골프자세 교정 소프트웨어

20162911 김경빈
20171464 김인혜
20185158 안진영

목차

설계 주제 및 선정배경

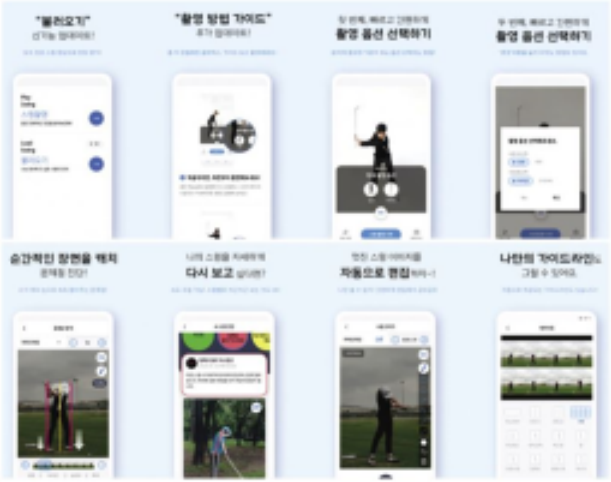
OpenPose와 딥러닝을 이용해서 프로선수의 골프 자세를 학습시켜 이용자의 자세를 교정해주는 프로그램

스포츠를 좋아하는데 혼자서는 자세 교정하는 것이 쉽지 않습니다. 자세를 교정해 주는 소프트웨어가 있으면 좋겠다고 생각했습니다.

현기술의 현황 및 시장 현황

AI 스윙분석 앱 골프픽스 서비스 업데이트

민경준 | 승인 2021.02.08 09:54 | 댓글 0



HOME > 골프일반 > 일반

내수진작 효과 최대 3.1조-2023년 골프산업 시장규모는 9.2조원

이계윤 | 승인 2020.11.18 09:55 | 댓글 0

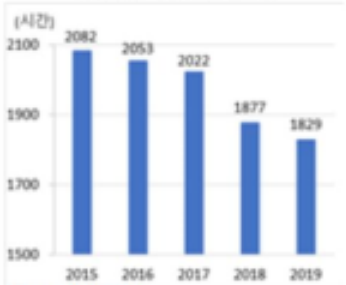
> 현대경제연구원 경제주평 '골프산업의 재발견과 시사점'
>
> 골프산업계, 정부 공조 통한 산업적 가치 창출과 지속 성장 전략 필요

현대경제연구원은 코로나19 실물경제 충격에도 불구하고 대한민국 골프산업에 대한 스포츠 활동 및 산업적 관심이 지속됨에 따라 산업적 가치 창출과 지속 성장을 위한 전략 마련이 필요하다고 내다봤다(사진은 기사의 특정 내용과 관계 없음).

현기술의 현황 및 시장 현황

여가 활동으로서 골프 수요 확대

< 연평균 총 근로시간 >



자료 : 고용노동부.
주 : 연평균 근로시간은 월평균 총 근로시간x12개월.

< 체육 동호회 인기 종목 상위 10개 >

자료 : 문화체육관광부 '2019 국민생활체육조사 보고서'.
주 : 국내 10세 이상 9,000명 중 체육 동호회 가입자(약 13%) 대상 설문조사 결과.

▲ 출처=현대경제연구원

골프의 대중화

< 골프장 수 현황 >

자료 : 한국골프장경영협회.

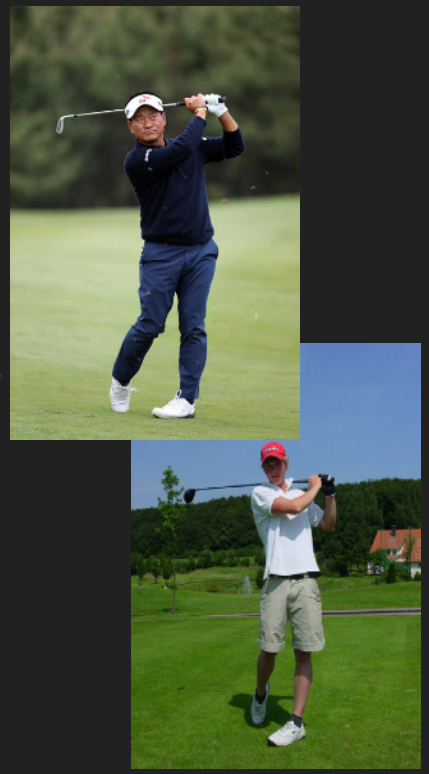
< 골프장 이용객 수 현황 >

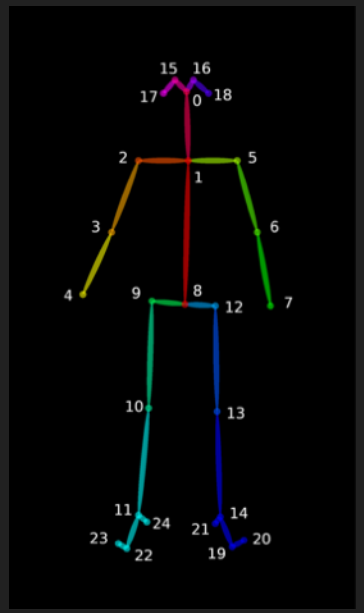
자료 : 한국골프장경영협회.

