

# 차량지능기초 전반부 최종 프로젝트

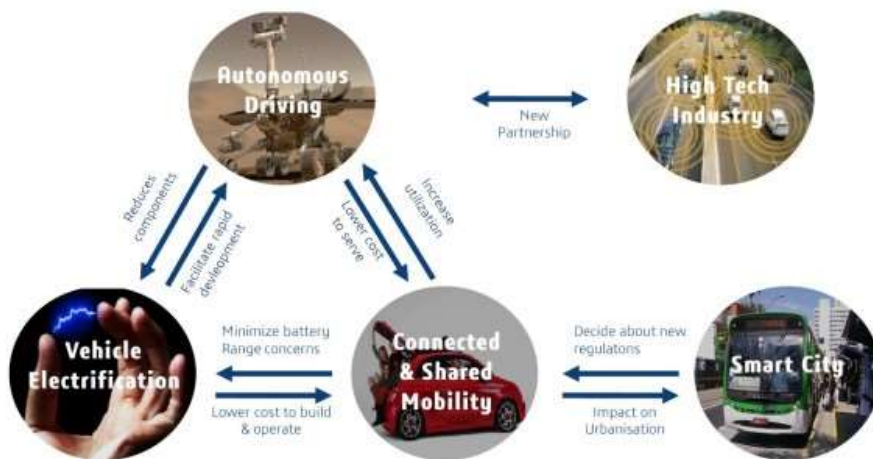
20173396 김찬영

## 1. 차량지능(지능형차량)의 개념

지능형자동차(Intelligent Vehicle)는 자동차에 기계, 전자, 통신, 제어, 인공지능 등 각종 첨단기술을 접목시켜 편의성을 크게 향상시킨 자동차로 운전자가 주행에 신경 쓸 필요 없이 자동으로 목적지까지 사고 없이 도착하게 하는 미래형 자동차를 말합니다. 주로 지능형 자동차 하면 자율주행자동차에 대해 많이 생각하게 됩니다. 하지만 저는 자율주행은 물론 운전자의 편의성과 안전성을 높여주는 지능형 자동차에 대해서도 생각해보고자 합니다.

