

# 캡스톤 디자인 11월 2주차 보고서

주제명	제스처인식 기반 커서제어	
4조	팀장 : 임준혁, 박세준, 이재운	2021.11.05.~11.11

기존,  
개인별 계획

## <전체 계획>

	9	10	11	12	13	14	15	16
커서 제어 설계								
마우스 이벤트 설계								
키보드 이벤트 설계								
데이터 셋 수집 모델 설계								
데이터 셋 수집								
데이터 학습 모델 구현								
프로그램 통합								
웹 서버 구축								
메뉴얼 제작								
테스트 및 보완								

< 공통 >

[illegible]

## 개인별 계획

### <박세준>

	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
마우스 이벤트 구현										

#### <금주 내용>

- 마우스 이벤트 구현 ---80%
- 커서와 캡 정밀도, 사용성 조정 ---80%

### <임준혁>

	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
웹 서버 구축										
메뉴얼 제작										

#### <금주 내용>

- 제스처 인식 오류 해결 ---50%
- 웹 서버 구축 ---20%

### <이재운>

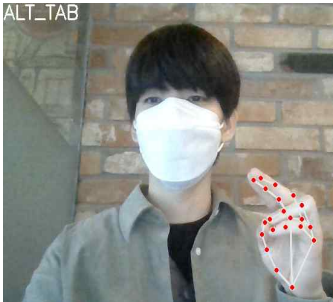


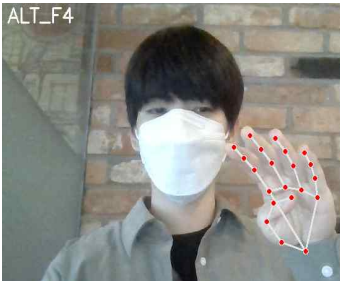
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
데이터 셋 수집 모델 구현										
데이터 셋 수집										
데이터 학습 모델 구현										
키보드 이벤트 구현										

#### <금주 내용>

- 데이터 셋 수집 모델, 학습 모델 제작 ---100%
- 인식률 조사 --- 100%
- 키보드 이벤트 구현 --- 40%

## 금주 계획

- 마우스 좌클릭, 우클릭, 휠클릭 위한 이벤트 구현
- 키보드 이벤트 위한 모델 구현

<p>금주의 수행 내용</p> <p>및 이슈,</p> <p>이슈 해결 방안</p>	<p>&lt;수행 내용&gt;</p> <p>-데이터 셋 수집 모델 구현:</p> <pre>actions =['ALT_TAB', 'ALT_F4', 'FULL', 'SOUND_CONTROL']</pre> <p>기존 설계안 제스처의 인식이 저조하고, 마우스 이벤트와의 충돌이 일어나 새로운 제스처로 설계</p> <p>&lt;기존 설계안의 제스처 인식 문제점&gt;</p> <ul style="list-style-type: none"> <li>-mediapipe의 손인식에서 손날 만의 인식과 빠른 행동의 인식의 어려움</li> <li>-기존 설계안에 작성했던 제스처 또한 해결방안을 찾기 위해 고민 중</li> </ul> <p>&lt;새로운 제스처&gt;</p> <ul style="list-style-type: none"> <li>-엄지 약지 새끼의 세 손가락이 맞닿아 있을 때만 키보드 이벤트가 발생하도록 설정</li> <li>-화면전환(ALT_TAB): 나머지 두 손가락을 슬래시 하는 제스처</li> <li>-전체화면(FULL): 두 손가락을 위로 올리고 내리는 제스처</li> <li>-볼륨조절: 검지손가락을 만을 움직여 볼륨을 제어하는 제스처</li> <li>-창닫기(ALT_F4): 검지 제외 네 손가락을 펴서 슬래시 하는 제스처</li> </ul> <div style="display: flex; flex-wrap: wrap;"> <div style="width: 50%;">  <p>그림 1 화면전환(ALT_TAB)</p> </div> <div style="width: 50%;">  <p>그림 2 전체화면(F)</p> </div> <div style="width: 50%;">  <p>그림 3 볼륨 제어</p> </div> <div style="width: 50%;">  <p>그림 4 창닫기(ALT_F4)</p> </div> </div>

```

v1 = joint([[0,1,2,3,0,5,6,7,0,9,10,11,0,13,14,15,0,17,18,19], :2]
v2 = joint([[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20], :2]
v = v2 - v1
v = v / np.linalg.norm(v, axis=1)[:, np.newaxis]

angle = np.arccos(np.einsum('nt,nt->n',
    v[[0,1,2,4,5,6,8,9,10,12,13,14,16,17,18],:],
    v[[1,2,3,5,6,7,9,10,11,13,14,15,17,18,19],:]))
angle = np.degrees(angle)

```

인접한 간선 벡터의 각도와 x, y좌표값을 받아 데이터로 수집.  
총 2463개의 데이터 셋으로 구성.