▲ 출처=현대경제연구원

데이터 베이스

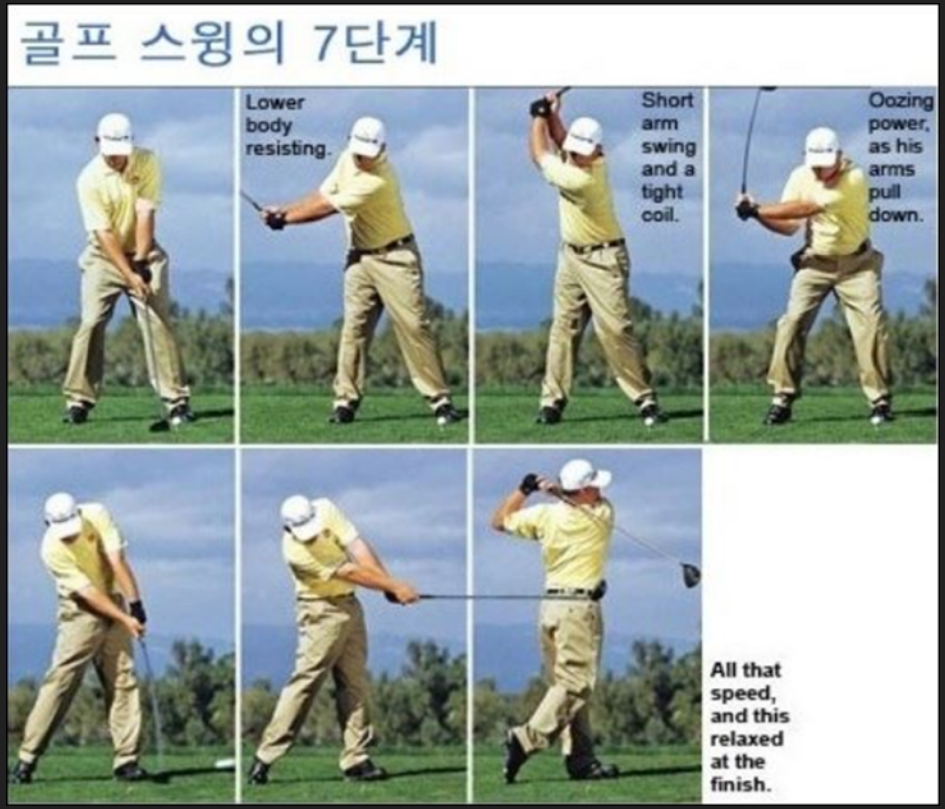
YouTube





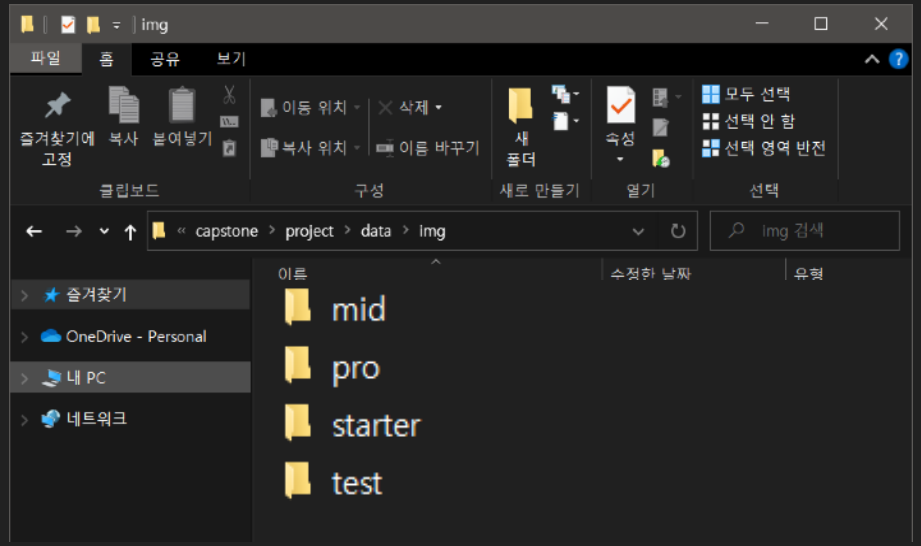
test_0.npy
test_1.npy
test_2.npy
test_3.npy
test_4.npy
test_5.npy
test_6.npy
test_7.npy
test_8.npy

데이터 베이스




```
label = np.zeros((9))
for i in range(len(video_path)):
    cv.destroyAllWindows()
    cap = cv.VideoCapture(video_path[i])      ## data 폴더에 있는 영상입력

    frame_num = 0
    print("path : ", video_path[i])
    while (cap.isOpened()):
        ret, frame = cap.read()
        if not ret:
            break
        cv.imshow("video", cv.resize(frame,dsize=None, fx=.5, fy=.5))

        tmp = cv.waitKey() & 0xff       # 프레임별 자세 입력
        print("tmp : ", tmp)
        if tmp == ord('n'):             # n일 경우 어떠한 자세에 속하지 않을 경우
            continue
        elif tmp == ord('q'):           # q일 경우 종료
            break
        elif 48 <= tmp <= 59:                           # 숫자일 경우 저장시킨다.
            while os.path.isfile(img_dir+"/%d/%d.png"%(tmp-48, label[tmp-48])):
                label[tmp - 48] += 1
            cv.imwrite(img_dir+"/%d/%d.png"%(tmp-48, label[tmp-48]), frame)

            label[tmp-48] += 1
```

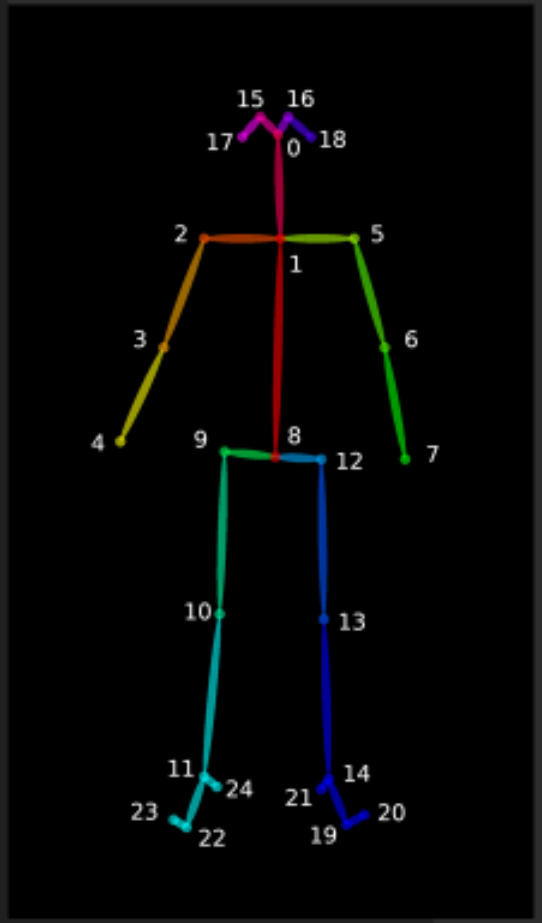
데이터 베이스


img
파일
홈
공유
보기
즐겨찾기에 고정
복사
붙여넣기
이동 위치
복사 위치
삭제
이름 바꾸기
새 폴더
속성
모두 선택
선택 안 함
선택 영역 반전
클립보드
구성
새로 만들기
열기
선택
capstone
project
data
img
img 검색
이름
수정한 날짜
유형
즐겨찾기
OneDrive - Personal
내 PC
네트워크
mid
pro
starter
test


0
1
2
6
7
4
5
6
8
9
10
11
14
4
5
6
7
8
9
10
11
12
13
14
15

데이터 베이스

front_starter1.npy : (55, 25, 3) front_starter2.npy : (47, 25, 3) front_starter3.npy : (72, 25, 3) front_starter4.npy : (23, 25, 3) front_starter5.npy : (32, 25, 3) front_starter6.npy : (26, 25, 3) front_starter7.npy : (60, 25, 3)	starter 1: (91, 25, 3) starter 2: (56, 25, 3) starter 3: (82, 25, 3) starter 4: (29, 25, 3) starter 5: (37, 25, 3) starter 6: (29, 25, 3) starter 7: (85, 25, 3)
1학기 DB	2학기 DB



