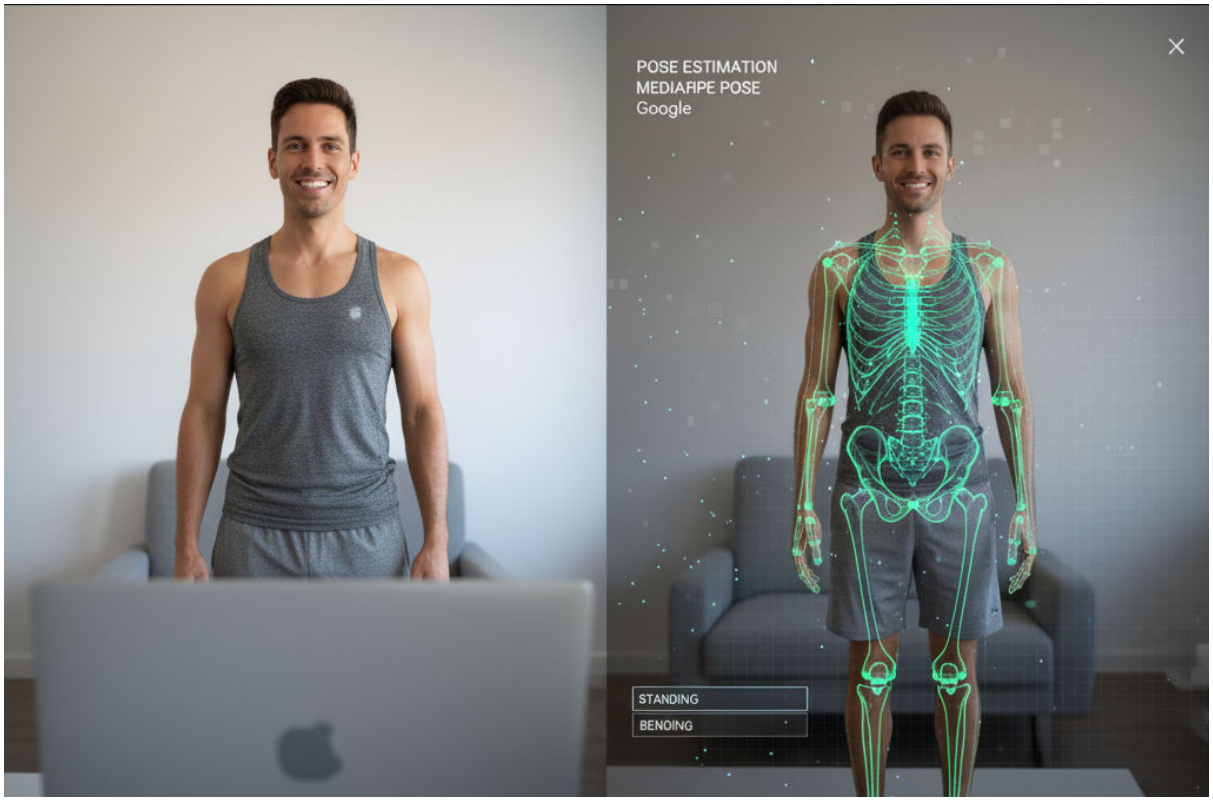
POLITEKNIK NEGERI SEMARANG
JURUSAN TEKNIK ELEKTRO
PROGRAM STUDI STR TEKNOLOGI REKAYASA KOMPUTER

JOBSHEET 04:
TEKNIK ANALISIS POSE & GEOMETRI TUBUH PADA GAMBAR



Dosen:
Ir. Prayitno, S.ST., M.T., Ph.D.

Nama Mahasiswa : _____
NIM Mahasiswa : _____

Tahun Akademik 2025

Jobsheet ini membahas penerapan teknologi *Computer Vision* untuk mengenali dan menganalisis pose serta gestur tubuh manusia menggunakan **MediaPipe** dan **cvzone** pada bahasa pemrograman Python. Melalui enam kegiatan praktikum yang dijalankan secara lokal di komputer, mahasiswa akan mempelajari tahapan mulai dari akuisisi citra kamera, deteksi *landmark* tubuh, wajah, dan tangan, hingga penerapan analisis geometri untuk menghitung sudut, mengenali gestur, serta membuat sistem penghitung aktivitas sederhana. Kegiatan ini bertujuan membekali mahasiswa dengan pemahaman praktis mengenai bagaimana sistem visi komputer dapat digunakan untuk interpretasi gerakan manusia secara *real-time*.