-학습 모델 구현:

```

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.1, shuffle=True, random_state=1)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, shuffle=True, random_state=1)

```

테스트 데이터 10%와, 나머지 검증 데이터 20%

```

model = Sequential([
    LSTM(64, activation='relu', input_shape=x_train.shape[1:3]),
    Dense(32, activation='relu'),
    Dense(32, activation='relu'),
    Dense(len(gesture), activation='softmax')
])

```

RNN알고리즘 적용

```

Epoch 00200: val_acc did not improve from 0.95942
77/77 [=====] - 1s 15ms/step - loss: 0.1192 - acc: 0.9610 - val_loss: 0.1537 - val_acc: 0.9545
ALT_TAB
[[246  7]
 [  7 83]]

ALT_F4
[[261  0]
 [  0 82]]

FULL
[[254 10]
 [  8 71]]

SOUND_CONTROL
[[250  1]
 [  3 89]]

```

학습 인식 정확도: 0.96

테스트 정밀도 = 0.88, 1, 0.87, 0.99 = 평균 0.94

-인식률 계획: 키보드와 마우스 제스처를 통합 했을 때, 실제 사용감이 어떨지 확인  
해 보고, 다시 인식률 테스트를 거쳐 인식이 떨어지는 제스처의  
데이터 셋을 추가 수집할 예정

-마우스 이벤트의 제스처 변경:

```
v_out = np.array([
    landmarks[0].landmark[13].x - landmarks[0].landmark[16].x,
    landmarks[0].landmark[13].y - landmarks[0].landmark[16].y,
    landmarks[0].landmark[13].z - landmarks[0].landmark[16].z
])
v_in = np.array([
    landmarks[0].landmark[0].x - landmarks[0].landmark[13].x,
    landmarks[0].landmark[0].y - landmarks[0].landmark[13].y,
    landmarks[0].landmark[0].z - landmarks[0].landmark[13].z
])
v_out = v_out / np.linalg.norm(v_out)
v_in = v_in / np.linalg.norm(v_in)
# print('4', v_in, v_out)
if np.dot(v_out, v_in) < threshold:
    return True
else:
    return False
```

손 마디가 맞닿는 방식에서 손가락이 구부러지는 정도를 인식하도록 설정



그림 11 좌클릭

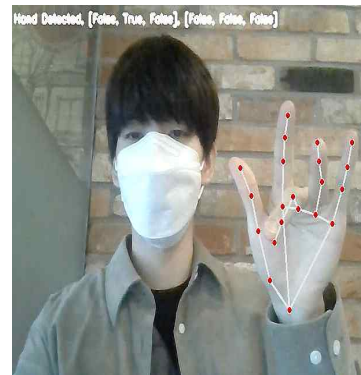


그림 10 우클릭



그림 12 휠클릭

	<p>- 사용성 상향을 위해, 캠 인식과 커서 기능과 정밀도 조정:</p> <ol style="list-style-type: none"> <li>1. 좌클릭 유지 상태의 드래그 기능과 같이, 제스처 각도 유지에 따라 마우스 이벤트의 유지 실행이 가능하도록 변경</li> <li>2. 커서 제어에 사용하던 pyautogui 라이브러리가 무거워 mouse 라이브러리로 대체</li> <li>3. 마우스 커서가 튀는 현상을 제거</li> </ol> <pre>def position_mouse(landmarks):     cursor_position_list_x.pop(0)     cursor_position_list_y.pop(0)     cursor_position_x, cursor_position_y = arithmetic_mean(list((landmarks[0].landmark[0], landmarks[0].landmark[17])))     cursor_position_list_x.append((((cursor_position_x - 0.5) / ratio) + 0.5) * display_width)     cursor_position_list_y.append((((cursor_position_y - 0.5) / ratio) + 0.5) * display_height)     mouse.move(sum(cursor_position_list_x) / len(cursor_position_list_x), sum(cursor_position_list_y) / len(cursor_position_list_y))</pre> <ol style="list-style-type: none"> <li>4. 시간복잡도를 줄이는 코드 최적화 과정</li> </ol>
차주 수행 계획	<p>&lt;차주 수행 계획&gt;</p> <ul style="list-style-type: none"> <li>-웹 서버 구축</li> <li>-인식률 보완</li> <li>-마우스와 키보드 이벤트 실제 실행 구현완성과 통합</li> </ul> <p>&lt;진척 상황 해결 방향안&gt;</p> <p>다음 주까지 마우스와 키보드 사용을 통합시키고, 프로그램 실제 사용성이 어떨지 체크하여 문제점을 찾고 정리 후, 프로젝트의 남은 기간 내에 최대한 사용성과, 인식률 증가를 계획해 코드 수정과 데이터셋 추가확보 등을 시행할 예정입니다.</p>
참고 문헌	<p><a href="https://google.github.io/mediapipe/solutions/hands.html">https://google.github.io/mediapipe/solutions/hands.html</a>  <a href="https://github.com/lena1005a/capstone_4_houseproject">https://github.com/lena1005a/capstone_4_houseproject</a></p>