## 2. 지능형차량에 대해

### Smart Mobility | Working with Innovation leaders

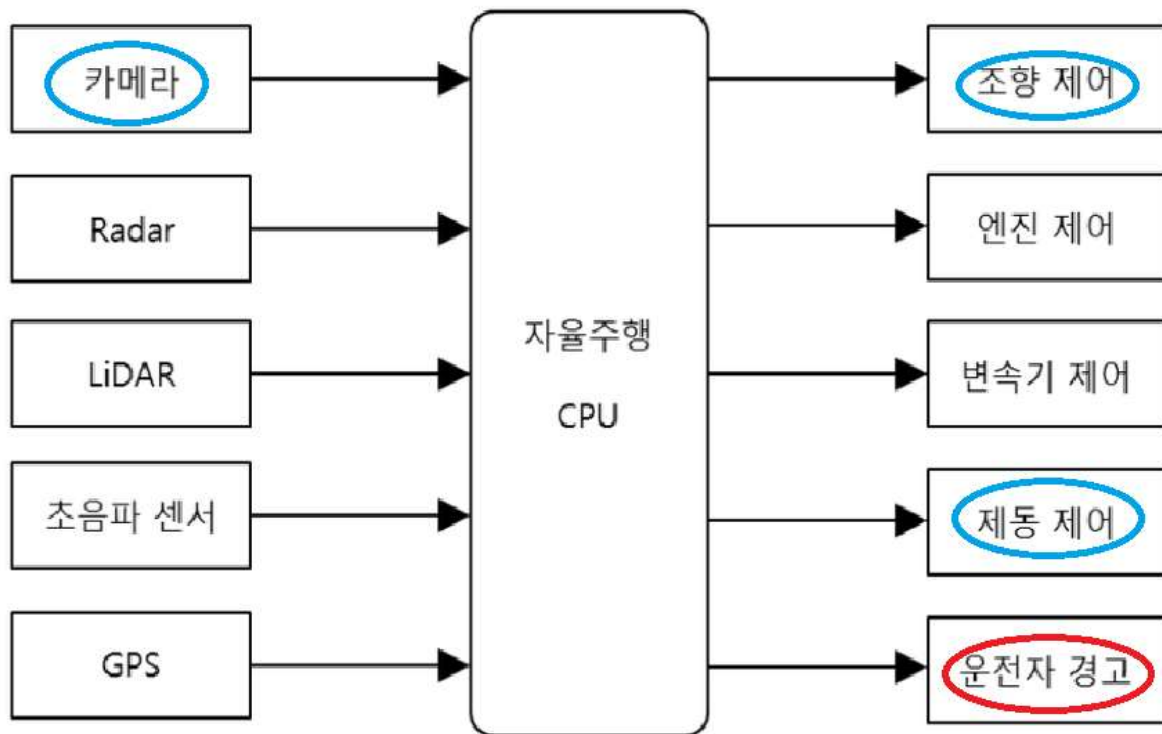


DS DASSAULT SYSTEMES

위의 그림은 자율주행 차량과 연결된 다양한 기술요소입니다. 지능형 차량은 차량지능기초 수업 첫 주차에 배운 내용에서 볼 수 있듯이 전기자동차와 공유경제(셰어링카)와 밀접한 관련이 있습니다. 그리고 지능형 차량끼리는 서로 연결되어서 도로의 교통상황을 파악하여 더욱 효율적으로 운용 할 수 있습니다.

### 3. 자율주행

일단 차량지능에 대해서 가장 널리 알려져 있는 자율주행에 대해서 알아보겠습니다. 자율주행의 핵심 기술은 '인식-판단-제어', 3단계로 나누어져 있습니다. 이런 단계를 거치기 위해서는 아래의 사진과 같이 주변환경을 센서기술과 카메라 기술로 환경을 인식하고 컴퓨터의 CPU, 사람의 뇌와 같이 판단하고 제어를 하게 됩니다. 아래에 동그라미친 것은 Mediapipe를 활용하여 알아볼 부분들을 표시해놓았습니다.



#### (1) 인식

자율주행차량에 장착된 레이더, 라이다, 카메라, GPS 등 다양한 센서기술로 주변환경을 인식합니다. 서로의 센서는 서로의 장점과 단점을 보완하여 높은 정확도로 주변환경을 파악합니다. 간단하게 설명하자면 레이더의 장점과 단점은 장거리의 물체와의 거리를 측정할 수 있지만 작은 물체를 식별할 수 없으므로 정밀한 이미지를 제공하지 못합니다. 반대로 라이다는 작은 물체를 감지할 수 있고 고해상도의 3D 이미지를 제작할 수 있습니다. 또한 카메라는 사물을 정확하게 표현할 수 있지만 얼마나 떨어져 있는지 거리감을 알기 힘들지만 레이더와 라이다가 있다면 어느정도 깊이정보에 대해 알 수 있습니다. 이렇게 서로의 단점을 보완하고 장점을 극대화시켜주며 정확한 인식을 제공합니다.