A. Tujuan Instruksional Khusus

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu untuk:

1. Mengoperasikan kamera komputer dan melakukan akuisisi citra secara *real-time* menggunakan Python dan OpenCV.
2. Menerapkan deteksi pose tubuh manusia menggunakan modul *Pose Estimation* dari cvzone berbasis MediaPipe.
3. Menghitung sudut dan rasio geometri tubuh dari data *landmark* yang dihasilkan oleh deteksi pose.
4. Melakukan deteksi wajah serta mengidentifikasi perubahan ekspresi sederhana, seperti kedipan mata, menggunakan *Face Mesh*.
5. Menerapkan deteksi tangan dan mengenali jumlah jari yang terangkat menggunakan modul *Hand Tracking*.
6. Mendesain sistem sederhana untuk mengenali gestur tangan, seperti *Thumbs Up*, *OK*, dan *Rock–Paper–Scissors*, berdasarkan relasi antar *landmark*.
7. Mengembangkan sistem penghitung aktivitas tubuh, seperti *squat* dan *push-up*, menggunakan kombinasi analisis pose, geometri, dan logika kondisi.
8. Mengintegrasikan hasil deteksi *pose*, *face*, dan *hand landmark* ke dalam aplikasi sederhana berbasis Python yang dapat dijalankan secara lokal.
9. Menyusun laporan praktikum yang menjelaskan tahapan implementasi, hasil pengujian, serta analisis fungsional dari sistem yang telah dibuat.

B. Alat dan Bahan

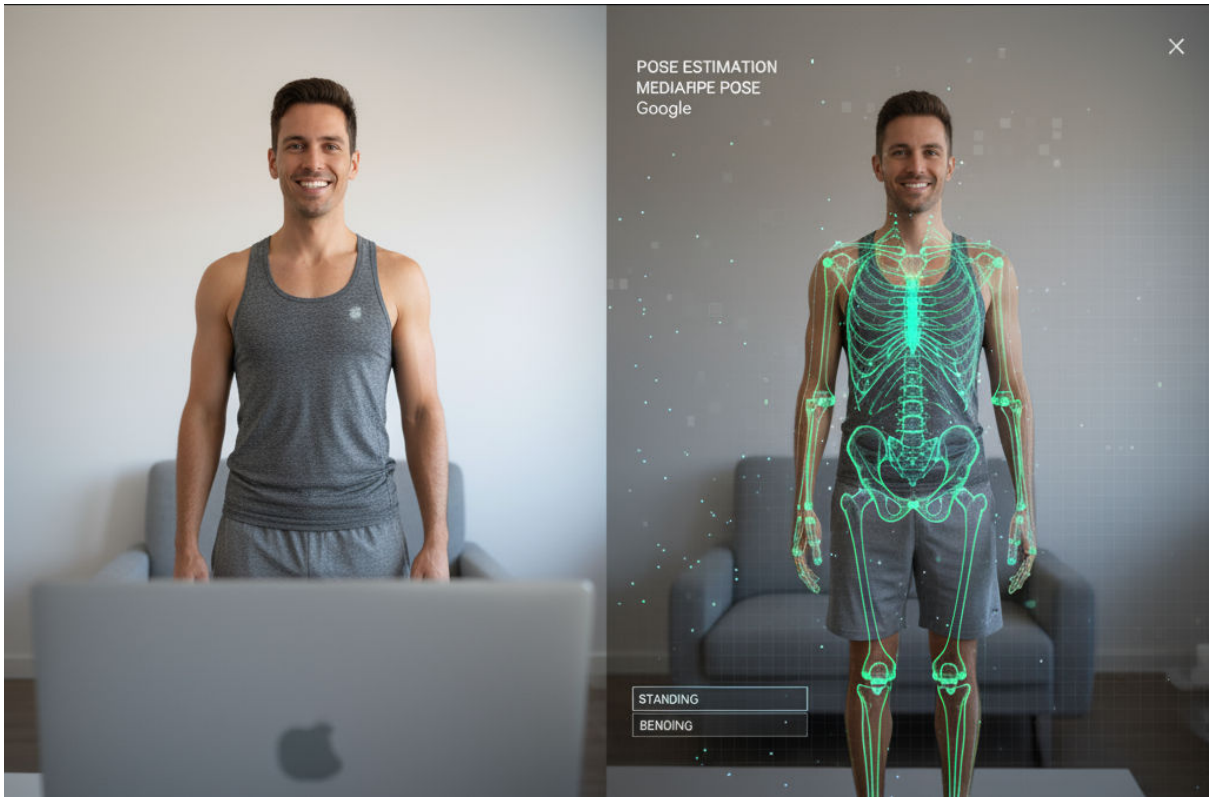
Untuk melaksanakan enam praktikum pada topik Pose Estimation dan Gesture Recognition berbasis cvzone dan MediaPipe, diperlukan perangkat keras serta perangkat lunak yang mendukung komputasi visual secara real-time. Tabel berikut menjelaskan secara rinci alat dan bahan yang digunakan selama kegiatan praktikum.