특징점 정규화

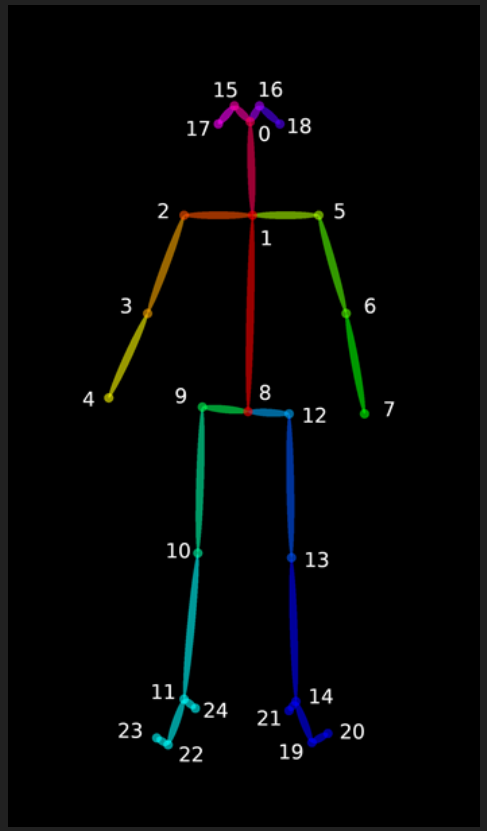
- 8번 점 (골반 중앙) 을 (0,0) 으로 설정.
- 위로 먼 부분 1, 아래로 먼 부분 -1 오른쪽 1 왼쪽 -1 으로 정규화를 진행했습니다.

```
[[ 73.60588074  41.96797562]
 [ 86.21298588  67.27613068]
 [ 70.79972076  68.65390778]
 [ 69.37403107  99.59199524]
 [ 65.13619232 131.22970581]
 [ 99.4027832   66.57342529]
 [105.90407562 101.80734711]
 [109.30562592 132.64654541]
 [ 82.73176575 134.72846985]
 [ 72.89514923 134.72277832]
 [ 81.31771088 179.70068359]
 [ 85.55294037 219.07827759]
 [ 92.55344391 134.7286377 ]
 [ 93.97978973 179.01673889]
 [ 95.39020538 223.99559021]
 [ 72.87996674  37.76979828]
 [ 78.49486542  37.75500107]
 [  0.           0.        ]
 [ 90.45692444  40.57019424]
 [ 79.22891235 233.81315613]
 [ 84.84633636 234.53764343]
 [ 99.58813477 228.98542603]
 [ 70.07859802 228.9150238 ]
 [ 70.05908203 226.79194641]
 [ 87.63347626 223.27368164]]
```

```
[[-5.18646639e-01 -9.56555390e-01]
 [ 1.50228447e-01 -6.95575192e-01]
 [-6.78127657e-01 -6.81367418e-01]
 [-7.59153132e-01 -3.62330801e-01]
 [-1.00000000e+00 -3.68796008e-02]
 [ 7.28668806e-01 -7.02821559e-01]
 [ 1.00000000e+00 -3.47735552e-01]
 [ 7.58399152e-01 -2.14698107e-02]
 [ 0.00000000e+00  0.00000000e+00]
 [-5.59039270e-01 -5.86916029e-05]
 [-8.03642391e-02  4.50581967e-01]
 [ 1.21747665e-01  8.45110772e-01]
 [ 4.23854083e-01  1.68167588e-06]
 [ 4.85407974e-01  4.43729443e-01]
 [ 5.46274398e-01  8.94377913e-01]
 [-5.59902128e-01 -9.99847410e-01]
 [-2.40793535e-01 -1.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00]
 [ 3.33378879e-01 -9.70969450e-01]
 [-1.99075831e-01  9.92741275e-01]
 [ 9.12542008e-02  1.00000000e+00]
 [ 7.27435854e-01  9.43570143e-01]
 [-7.19110848e-01  9.43663304e-01]
 [-7.20219990e-01  9.22394939e-01]
 [ 2.11533099e-01  8.87145025e-01]]
```

특징점 + 각도추가

```python
def normalizeAndCalcDegree(data):
    """
    25 feature point + 10 degree data(3, 2, 1(vertical), 1(horizon), 5, 6, 9, 10, 12, 13, 8)
    :param data: OpenPose feature point
    :return: 60 x 1 ndarray (normalized point + normalized degree)
    """
    result = np.zeros((61), dtype=np.float64)

    result[:50] = normalization(data).reshape((50))

    result[50] = calcDegree([data[4], data[3], data[2]])
    result[51] = calcDegree([data[3], data[2], data[1]])
    result[52] = calcDegree([data[0], data[1], data[8]])
    result[53] = calcDegree([data[2], data[1], data[5]])
    result[54] = calcDegree([data[1], data[5], data[6]])
    result[55] = calcDegree([data[5], data[6], data[7]])
    result[56] = calcDegree([data[8], data[9], data[10]])
    result[57] = calcDegree([data[9], data[10], data[11]])
    result[58] = calcDegree([data[8], data[12], data[13]])
    result[59] = calcDegree([data[12], data[13], data[14]])
    result[60] = calcDegree([data[9], data[8], data[12]])

    return result
```



데이터 학습

데이터 베이스를 계속 추가했기 때문에 모든 모델을 새로 학습시켰습니다.

```
2068/2068 - 0s - loss: 0.2722 - categorical_accuracy: 0.9478
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 256)               13056
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 455
=================================================================
Total params: 120,455
Trainable params: 120,455
Non-trainable params: 0
_________________________________________________________________
```

```
0 : [4 0 4 0 0 0 0 1 1 0]
1 : [3 1 1 3 1 1 1 1 1 1]
2 : [2 2 2 2 2 2 2 2 2 2]
3 : [3 3 3 3 1 1 1 3 1 4]
4 : [5 4 5 4 0 0 0 3 3 5]
5 : [1 5 5 5 5 5 5 5 5 5]
6 : [6 6 5 2 6 6 6 6 6 6]
70/70 - 0s - loss: 1.1938 - categorical_accuracy: 0.6857
```

```
0 : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
1 : [3 3 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 3]
2 : [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
3 : [2 6 3 3 0 0 3 3 1 1 3 3 3 3 3 3 3 3 3 3]
4 : [0 0 5 4 0 0 4 4 4 0 4 4 4 4 3 4 4 4 4 3]
5 : [5 5 5 5 5 5 5 5 5 5 6 5 5 5 5 5 5 5 5 5]
6 : [6 6 6 6 6 6 6 6 6 6 6 6 2 2 6 6 6 6 2 2]
[ 10  25  59  72  91 103 124]
140/140 - 0s - loss: 1.6439 - categorical_accuracy: 0.8357
```

데이터 학습자세 12 판별

```
Epoch 100/100
970/970 - 0s - loss: 0.0725 - categorical_accuracy: 0.9856
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 256)               13056
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
=================================================================
Total params: 112,131
Trainable params: 112,131
Non-trainable params: 0
_________________________________________________________________
40/40 - 0s - loss: 0.0175 - categorical_accuracy: 1.0000
```

```python
model_is12 = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(25, 2)),
    tf.keras.layers.Dense(256, kernel_initializer='he_normal', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(256, kernel_initializer='he_normal', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(128, kernel_initializer='he_normal', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dense(3, activation='softmax')
])

SGD = tf.keras.optimizers.SGD(lr=.00005)
adam = tf.keras.optimizers.Adam(learning_rate=.00001)
model_is12.compile(optimizer=adam, loss='categorical_crossentropy', metrics=['categorical_accuracy'])
model_is12.fit(x_train, y_train, epochs=100, verbose=2)

model_is12.summary()

predict = model_is12.predict(x_test)
```