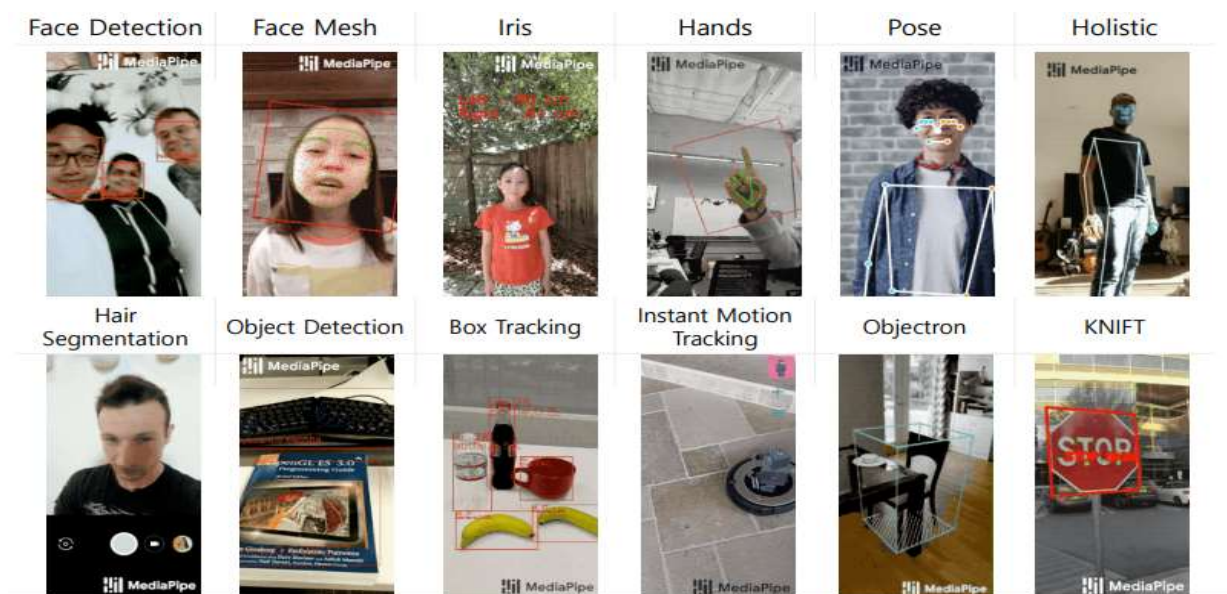


## (2) 판단과 제어

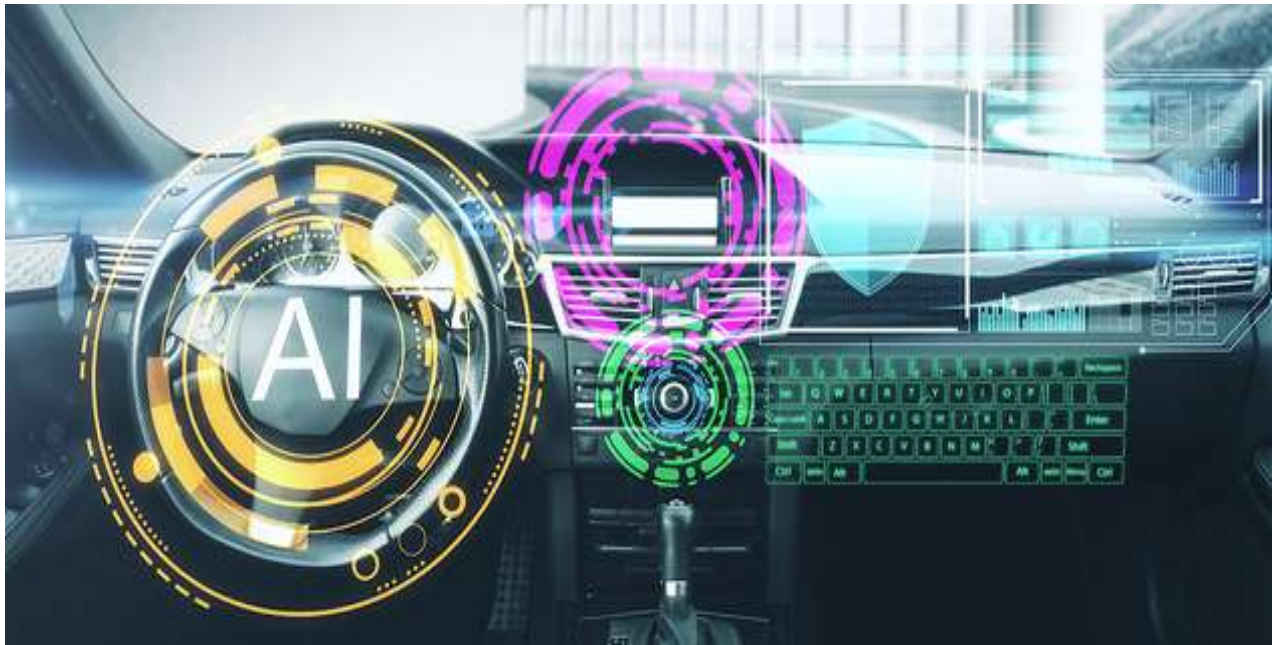
아무리 인식 받은 정보들의 질이 좋고 양이 많아도 그것을 이용할 수 없으면 필요가 없게 됩니다. 그래서 주변상황을 판단하는 것은 매우 중요한 기술입니다. 또한 정확한 판단은 정확한 제어를 할 수 있게 해줍니다. 이러한 판단과 그 후 제어를 할 수 있게 해주는 것이 머신러닝과 딥러닝입니다. 인공지능 훈련을 통해 사물을 단순 인식하는 수준을 넘어서 사람과 같이 사물의 의미를 이해하고 보행자의 다양한 형태와 움직임을 분류하고 파악하며, 차량의 진행 방향, 차도와 인도를 구분하는 등 높은 수준의 판단 능력을 갖게 됩니다. 이런 판단을 바탕으로 차속 조절, 조향, 제동, 운전자 알림 등에 대해 명령받은 대로 작동하여 자율주행을 가능하게끔 합니다.

## 4. MediaPipe와 활용 방법 아이디어

MediaPipe는 구글의 그래프 기반 머신러닝 프레임워크로 오디오, 이미지, 영상을 입력으로 받아 각종 Detection이나 Tracking을 출력할 수 있습니다. 아래의 사진을 보면 MediaPipe를 쉽게 이해할 수 있습니다.



위의 사진과 같이 Mediapipe는 다양한 종류를 제공하지만 Python으로 활용할 수 있는 것은 한정적입니다. 모든 기능을 사용하여 차량지능과 관련해서 알아보고 싶지만 Python을 이용하기 때문에 face detection, pose, holistic을 주로 이용하여 알아보고자 합니다.



