준으로 측정)
w = 7.5/handsize 를 사용하여 가속

이동 및 이동속도 개선



이동 및 이동속도 개선

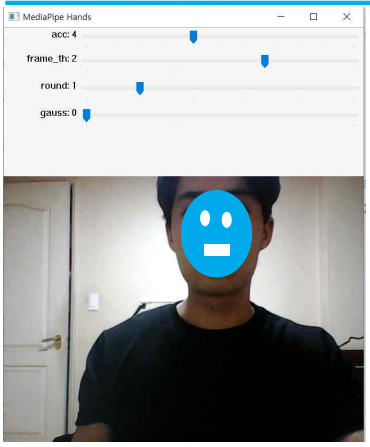
```
        ax, ay = nx/frame_th-bx, ny/frame_th-by
        cx, cy = pyautogui.position()

        if bx==0 and by==0:
            pyautogui.moveTo(moniter_x//2, moniter_y//2)

        else:
            pyautogui.moveTo(cx+(ax*moniter_x)*(-3)*w, cy+(ay*moniter_y)*3*w)

        bx, by = nx/frame_th, ny/frame_th
        xpert.append(bx)
        ypert.append(by)
        nx, ny = 0, 0

    #거리가반 가속
    d = ((hp[5][0]-hp[13][0])**2+(hp[5][1]-hp[13][1])**2+(hp[5][2]-hp[13][2])**2)**(1/2) #가속기능을 위한 거리계산
    print(d)
    w = 7.5/(d*100)
    #print(round(accel_distance,3))
```

부가기능

1. trackbar 기능 추가

2. click기능

trackbar 기능



acc : 사용자 설정 가속상수
default 4, 값이 커질수록 빨라집니다.

frame_th : default 2,
값이 커질수록 정확, 움직임이 느려집니다.

round : default 0, 설정 값 +2
값이 커질수록 부드럽지만 떨립니다.

gauss : default 1, 11x11 행렬
값이 커질수록 출력 화면이 흐려집니다.

클릭 기능

총 2가지 방법으로 접근

1. ***좌표 기반*** 으로 click 상태인지 아닌지 판별

2. ***딥러닝*** 으로 학습하여 click 상태인지 아닌지 판별

   - 직접 데이터 수집 (약 5000개 data)
   - 이진분류를 통해 상태확인

클릭기능 방법 1

```
#클릭기능 설정
cd = ((hp[5][0]-hp[1][0])**2+(hp[5][1]-hp[1][1])**2)**(1/2)
click_distance = ((hp[5][0]-hp[4][0])**2+(hp[5][1]-hp[4][1])**2)**(1/2)

if click_distance*100<4 and isclick==0:
    pyautogui.click()
    isclick=1

if click_distance*100>10:
    isclick=0
```

~~cd와 click_distance 간의 상관관계(미적용)~~

click_distance가 일정 값 미만일때 클릭상태이도록 설정

클릭기능 방법 1

연산속도가 크게 느려질 부분이 없어 마우스의 움직임에 제약을 많이 걸지 않는다.

손이 회전하면서 제대로 클릭인지 아닌지 구분하지 못한다.

가장 큰 문제점 -> 정확도가 현저히 떨어짐

-> Deep learing, 이진분류 모델을 통해 문제해결 시도

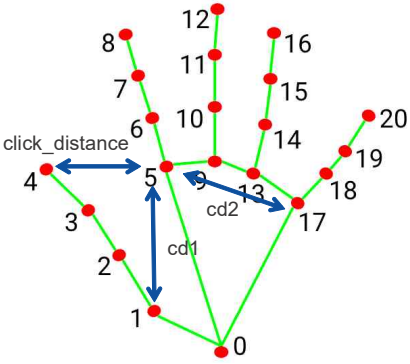
클릭기능 방법 2

데이터 수집 : 총 3개의 특성

cd1 (5번 1번과의 유클리드거리)

cd2 (5번 17번과의 유클리드거리)