데이터학습자세 47 판별

```
1028/1028 - 0s - loss: 0.1004 - categorical_accuracy: 0.9679
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 256)               13056
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 645
=================================================================
Total params: 128,901
Trainable params: 128,901
Non-trainable params: 0
_________________________________________________________________
```

```
model_is47 = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(25, 2)),
    tf.keras.layers.Dense(256, kernel_initializer='he_normal', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(256, kernel_initializer='he_normal', activation='tanh',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(128, kernel_initializer='he_normal', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(128, kernel_initializer='he_normal', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dense(5, activation='softmax')
])

SGD = tf.keras.optimizers.SGD(lr=.00005)
adam = tf.keras.optimizers.Adam(learning_rate=.00005)
model_is47.compile(optimizer=adam, loss='categorical_crossentropy', metrics=['categorical_accuracy'])
model_is47.fit(x_train, y_train, epochs=200, verbose=2)

model_is47.summary()

predict = model_is47.predict(x_test)
model_is47.evaluate(x_test, y_test, verbose=2)
```

데이터 학습 포즈점수

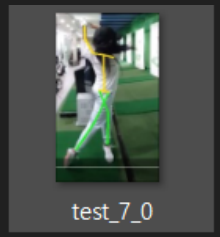

```
train1 = np.load("../data/np/front_pro%d.npy"%(num) )
train2 = np.load("../data/np/front_starter%d.npy"%(num))
train3 = np.load("../data/np/front_mid%d.npy"%(num))
train4 = np.load("../../front_0.npy")
test = np.load("../data/np/test%d.npy"%(num))

for pose_num in range(6):
    x_train = []
    y_train = []
    x_test = []
    y_test = []

    for i in range(len(train1)):
        x_train.append(utils.chosePart(utils.normalizeAndCalcDegree(train1[i]),pose_num).tolist())
        y_train.append(1)
    for i in range(len(train2)):
        x_train.append(utils.chosePart(utils.normalizeAndCalcDegree(train2[i]),pose_num).tolist())
        y_train.append(0.1)
    for i in range(len(train3)):
        x_train.append(utils.chosePart(utils.normalizeAndCalcDegree(train3[i]),pose_num).tolist())
        y_train.append(0.75)
    for i in range(len(train4)):
        x_train.append(utils.chosePart(utils.normalizeAndCalcDegree(train4[i]), pose_num).tolist())
        y_train.append(0.0)
    for i in range(len(test)):
        x_test.append(utils.chosePart(utils.normalizeAndCalcDegree(test[i]),pose_num).tolist())
```

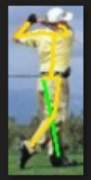
데이터 학습 포즈점수


```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(256, kernel_initializer='normal', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(256, kernel_initializer='normal', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(256, kernel_initializer='uniform', activation='relu',\
                          kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(1, activation='relu')
])
optimizer = tf.keras.optimizers.Adam(lr=1e-4)
model.compile(optimizer=optimizer, loss='mse', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=50, verbose=0)
model.evaluate(x_train, y_train, verbose=2)
print("pose", num," Scores", pose_dict[int(pose_num)])
model.summary()

print(model.predict(x_test))
```

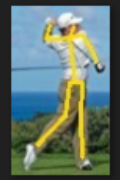
학습 결과

```
[[0.581 0.488 0.518 0.565 0.688 0.751]
 [0.646 0.581 0.47  0.557 0.655 0.671]
 [0.66  0.66  0.536 0.591 0.626 0.703]
 [0.472 0.477 0.632 0.57  0.639 0.498]
 [0.287 0.126 0.125 0.353 0.129 0.274]
 [0.624 0.666 0.62  0.709 0.798 0.88 ]
 [0.767 0.785 0.815 0.75  0.933 0.931]
 [0.994 1.08  1.09  0.944 1.087 1.122]
 [1.021 1.107 1.107 0.964 1.104 1.137]
 [0.    0.    0.    0.    0.    0.   ]]
```