### **(1) 자율주행자동차의 인식**

자율주행에 있어서 사람을 인식하는 것이 제일 중요하다고 생각합니다. 그래서 카메라를 통해 사람을 인식하는 것에 대해 알아보았습니다.

### **(2) Mediapipe를 이용한 차량내부 탑승자들의 상태를 확인**

100% 자율주행을 믿기에는 아직 어렵기 때문에 최소한운전석에 차량을 제어할 수 있는 사람은 줄면 안될 것이라고 생각합니다. 그래서 운전자가 조는것과 상태를 파악하는 것에 대해 생각해 봤습니다.

### **(3) Mediapipe hand를 이용한 편리기능 제공**

차량내부카메라를 통해 탑승자들에게 편의기능을 제공하면 좋을 것 같다고 생각했습니다.



## 5. 5. Mediapipe를 활용한 사람 인식

### (1) 문제정의

일단 자율주행의 인식부분에 있어서 가장 중요한 부분은 사람을 인식하는 것이라고 생각합니다. 도로에 무단횡단을 하는 사람이나 멀리서 횡단보도를 건너고 있는 사람들을 향해 오고 있을 때 사람을 인식하여 차량을 제어할 수 있어야합니다.

### (2) 제안방법

카메라를 통해 환경을 인식하고 face detection, pose, holistic을 이용하여 사람을 인식합니다.

### (3) 구현결과와 보완점



왼쪽의 사진은 원본사진을 Holistic을 이용해서 오른쪽은 pose를 이용하여 학습시킨 데이터입니다. <https://google.github.io/mediapipe/>에 있는 Python colab을 통해 인식한 결과입니다. (Mediapipe colab코드를 그대로 활용)

위의 두 가지 사진을 보면 왼쪽의 여성은 사람으로 인식을 하였지만 오른쪽의 남성을 잘 인식하지 못한 부분이 있었습니다. 이러한 부분은 여러 대의 카메라를 통해 다양한 각도로 인식하고 또한 다른 레이더나 라이더 초음파 센서를 이용하여 여러 사람을 정확하게 인식해야 된다고 생각합니다.

### (4) 예상성과

아직은 인식을 못하는 등 부족한 부분이 있지만 다양한 센서를 활용하면 사람을 더욱 더 잘 인식될거라고 생각합니다.