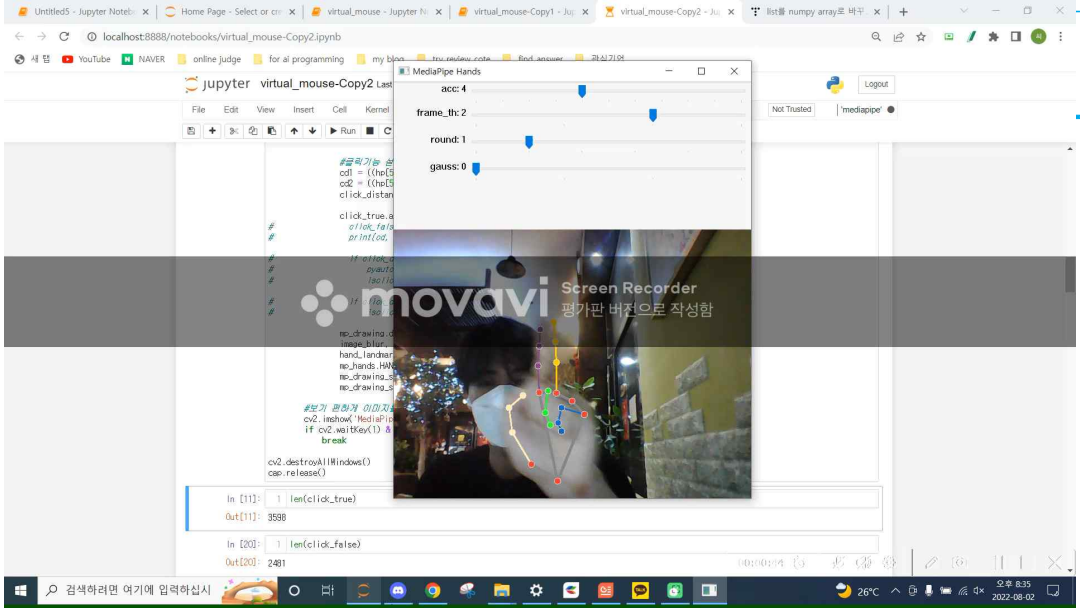
click_distance
(5번 4번과의 유클리드거리)



클릭기능 방법 2 : 데이터 수집

데이터 수집 : 총 3개의 특성, cd1, cd2, click_distance

```
#클릭기능 설정
cd1 = ((hp[5][0]-hp[1][0])**2+(hp[5][1]-hp[1][1])**2)**(1/2)
cd2 = ((hp[5][0]-hp[17][0])**2+(hp[5][1]-hp[17][1])**2)**(1/2)
click_distance = ((hp[5][0]-hp[4][0])**2+(hp[5][1]-hp[4][1])**2)**(1/2)

  click_true.append([click_distance, cd1, cd2])
click_false.append([click_distance, cd1, cd2])
  print(cd, click_distance)
```


MediaPipe Hands
acc: 4
frame_th: 2
round: 1
gauss: 0
jupyter virtual_mouse-Copy2
Logout
File Edit View Insert Cell Kernel
Not Trusted
'mediapipe'
click_true.a
click_false.a
mp_drawing.d
image_blur,
hand_landmar
mp_hands.HAN
mp_drawing_s
mp_drawing_s
cv2.imshow('MediaPip
if cv2.waitKey(1) &
break
cv2.destroyAllWindows()
cap.release()
In [11]: len(click_true)
Out[11]: 3598
In [20]: len(click_false)
Out[20]: 2481
Screen Recorder
평가판 버전으로 작성함

클릭기능 방법 2 : 데이터 수집

수집된 데이터 확인

```
print("click_true:", len(click_true))
print("click_false:", len(click_false))
print(click_true[0],"₩n", click_true[1])
print(click_false[0],"₩n", click_false[1])
```

```
click_true: 2960
click_false: 2449
[0.05084685744895077, 0.2053072807774375, 0.11303833070436509]
 [0.05132383789134265, 0.20808914684164947, 0.11484261509781575]
[0.13789889931178587, 0.20200437652336622, 0.10511311753852137]
 [0.14141757631915533, 0.21341326528022686, 0.10891832898624165]
```