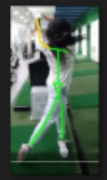

test_7_0


test_7_1


test_7_2


test_7_3


test_7_4


test_7_5

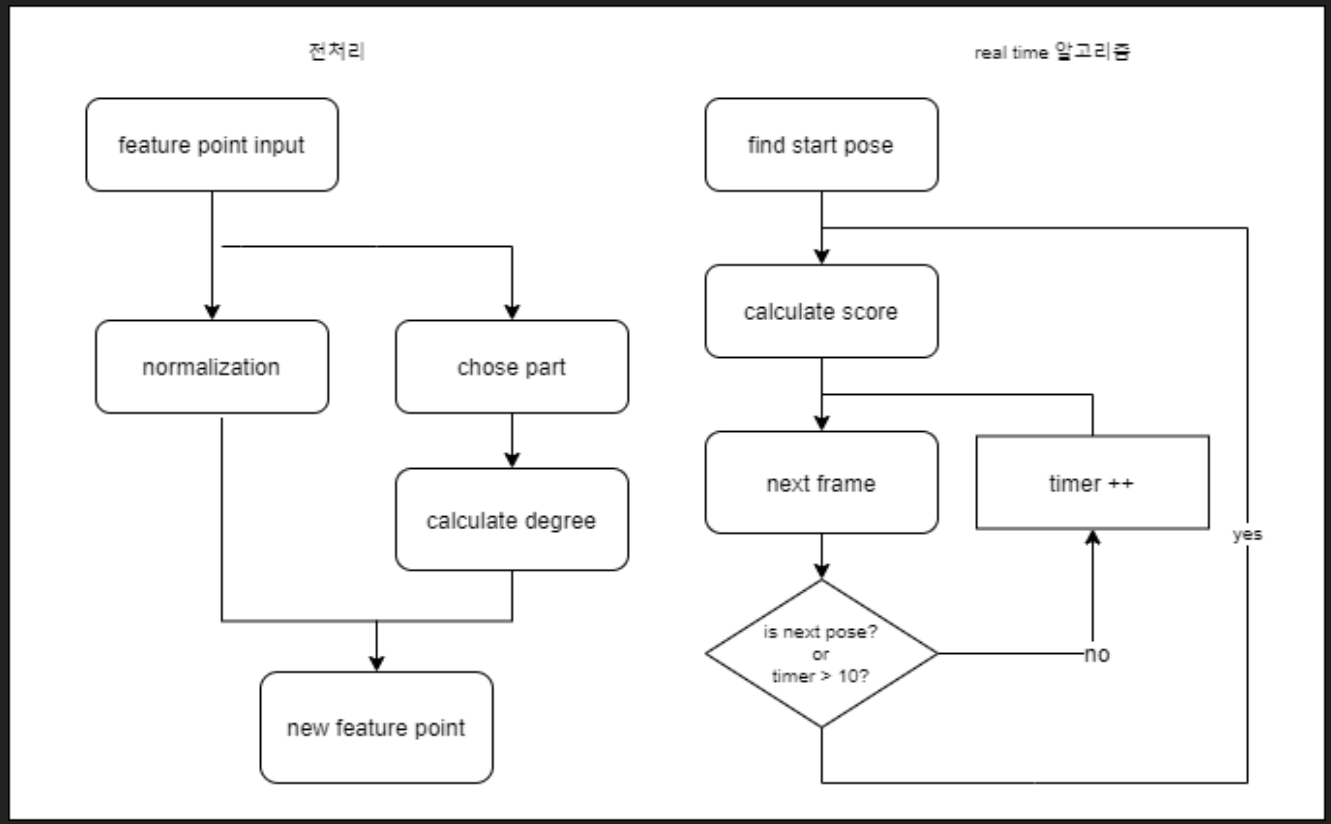

test_7_6


test_7_7


test_7_8


test_7_9

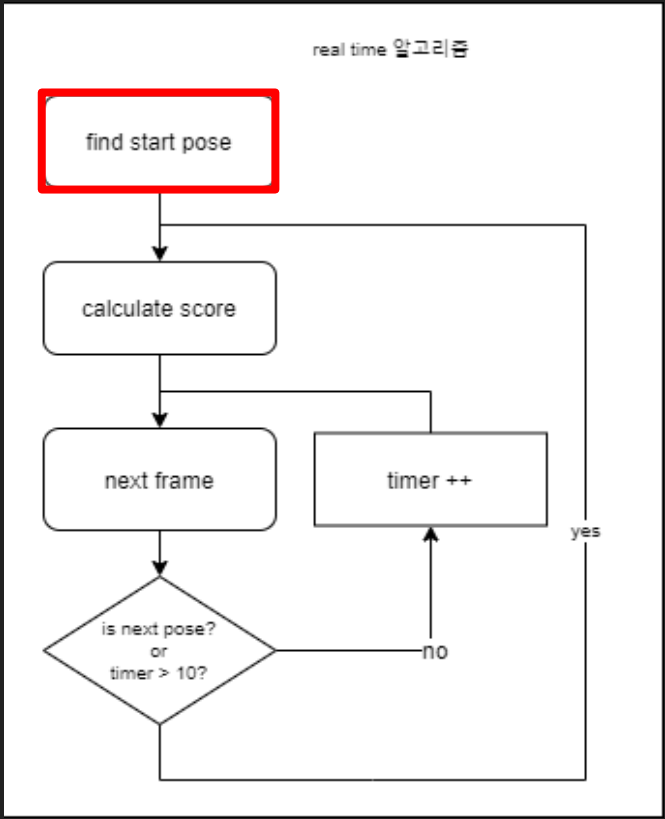
설계 알고리즘


전처리
feature point input
normalization
chose part
calculate degree
new feature point
real time 알고리즘
find start pose
calculate score
next frame
timer ++
is next pose?
or
timer > 10?
no
yes

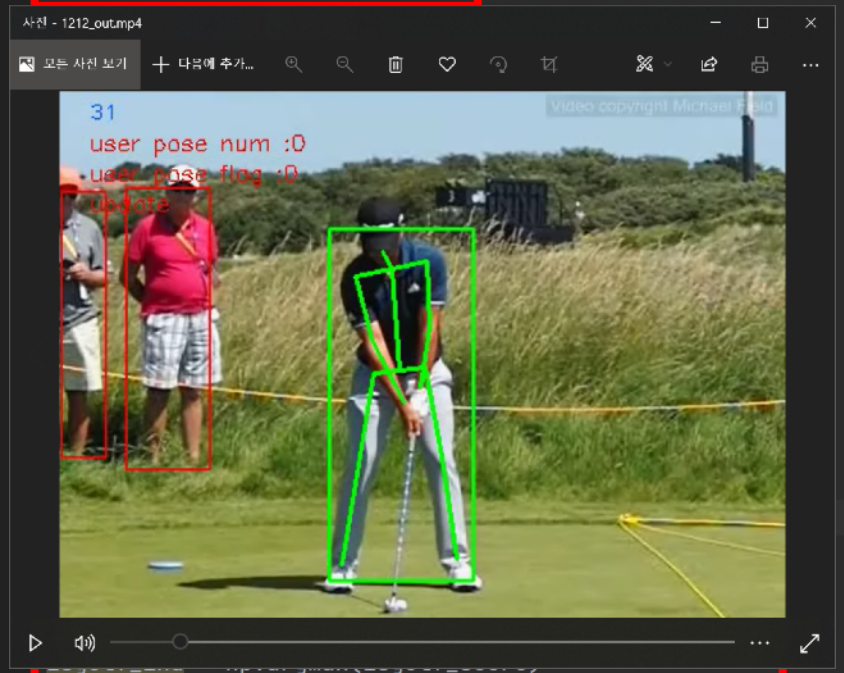
알고리즘

0.npy	2021-12-05 오후 9:45	NPY 파일
1.npy	2021-12-05 오후 9:45	NPY 파일
2.npy	2021-12-05 오후 9:45	NPY 파일
3.npy	2021-12-05 오후 9:46	NPY 파일
4.npy	2021-12-05 오후 9:46	NPY 파일
5.npy	2021-12-05 오후 9:46	NPY 파일
6.npy	2021-12-05 오후 9:46	NPY 파일
7.npy	2021-12-05 오후 9:46	NPY 파일
8.npy	2021-12-05 오후 9:47	NPY 파일
9.npy	2021-12-05 오후 9:47	NPY 파일
10.npy	2021-12-05 오후 9:47	NPY 파일
11.npy	2021-12-05 오후 9:47	NPY 파일
12.npy	2021-12-05 오후 9:47	NPY 파일
13.npy	2021-12-05 오후 9:47	NPY 파일
14.npy	2021-12-05 오후 9:47	NPY 파일
15.npy	2021-12-05 오후 9:48	NPY 파일
16.npy	2021-12-05 오후 9:48	NPY 파일
17.npy	2021-12-05 오후 9:48	NPY 파일
18.npy	2021-12-05 오후 9:48	NPY 파일

real time 알고리즘

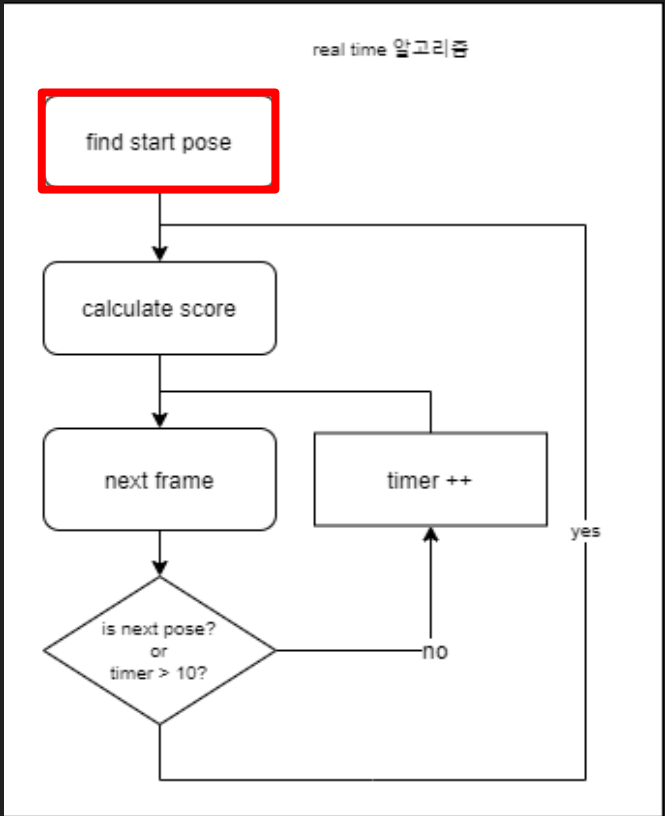



```
imageToProcess = cv2.resize(frame, dsize=None, fx=0.5, fy=0.5)
datum.cvInputData = imageToProcess
opWrapper.emplaceAndPop(op.VectorDatum([datum]))

keypoint = datum.poseKeypoints
```