### (5) 코드

<https://google.github.io/mediapipe/> 의 colab코드 사용

## 6. Mediapipe를 이용한 차량내부 탑승자들의 상태를 확인

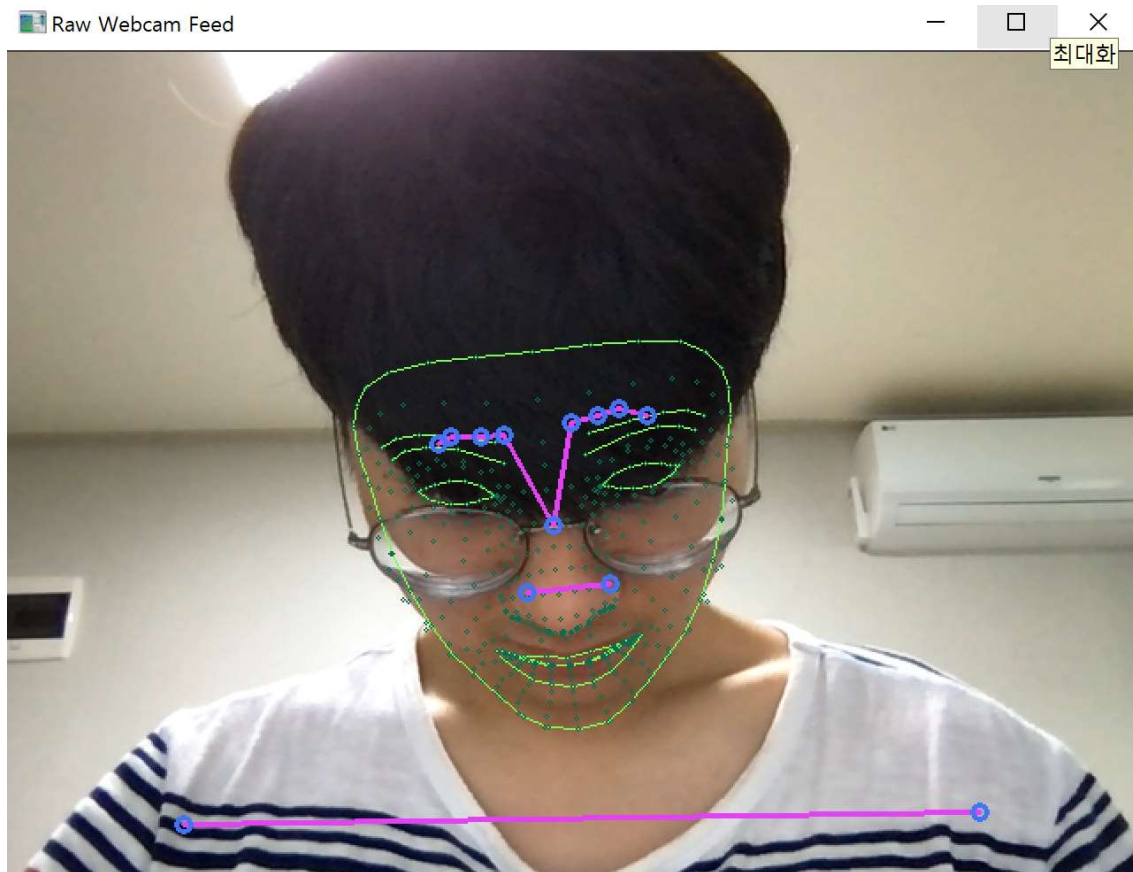
### (1) 문제정의

100% 자율주행을 믿기에는 아직 어렵기 때문에 최소한운전석에 차량을 제어할 수 있는 사람은 졸면 안될 것이라고 생각합니다. 그래서 운전자가 조는것과 상태를 파악하는 것에 대해 알아보았습니다.

### (2) 제안방법

카메라로 받아들인 데이터로 Mediapipe를 통해 사람이 졸고 있는 모습을 인식하고 네비게이션의 디스플레이와 오디오를 통해 운전자에게 경고메세지를 줍니다.

### (3) 구현결과



보통 사람이 졸고 있으면 위의 사진처럼 고개가 아래로 내려가거나 쉽게 움직입니다. 평상시 전방을 주시하거나 다른 행동을 하고 있을 때에는 사람의 머리가 졸고 있을 때에 비해 비교적 긴 시간 동안 가만히 있습니다.

그래서 카메라로 인식 받은 데이터를 통해 고개의 움직임이 심하다, 즉 짧은 시간동안 계속변화가 있고, 또한 고개가 아래로 내려간 시간이 길 때를 네비게이션의 디스플레이를 통해 운전자에게 경고를 줍니다.

#### (4) 예상성과

운전자가 보이게끔 카메라를 설치하여 mediapipe를 통해 운전자가 졸거나 의식이 없는 상태를 인식해 네비게이션에 경고메세지를 주고 1분이내에 반응이 없다면 자동으로 신고하게 하는 기능이 있게 된다면 더욱 더 안전한 자율주행자동차를 만드는데 도움을 줄 것이라고 생각합니다.

#### (5) 코드

<https://github.com/chans98/mediapipe.git> 의 mediapipe1.ipynb

Mediapipe opencv를 다운하고 웹 캠을 설정하고 킵니다.

```
In [1]: !pip install mediapipe opencv-python
Requirement already satisfied: mediapipe in c:\programdata\anaconda3\lib\site-packages (0.8.4)
Requirement already satisfied: opencv-python in c:\programdata\anaconda3\lib\site-packages (4.5.1.48)
Requirement already satisfied: opencv-contrib-python in c:\programdata\anaconda3\lib\site-packages (from mediapipe) (4.5.1.48)
Requirement already satisfied: attrs>=19.1.0 in c:\programdata\anaconda3\lib\site-packages (from mediapipe) (20.3.0)
Requirement already satisfied: protobuf>=3.11.4 in c:\programdata\anaconda3\lib\site-packages (from mediapipe) (3.15.6)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from mediapipe) (1.19.2)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from mediapipe) (1.15.0)
Requirement already satisfied: wheel in c:\programdata\anaconda3\lib\site-packages (from mediapipe) (0.35.1)
Requirement already satisfied: absl-py in c:\programdata\anaconda3\lib\site-packages (from mediapipe) (0.12.0)

In [2]: import mediapipe as mp
import cv2

In [3]: mp_drawing = mp.solutions.drawing_utils
mp_holistic = mp.solutions.holistic

In [4]: cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    cv2.imshow('Raw Webcam Feed', frame)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

아래의 코드는 신체의 특징점을 나타낸 코드로 각각의 라인을 주석 처리하게 되면 예를 들어서 얼굴을 제외한 손만 특징점을 잡아주거나 얼굴만 잡아주거나 하는 등 다양하게 출력을 낼 수 있습니다.

KeyboardInterrupt:

```
cap.release()
cv2.destroyAllWindows()

cap = cv2.VideoCapture(0)
# Initiate holistic mode!
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():
        ret, frame = cap.read()

        # Recolor Feed
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        # Make Detections
        results = holistic.process(image)
        # print(results.face_landmarks)

        # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks

        # Recolor image back to BGR for rendering
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Draw face landmarks
        mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS)

        # Right hand
        mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS)

        # Left hand
        mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS)

        # Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS)

        cv2.imshow('Raw Webcam Feed', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

cap.release()
cv2.destroyAllWindows()
```

## 7. Mediapipe hand를 이용한 편리기능

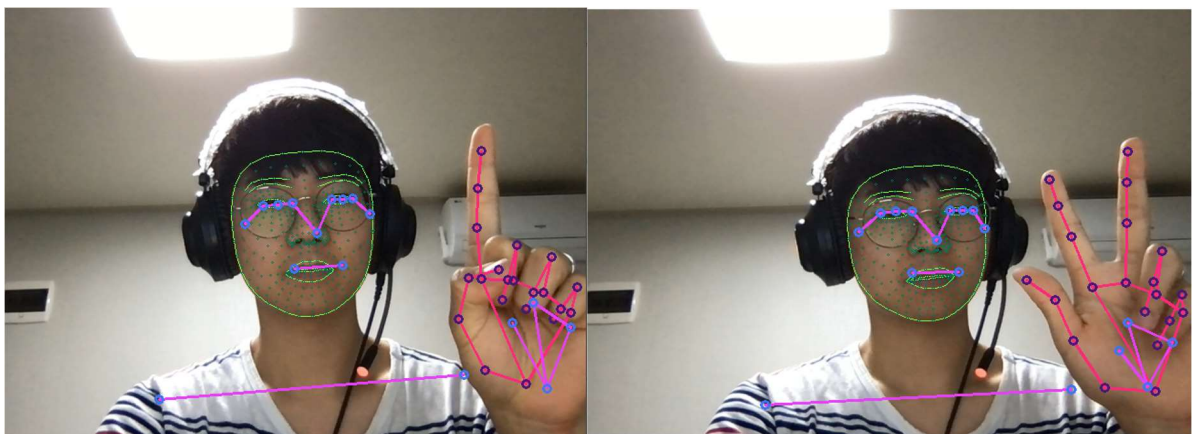
### (1) 문제정의

6번의 차량내부탑승자들의 상태를 체크하는 것과 비슷하게 차량내부카메라를 통해 탑승자들에게 편의기능을 제공하면 좋을 것 같다고 생각했습니다.

### (2) 제안방법

카메라로 받아들이는 데이터를 Mediapipe hand를 이용하여 단축키처럼 활용하여 편리기능을 제공하면 더욱 편리한 자동차가 될 것이라고 생각합니다

### (3) 구현결과

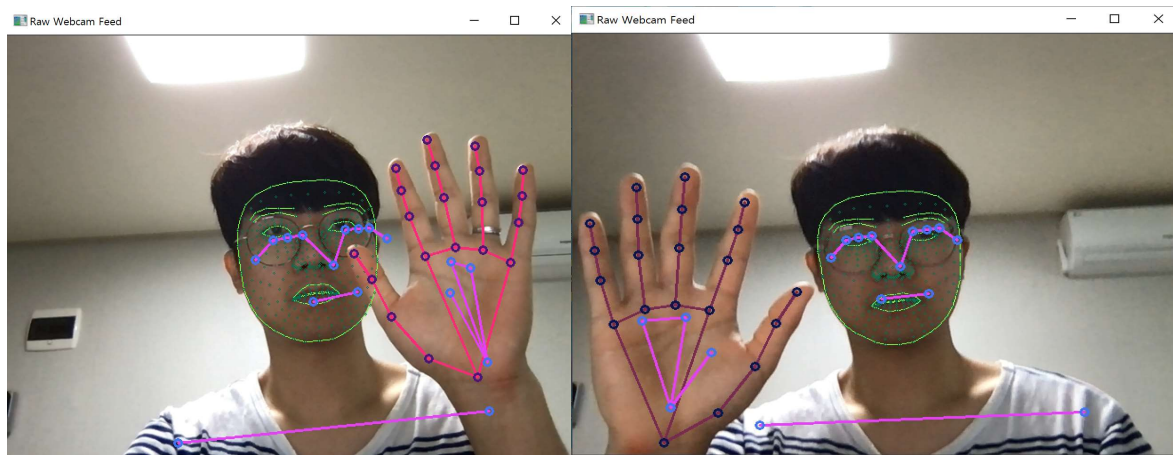


위의 사진은 Mediapipe hand와 웹 캠을 이용하여 손가락을 인식하였습니다. 손가락이 몇 개 펴고 있는지를 인식하는 부분이 중요 했는데 생각했던 거 보다 정확도가 매우 뛰어났습니다. 딜레이도 거의 없고 손가락의 마디마디를 잘 잡아서 결과값을 표현해주었습니다.



#### (4) 예상성과

손가락이 퍼져 있는 개수를 인식하고 단축키를 설정하듯이 예를 들어 손가락 1개를 피고 몇 초 동안 가만히 있으면 라디오가 켜지고 꺼지거나, 손가락을 세 개 피면 자동차의 창문이 내려가고 올라가고 하게 하는 등 디스플레이를 통해 단축행동을 저장해 놓으면 자동차를 이용하는데 더욱 편리하게 사용할 수 있을 것 같습니다. 또한 사람은 왼손과 오른손 둘다 있기 때문에 왼손과 오른손의 색깔을 아래의 사진과 같이 다르게 인식한다면 매우 다양한 단축행동을 설정하여 더욱 편리하게 자동차와 교감하는 느낌을 받으며 사용할 수 있을 것입니다.



#### (5) 응용

아래의 사진과 같이 사람의 감정을 읽을 수 있는 기능을 추가 하게 된다면 기분에 맞는 노래의 재생을 추천 하는 등의 기능도 추가 하면 좋을 것 같다고 생각했습니다.



## (6) 코드

위의 차량내부 탑승자들의 코드와 일치하고 아래의 사람의 감정을 나타내 주는 코드는 아래와 같습니다. (mediapipe2.ipynb에 위치)

### 4. Make Detections with Model