Imbalanced data로 인한 문제가 일어나지 않게
최대한 비슷한 개수로 수집

클릭기능 방법 2 : 데이터 수집

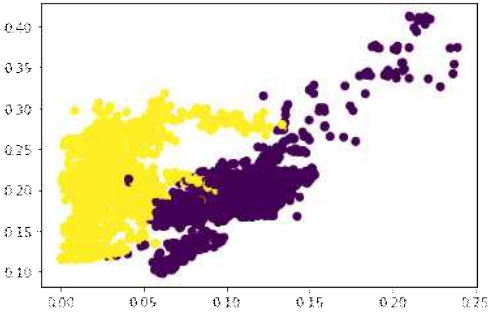
데이터 라벨링

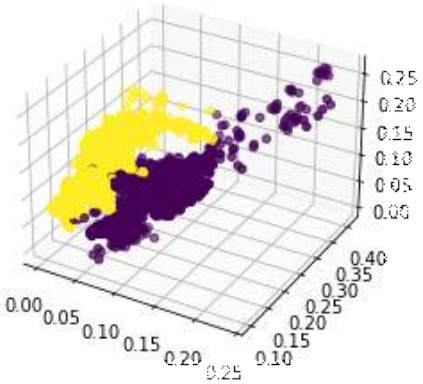
```
aaaa=0
for i in range(len(click_false)):
    click_false[i].append(0)
    aaaa+=1
    if aaaa<5:
        print(click_false[i])
```

```
[0.13789889931178587, 0.20200437652336622, 0.10511311753852137, 0]
[0.14141757631915533, 0.21341326528022686, 0.10891832898624165, 0]
[0.1406597523123106, 0.21309746610289604, 0.10810035356736766, 0]
[0.1396035011619723, 0.2073108601367239, 0.10483506661849953, 0]
```

클릭기능 방법 2 : 데이터 수집

데이터 시각화 True : ● , False: ●





클릭기능 방법 2 : 데이터 수집

train, test
데이터 준비

```
x_train, x_valid, y_train, y_valid = train_test_split(x, y,
                                                      test_size=0.2,
                                                      shuffle=True,
                                                      stratify=y,
                                                      random_state=42)
```

```
import numpy as np

# 데이터셋 생성
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test = np.array(x_valid)
y_test = np.array(y_valid)

print(x_train.shape)
print(y_train.shape)
```

```
(4327, 3)
(4327, 1)
```

클릭기능 방법 2 : 데이터 수집

학습모델 준비

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(64, input_dim=3, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

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

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 321
Trainable params: 321
Non-trainable params: 0
_________________________________________________________________
```

클릭기능 방법 2 : 데이터 수집

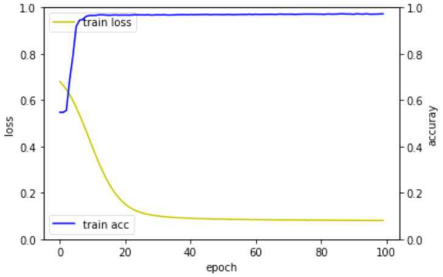
모델 학습

```
# 3. 모델 학습과정 설정하기
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

# 4. 모델 학습시키기
hist = model.fit(x_train, y_train, epochs=100, batch_size=64)
```

```
Epoch 98/100
68/68 [==============================] - 0s 2ms/step - loss: 0.0805 - accuracy: 0.9716
Epoch 99/100
68/68 [==============================] - 0s 1ms/step - loss: 0.0807 - accuracy: 0.9723
Epoch 100/100
68/68 [==============================] - 0s 1ms/step - loss: 0.0806 - accuracy: 0.9725
```

클릭기능 방법 2 : 데이터 수집

학습과정 시각화 및 모델 평가



accuracy : 97.32%

```
34/34 [==============================] - 0s 1ms/step - loss: 0.0764 - accuracy: 0.9732
loss_and_metrics : [0.07636195421218872, 0.9731977581977844]
```

클릭기능1

클릭기능2

WIN

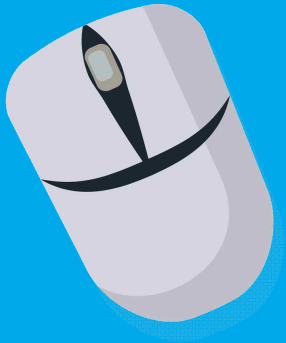
마우스 움직임

VS.

클릭 정확도

WIN

전체코드리뷰 및 동작



PART 3.

자가 피드백

장점 및 단점 분석

장점	단점
무료사용가능	성능 문제 (가상마우스<기존마우스)
위생 문제점 해결 가능	보안 문제
-	수익성 문제

개선점

- 오프라인에서 사용시 보안문제 해결 및 편리함을 제공

- 제스처에 따른 스크롤, 더블클릭 등 pyautogui 모듈 활용,
더 다양한 기능의 가상마우스 제작시도

- 카메라에 따른 화면 밝기 자동조정을 통해 손 인식률 향상


Q&A

감사합니다