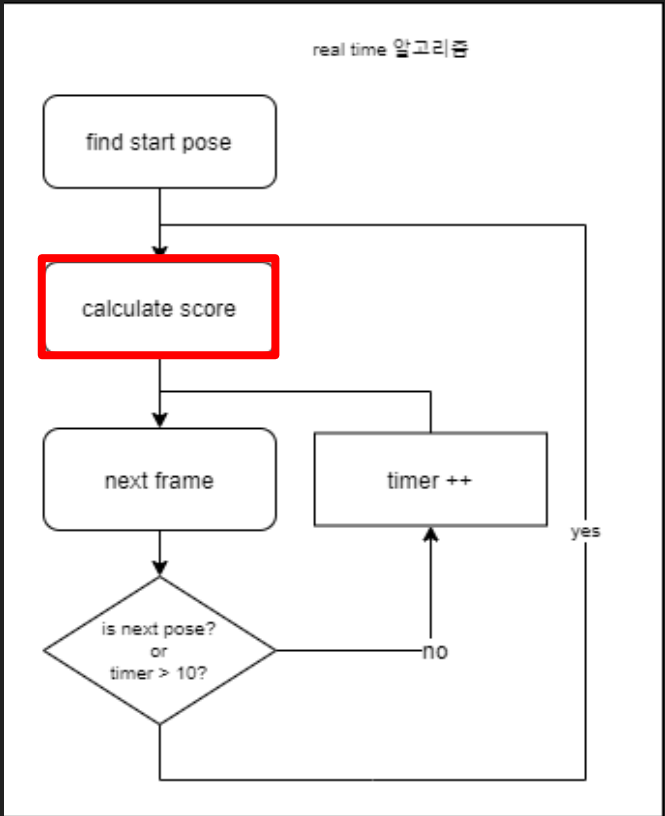



```
keypointGolf = keypoint[isgolf_ind]
keypointGolfNorm = utils.normalization(keypointGolf)
```

real time 알고리즘



real time 알고리즘




```
predict3 = model_pose_num.predict(keypointGolfNorm.reshape((1, 25, 2)))
print("predict 3: ", predict3, predict3.shape)
```

```
if np.sum(predict3[0, 5: 7]) > .95 and userPoseNum < 2:
    userPoseFlag = 1
elif np.sum(predict3[0, :2]) > .95 and (userPoseNum == 3 or userPoseNum >= 5):
    userPoseFlag = 0
```

```
if (predict3[0, 2] > 0.8 and 1 <= userPoseNum <= 2) or predict3[0, 2] > 0.95:
    userPoseFlag = 1
    userPoseNum = 2
    updateFlag = True
    for i in range(6):
        tmp = utils.chosePart(utils.normalizeAndCalcDegree(keypointGolf), i)
        partScore[i] = (model_score[2][i].predict(tmp.reshape(1, 53)))
```

real time 알고리즘

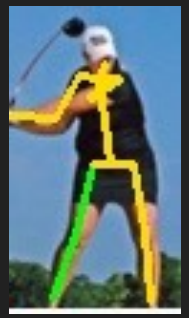

```
elif userPoseFlag == 0:
    predict12 = model_is12.predict(keypointGolfNorm.reshape(1,25,2))
    print("12", predict12)
    mypose = np.argmax(predict12, axis=1)[0]
    M = np.max(predict12)
    userPoseNum = mypose
    print("pose score : ", M + predict3[0, userPoseNum], np.count_nonzero(keypointGolfNorm))
    if (M > .98 or frameNumber % 10 == 0) and np.count_nonzero(keypointGolfNorm) >= 45:
        updateFlag = True
        for i in range(6):
            tmp = utils.chosePart(utils.normalizeAndCalcDegree(keypointGolf),i)
            partScore[i] = (model_score[mypose][i].predict(tmp.reshape((1,53))))
    # print("part score : ", mypose, partScore)
```

real time 알고리즘


```
elif userPoseFlag == 1:
    predict47 = model_is47.predict(keypointGolfNorm.reshape(1,25,2))
    print("47",predict47)
    mypose = np.argmax(predict47, axis=1)[0]
    userPoseNum = mypose+3
    M = np.max(predict47)
    print("pose score : ", M + predict3[0, userPoseNum], np.count_nonzero(keypointGolfNorm))
    if (M > .98 or frameNumber % 10 == 0) and np.count_nonzero(keypointGolfNorm) >= 45:
        updateFlag = True
        for i in range(6):
            tmp = utils.chosePart(utils.normalizeAndCalcDegree(keypointGolf), i)
            partScore[i] = (model_score[userPoseNum][i].predict(tmp.reshape(1,53)))
        # print("part score : ", userPoseNum, partScore)
        cv2.waitKey(10)
```