No.	Jenis	Nama / Spesifikasi	Keterangan / Fungsi
1	**Perangkat Keras**		
1.1	Laptop / Komputer	Prosesor minimal Intel Core i3 / AMD Ryzen 3, RAM ≥ 8 GB, OS Windows 10/11, Linux, atau macOS	Sebagai perangkat utama untuk menjalankan program dan pengolahan citra.
1.2	Kamera / Webcam	Resolusi minimal 720p (HD), internal atau eksternal (USB)	Digunakan untuk akuisisi citra dan video secara *real-time*.
1.3	Perangkat Input	Keyboard dan mouse	Untuk memberikan perintah serta interaksi dengan program Python.
1.4	Perangkat Penyimpanan	Kapasitas kosong minimal 5 GB	Menyimpan file kode, hasil tangkapan gambar, dan laporan praktikum.
2	**Perangkat Lunak**		

No.	Jenis	Nama / Spesifikasi	Keterangan / Fungsi
2.1	Python	Versi 3.10 atau lebih baru	Bahasa pemrograman utama untuk implementasi sistem visi komputer.
2.2	Visual Studio Code / PyCharm	IDE untuk pengembangan Python	Lingkungan pengembangan terintegrasi untuk menulis dan menjalankan kode.
2.3	OpenCV (`opencv-python`)	Library pengolahan citra dan video	Digunakan untuk membaca, menampilkan, dan memproses *frame* kamera.
2.4	cvzone	Antarmuka pembungkus (*wrapper*) untuk OpenCV dan MediaPipe	Mempermudah implementasi deteksi pose, wajah, dan tangan.
2.5	MediaPipe	Framework *machine learning* untuk deteksi *landmark* tubuh	Menyediakan model deteksi pose, wajah, dan tangan secara *real-time*.
2.6	NumPy	Library perhitungan matematis	Digunakan untuk menghitung jarak, sudut, dan rasio antar titik *landmark*.
2.7	Terminal / Command Prompt	Lingkungan perintah sistem	Digunakan untuk instalasi paket dan eksekusi program.
3	**Data dan Lingkungan Uji**		
3.1	Data video (opsional)	File `.mp4` atau `.avi`	Alternatif sumber data apabila kamera tidak tersedia.
3.2	Ruang uji praktikum	Area dengan pencahayaan cukup dan latar belakang sederhana	Menjamin hasil deteksi stabil dan bebas gangguan bayangan.