```
with open('body_language.pkl', 'rb') as f:
    model = pickle.load(f)

model

cap = cv2.VideoCapture(0)
# Initiate holistic mode
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():
        ret, frame = cap.read()

        # Recolor Feed
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make Detections
        results = holistic.process(image)
        # print(results, face_landmarks)

        # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks

        # Recolor image back to BGR for rendering
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # 1. Draw face landmarks
        mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                                  mp_drawing.DrawingSpec(color=(80,255,121), thickness=1, circle_radius=1))

        # 2. Right hand
        mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2))

        # 3. Left Hand
        mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2))

        # 4. Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2))

        # Export coordinates
```

```

try:
    # Extract Pose landmarks
    pose = results.pose_landmarks.landmark
    pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in pose]).flatten())

    # Extract Face landmarks
    face = results.face_landmarks.landmark
    face_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in face]).flatten())

    # Concatenate rows
    row = pose_row+face_row

#     # Append class name
#     row.insert(0, class_name)

#     # Export to CSV
#     with open('coords.csv', mode='a', newline='') as f:
#         csv_writer = csv.writer(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
#         csv_writer.writerow(row)

    # Make Detections
    X = pd.DataFrame([row])
    body_language_class = model.predict(X)[0]
    body_language_prob = model.predict_proba(X)[0]
    print(body_language_class, body_language_prob)

    # Grab ear coords
    coords = tuple(np.multiply(
        np.array(
            (results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].x,
             results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].y)),
        [640,480]).astype(int))

    cv2.rectangle(image,
                  (coords[0], coords[1]+5),
                  (coords[0]+len(body_language_class)+20, coords[1]-30),
                  (245, 117, 16), -1)
    cv2.putText(image, body_language_class, coords,
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

    # Get status box
    cv2.rectangle(image, (0,0), (250, 60), (245, 117, 16), -1)

    # Display Class
    cv2.putText(image, 'CLASS'
                , (95,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
    cv2.putText(image, body_language_class.split(' ')[0]
                , (90,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

    # Display Probability
    cv2.putText(image, 'PROB'
                , (15,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
    cv2.putText(image, str(round(body_language_prob[np.argmax(body_language_prob)],2))
                , (10,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

except:
    pass

cv2.imshow('Raw Webcam Feed', image)