visualize

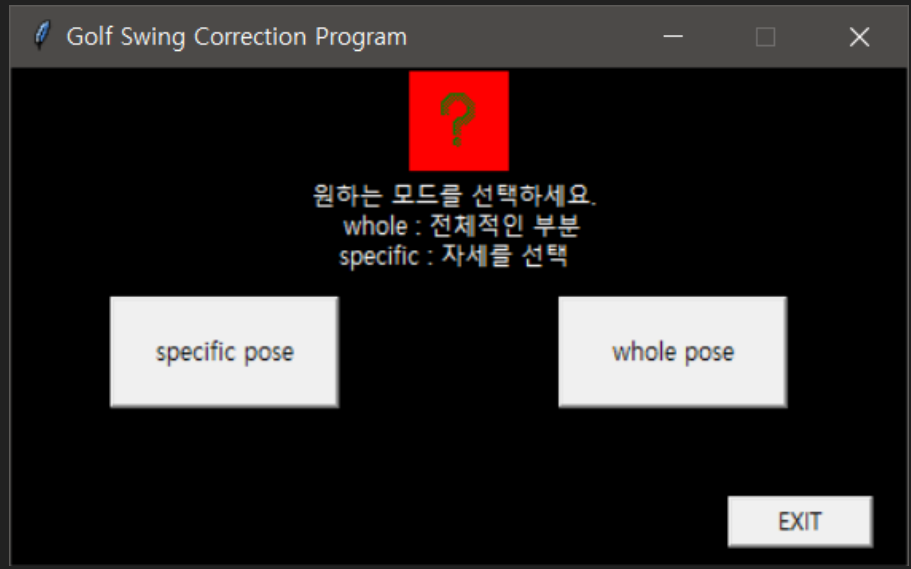
```python
def visualize(pose, score, img):
    def line(IMG, pt1,pt2,color, thick=2):
        if (int(pose[pt1][0]) +int(pose[pt1][1])) != 0 and (int(pose[pt2][0]) + int(pose[pt2][1])) != 0:
            cv.line(IMG,(int(pose[pt1][0]),int(pose[pt1][1])),(int(pose[pt2][0]),int(pose[pt2][1])),color,thickness=thick)
    image = cv.resize(img, dsize=None, fx=.5, fy=.5)
    for i in range(4):
        line(image, i,i+1, (0,255,0))
    line(image, 1, 5, (0, 255, 0))
    line(image, 1, 8, (0, 255, 0))
    line(image, 12, 8, (0, 255, 0))
    for i in range(5,7):
        line(image, i, i + 1, (0, 255, 0))
    for i in range(8,11):
        line(image, i, i + 1, (0, 255, 0))
    for i in range(12,14):
        line(image, i, i + 1, (0, 255, 0))

    if score[0] < 0.66:
        line(image, 2, 3, (0, 0, 255), thick=2)
        line(image, 4, 3, (0, 0, 255), thick=2)

    if score[1] < 0.66:
        line(image, 0, 1, (0, 0, 255), thick=2)
        line(image, 2, 1, (0, 0, 255), thick=2)
        line(image, 5, 1, (0, 0, 255), thick=2)

    if score[2] < 0.66:
        line(image, 5, 6, (0, 0, 255), thick=2)
        line(image, 7, 6, (0, 0, 255), thick=2)
```



UI

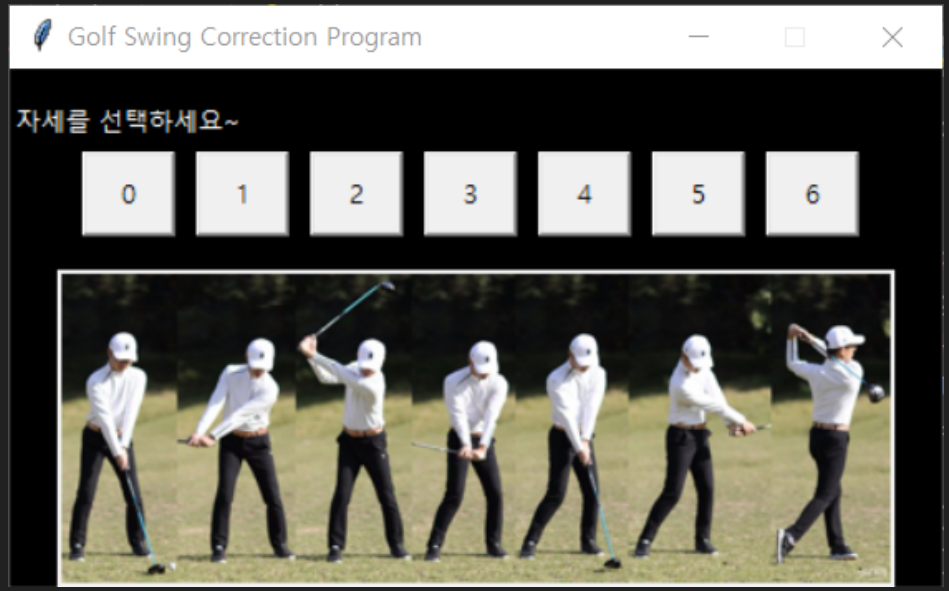
UI

```
Tk.title("Golf Swing Correction Program")

Tk.geometry("{}x{}+50+50".format(int(450), int(250)))
Tk.configure(bg='black')
Tk.resizable(False, False)

label1 = tk.Label(Tk, width=50, height=50,\
                fg='green',relief='solid', bitmap='question', bg='red')
label1.pack()

label2 = tk.Label(Tk, text="원하는 모드를 선택하세요. \n whole : 전체적인 부분\n specific : 자세를 선택  ")
label2.pack()

Button1 = tk.Button(Tk, width=15, height=3, text="specific pose",\
                    overrelief='solid', command=lambda :cmd1(),\
                    repeatdelay=1000, state='normal')
Button1.place(x=50, y=115)

Button2 = tk.Button(Tk, width=15, height=3, text="whole pose",\
                    overrelief='solid', command=lambda :cmd2(),\
                    repeatdelay=1000, state='normal')
Button2.place(x=275, y=115)

Tk.mainloop()
print(modeNum)
```

