# Pose Estimation

Hendrich Ardthian Breshman Panjaitan - 5024221023

December 6, 2024

## 1 Introduction

*Pose estimation* adalah program yang digunakan untuk mendeteksi bagian tubuh manusia seperti tangan dan pose manusia. Pada project ini, saya mengaplikasikan *pose estimation* pada tangan untuk membuat *sign language detection*..

## 2 Source Code

### 2.1 Creating Imageset

Pertama-tama kita membuat *dataset* yang berisi 26 folder berisi 100 gambar yang berisi gambar dari *sign alphabet language* dan menyimpan dataset tersebut di folder data

**Code:**

```
import os
import cv2

DATA_DIR = './data'
if not os.path.exists(DATA_DIR):
    os.makedirs(DATA_DIR)

number_of_classes = 26
dataset_size = 100

cap = cv2.VideoCapture(0)
for j in range(number_of_classes):
    if not os.path.exists(os.path.join(DATA_DIR, str(j))):
        os.makedirs(os.path.join(DATA_DIR, str(j)))

    print('Collecting data for class {}'.format(j))

    done = False
    while True:
        ret, frame = cap.read()
        cv2.putText(frame, 'Ready? Press "Q" ! :)', (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255
                    cv2.LINE_AA)
        cv2.imshow('frame', frame)
        if cv2.waitKey(25) == ord('q'):
            break

    counter = 0
    while counter < dataset_size:
        ret, frame = cap.read()
        cv2.imshow('frame', frame)
```

```
        cv2.waitKey(25)
        cv2.imwrite(os.path.join(DATA_DIR, str(j), '{}.jpg'.format(counter)), frame)

        counter += 1

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

## 2.2 Collect Imageset to Dataset

Setelah kita membuat imageset, kita menggunakan mediapipe untuk mendeteksi tangan pada 26 folder *sign alphabet language* dan kemudian data deteksi tangan tersebut disatukan menjadi file dengan *extension.pickle*

**Code:**

```
import os
import pickle

import mediapipe as mp
import cv2

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

DATA_DIR = './data'

data = []
labels = []

for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []
        x_ = []
        y_ = []

        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        # Proses gambar menggunakan MediaPipe
        results = hands.process(img_rgb)

        # Jika ada tangan yang terdeteksi
        if results.multi_hand_landmarks:
            print(f"Hand detected in: {img_path}, {dir_}")
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y

                    x_.append(x)
                    y_.append(y)
```

```
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    data_aux.append(x - min(x_))
                    data_aux.append(y - min(y_))


                data.append(data_aux)
                labels.append(dir_)

# Simpan data dan label ke file pickle
with open('data.pickle', 'wb') as f:
    pickle.dump({'data': data, 'labels': labels}, f)
```

### 2.3 Training and Making Model

Setelah menyatukan hasil gambar 26 folder *sign alphabet language*, kita membuat model dengan *extension.p* dan train model dengan *data.pickle* yang telah dibuat. Model menggunakan *Random Forest Classifier* dan juga *test-train-split* dari library *scikit-learn* untuk melatih model ini. Random Forest Classifier adalah algoritma yang menghasilkan banyak decision tree yang digunakan untuk mencegah *overfitting*. Penggunaan *Random Forest Classifier* digunakan untuk memecahkan permasalahan *regression* dan *classification* , sedangkan *train-test-split* digunakan untuk membagi data menjadi *train data* and *test data*.

**Code:**

```
import pickle

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np


data_dict = pickle.load(open('./data.pickle', 'rb'))

data = np.asarray(data_dict['data'])
labels = np.asarray(data_dict['labels'])

x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, shuffle=True, strati

model = RandomForestClassifier()

model.fit(x_train, y_train)

y_predict = model.predict(x_test)

score = accuracy_score(y_predict, y_test)

print('{}% of samples were classified correctly !'.format(score * 100))

f = open('model.p', 'wb')
pickle.dump({'model': model}, f)
f.close()
```

## 2.4 Import Model on OpenCV to Detect Sign Alphabet Language

Setelah kita melatih dan membuat model dengan *Random Forest Classifier* dan *Train-Test-Split*, kita mengimpor model ke file *OpenCV*. File *OpenCV* ini menerima input tangan yang ditangkap camera laptop, kemudian dengan *mediapipe*, input tangan tersebut akan memiliki *landmark* atau kerangka. kemudian model tersebut memprediksi kerangka tangan tersebut dan memberikan prediksi *sign alphabet language* sesuai dengan gesture tangan dan menampilkan sentence pada output camera

**Code:**

```
import pickle
import string
import cv2
import mediapipe as mp
import numpy as np
import time

model_dict = pickle.load(open('./model.p', 'rb'))
model = model_dict['model']

camera = cv2.VideoCapture(0)

camera.set(cv2.CAP_PROP_FRAME_WIDTH, 720)
camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

labels_dict = {i: letter for i, letter in enumerate(string.ascii_uppercase)}

sentence = ""
last_added_time = time.time()
last_prediction_time = time.time()
reset_sentence = 3
while True:

    data_aux = []
    x_ = []
    y_ = []

    ret, frame = camera.read()

    H, W, _ = frame.shape

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = hands.process(frame_rgb)
    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(
                frame,
                hand_landmarks,
                mp_hands.HAND_CONNECTIONS,
                mp_drawing_styles.get_default_hand_landmarks_style(),
                mp_drawing_styles.get_default_hand_connections_style())
```

```
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y

                    x_.append(x)
                    y_.append(y)

                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    data_aux.append(x - min(x_))
                    data_aux.append(y - min(y_))

            x1 = int(min(x_) * W) - 10
            y1 = int(min(y_) * H) - 10

            x2 = int(max(x_) * W) - 10
            y2 = int(max(y_) * H) - 10

            prediction = model.predict([np.asarray(data_aux)])

            predicted_character = labels_dict[int(prediction[0])]

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)

            cv2.putText(frame, predicted_character, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0,
                        cv2.LINE_AA)

            if time.time() - last_added_time > 2:
                sentence += predicted_character
                last_added_time = time.time()
                last_prediction_time = time.time()

        cv2.putText(frame, f"Sentence = {sentence}", (20,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,0,0), 2,
        if time.time() - last_prediction_time > reset_sentence and sentence != "":
            sentence = ""
            last_prediction_time = time.time()

        if cv2.waitKey(1) & 0xFF == ord('r'):
            sentence = ""
        elif cv2.waitKey(1) & 0xFF == 8:
            sentence = sentence[:-1]
        cv2.imshow('frame', frame)
        cv2.waitKey(1)

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

# 3 Link Video Demo

Video Demo Project Sign Alphabet Language
Berikut link video demo project. Pada video demo tersebut, model masih bisa memprediksi sign yang error dikarenakan training yang kurang optimal.