if cv2.waitKey(10) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

```

```

tuple(np.multiply(np.array((results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].x,
results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].y)), [640,480]).astype(int))

```

## 8. 결론과 정리

Mediapipe의 face detection , pose , holistic, hands 등을 사용하여 지능형차량과 관련 지어서 생각하고 아이디어를 내봤습니다. 첫번째로는 차량의 사람인식 두번째로는 탑승자의 상태인식 마지막으로 탑승자에게 편리함을 제공하는 것에 대해서 알아보았습니다.

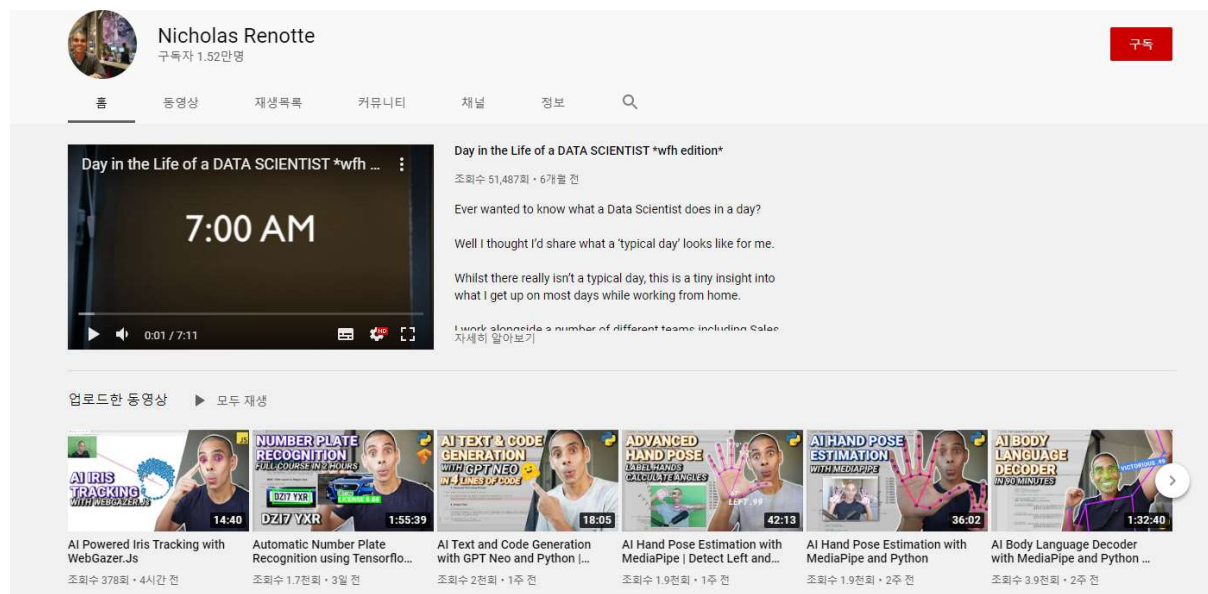
Python으로 할 수 있는 것이 다른 언어에 비해할 수 있는 것이 적었습니다. Objectron으로 사물은 물론 사람들을 포착하는 것을 해보고 싶었지만 네 가지 종류의 물체만 인식 할 수 있어 못했던 것이 아쉽습니다.

## 9. Github 주소와 참고

1. <https://github.com/chans98/mediapipe.git>

2. Youtube 채널 참고

<https://www.youtube.com/channel/UCHXa4OpASJEwrHrLelzw7Yg>



Nicholas Renotte  
구독자 1.52만명

홈 동영상 재생목록 커뮤니티 채널 정보 🔍

**Day in the Life of a DATA SCIENTIST \*wfh edition\***  
조회수 51,487회 · 6개월 전  
Ever wanted to know what a Data Scientist does in a day?  
Well I thought I'd share what a 'typical day' looks like for me.  
Whilst there really isn't a typical day, this is a tiny insight into what I get up on most days while working from home.  
I work alongside a number of different teams including Sales 자세히 알아보기

업로드한 동영상 ▶ 모두 재생

- AI Iris Tracking with WebGazer.js  
조회수 378회 · 4시간 전
- Automatic Number Plate Recognition using TensorFlow  
조회수 1.7천회 · 3일 전
- AI Text and Code Generation with GPT Neo and Python  
조회수 2천회 · 1주 전
- Advanced Hand Pose Estimation with MediaPipe | Detect Left and...  
조회수 1.9천회 · 1주 전
- AI Hand Pose Estimation with MediaPipe and Python  
조회수 1.9천회 · 2주 전
- AI Body Language Decoder with MediaPipe and Python  
조회수 3.9천회 · 2주 전

## 3. 자료 참고

<https://blogs.3ds.com/korea/c-a-s-e-%EC%B9%BC%EB%9F%BC-3->

autonomous%EC%9E%90%EC%9C%A8%EC%A3%BC%ED%96%89%EC%9D%98-%ED%95%B5%EC%8B%AC-%EA%B8%B0%EC%88%A0-%EC%9D%B8%EC%8B%9D-%ED%8C%90%EB%8B%A8-%EC%A0%9C%EC%96%B4/#