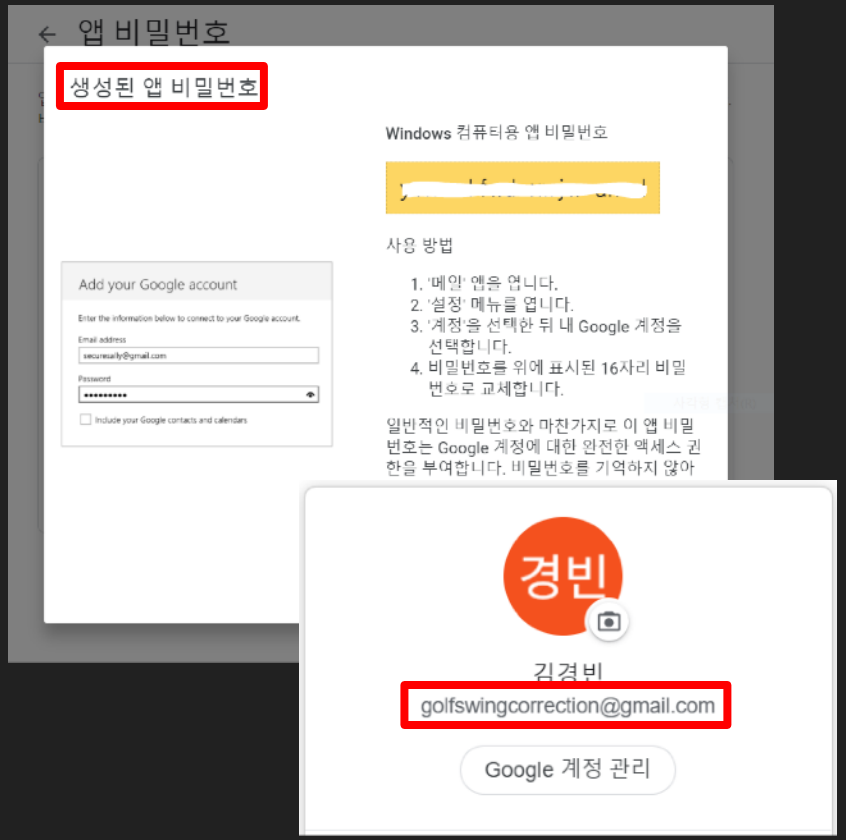


UI

```
Tk.title("Golf Swing Correction Program")

Tk.geometry("{}x{}+50+50".format(int(450), int(250)))
Tk.configure(bg='black')
Tk.resizable(False, False)
label = tk.Label(Tk, text="자세를 선택하세요~", bg='black', fg='white')
label.place(y=15)
buttonList = []
buttonList.append(tk.Button(Tk, text="{}".format(0), width=5, height=3,\
                            command=lambda: cmd(0)))
buttonList.append(tk.Button(Tk, text="{}".format(1), width=5, height=3,\
                            command=lambda: cmd(1)))
buttonList.append(tk.Button(Tk, text="{}".format(2), width=5, height=3,\
                            command=lambda: cmd(2)))
buttonList.append(tk.Button(Tk, text="{}".format(3), width=5, height=3,\
                            command=lambda: cmd(3)))
buttonList.append(tk.Button(Tk, text="{}".format(4), width=5, height=3,\
                            command=lambda: cmd(4)))
buttonList.append(tk.Button(Tk, text="{}".format(5), width=5, height=3,\
                            command=lambda: cmd(5)))
buttonList.append(tk.Button(Tk, text="{}".format(6), width=5, height=3,\
                            command=lambda: cmd(6)))

for i in range(7):
    buttonList[i].place(x=(i+1)*55-20, y=80)

Tk.mainloop()
return poseNum
```

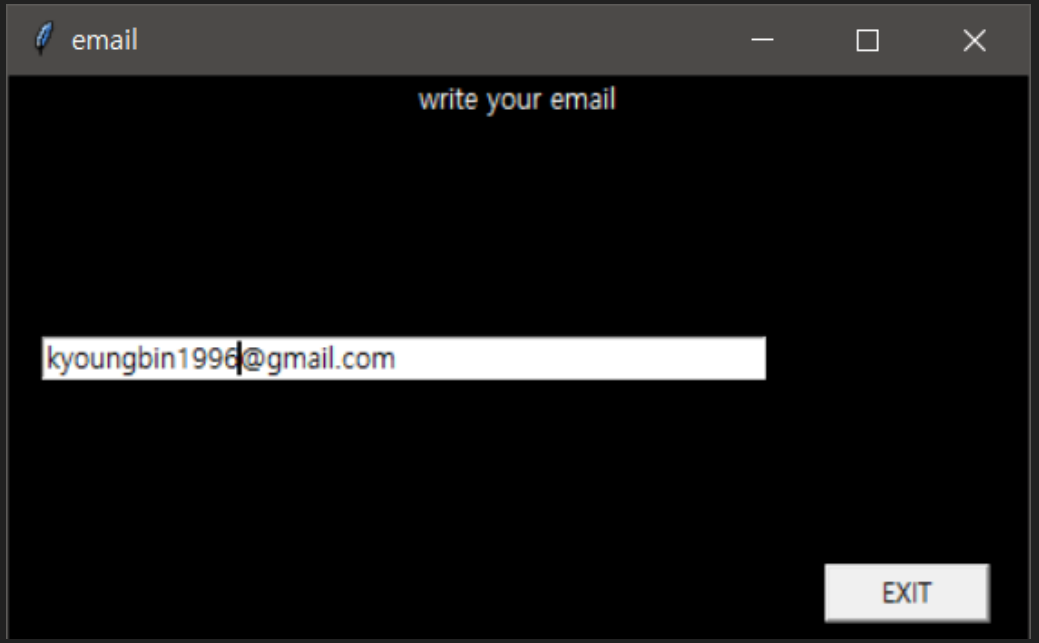


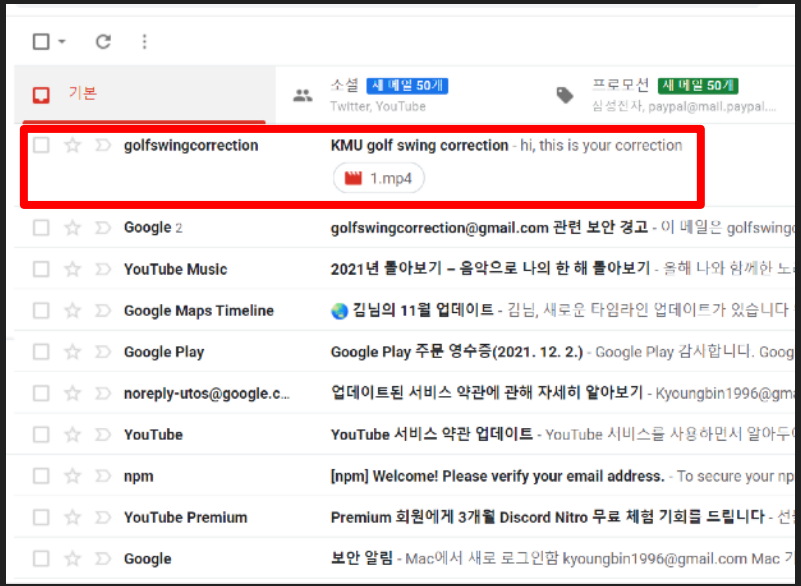
email




```python
## tkinter 창에서 엔터가 입력되었을때 실행되는 event call
def send_email(event):
    # 엔터키로 입력받기
    email_addr = str(entry.get())
    print(email_addr)
    label = tk.Label(root, text='sneding email')
    label.pack()
    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    s.login('golfswingcorrection@gmail.com', '                ')  # 로그인

    msg = MIMEMultipart()
    msg['Subject'] = 'KMU golf swing correction'  # 제목설정
    msg.attach(MIMEText('hi, this is your correction', 'plain'))  # 내용

    ## 첨부파일 1
    attachment = open('../data/resultvideo/out.mp4', 'rb')
    part = MIMEBase('application', 'octet-stream')
    part.set_payload((attachment).read())
    encoders.encode_base64(part)
    part.add_header('content-Disposition', 'attachment; filename= 1.mp4')
    msg.attach(part)

    ## 이메일 보내기
    s.sendmail('kmufacefilter@gmail.com', email_addr, msg.as_string())
    s.quit()
    label = tk.Label(root, text='successfully sned email')
    label.pack()
```

email


email
write your email
kyoungbin1996@gmail.com
EXIT


기본
소셜 새 메일 50개
Twitter, YouTube
프로모션 새 메일 50개
golfswingcorrection
KMU golf swing correction - hi, this is your correction
1.mp4
Google 2
golfswingcorrection@gmail.com 관련 보안 경고 - 이 메일은 golfswing
YouTube Music
2021년 돌아보기 – 음악으로 나의 한 해 돌아보기
Google Maps Timeline
김님의 11월 업데이트 - 김님, 새로운 타임라인 업데이트가 있습니다
Google Play
Google Play 주문 영수증(2021. 12. 2.) - Google Play 감사합니다.
noreply-utos@google.c...
업데이트된 서비스 약관에 관해 자세히 알아보기 - Kyoungbin1996@gm
YouTube
YouTube 서비스 약관 업데이트 - YouTube 시비스를 사용하민서 알아두
npm
[npm] Welcome! Please verify your email address. - To secure your np
YouTube Premium
Premium 회원에게 3개월 Discord Nitro 무료 체험 기회를 드립니다
Google
보안 알림 - Mac에서 새로 로그인함 kyoungbin1996@gmail.com Mac

구현영상

구현영상