C. Dasar Teori

Bayangkan suatu pagi Anda berdiri di depan kamera laptop, dan komputer di hadapan Anda bukan sekadar melihat “orang” di layar, tetapi benar-benar memahami bahwa yang ia lihat adalah tubuh manusia dengan struktur yang kompleks: kepala, bahu, siku, lutut, hingga pergelangan kaki. Ia bisa mengenali kapan Anda berdiri, menunduk, atau melompat. Kemampuan ini lahir dari konsep **pose estimation**, sebuah algoritma yang memungkinkan komputer menafsirkan posisi tubuh manusia dari gambar atau video. Salah satu framework yang paling populer dan efisien dalam bidang ini adalah **MediaPipe Pose**, buatan Google, yang dapat mendeteksi dan melacak gerakan tubuh manusia secara real-time bahkan dengan perangkat keras sederhana seperti webcam atau kamera ponsel.



MediaPipe bekerja layaknya seorang pelukis digital yang menggambar ulang kerangka tubuh manusia dengan menyambungkan 33 titik koordinat penting. Setiap titik L_i direpresentasikan sebagai:

$$L_i = (x_i, y_i, z_i, v_i), \qquad i = 1,2, \dots ,33$$

di mana x_i, y_i, z_i adalah koordinat posisi tubuh dalam ruang ter-normalisasi, dan $v_i \in [0,1]$ adalah tingkat kepercayaan (*visibility*) terhadap deteksi titik tersebut. Dengan menghubungkan antar titik — misalnya antara bahu, siku, dan pergelangan tangan — MediaPipe membentuk **pose skeleton**, yaitu model geometri tubuh yang dapat bergerak mengikuti setiap gerakan manusia. Struktur ini memungkinkan sistem untuk menganalisis postur, menghitung sudut, dan mendeteksi perubahan gerakan secara matematis.

Secara teoretis, masalah *pose estimation* dapat diformalkan sebagai pemetaan fungsi dari gambar ke posisi titik-titik tubuh. Kita definisikan fungsi:

$$f_\theta : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{N \times D}$$

Dengan H, W adalah dimensi citra (tinggi dan lebar), 3 menunjukkan kanal RGB, N jumlah landmark (33 untuk MediaPipe), dan D dimensi spasial (biasanya 2D atau 3D). Tujuan dari model dengan parameter θ ini adalah memprediksi koordinat landmark:

$$\hat{y} = f_\theta(x_i) = \left[\hat{L}_1, \hat{L}_2, \dots, \hat{L}_N\right]$$

di mana setiap $\hat{L}_j = (\hat{x}_j, \hat{y}_j, \hat{z}_j)$.

Masalah ini termasuk dalam kategori **regresi multivariabel kontinu**, karena model tidak mengklasifikasikan citra ke label diskrit (seperti "orang duduk" atau "berdiri"), melainkan memperkirakan nilai koordinat numerik kontinu. Proses pelatihannya dilakukan dengan meminimalkan fungsi kerugian (*loss function*) berbasis jarak, misalnya **Mean Squared Error (MSE)**:

$$L(\theta) = \frac{1}{N}\sum_{j=1}^{N} v_j \,.\, \left\|\widehat{L_j} - L_j\right\|^2$$

di mana v_j memberikan bobot lebih besar pada landmark yang terlihat jelas. Dengan kata lain, model belajar agar posisi prediksi $\hat{L}_j$ sedekat mungkin dengan posisi sebenarnya L_j.

Beberapa model lain seperti OpenPose atau HRNet tidak langsung memprediksi koordinat, melainkan menghasilkan **peta probabilitas (heatmap)** untuk setiap titik tubuh:

$$f_\theta(x) = \{H_1, H_2, \dots, H_N\}, H_j \in [0,1]^{H' \times W'}$$

di mana $H_j(u, v)$ menyatakan probabilitas bahwa landmark ke-j berada pada posisi(u, v). Dari heatmap ini, posisi landmark diperoleh dengan:

$$\left(\hat{x}_j, \hat{y}_j\right) = arg\max_{(u,v)} H_j(u, v)$$

atau dengan pendekatan *soft-argmax*:

$$\left(\hat{x}_j, \hat{y}_j\right) = \sum_{u,v} (u, v) \;.\, H_j(u, v)$$

Pendekatan ini masih tergolong **regresi probabilistik**, namun dilakukan pada ruang spasial dua dimensi.

Selain koordinat, MediaPipe juga memperkirakan **visibility score** $c_j = g_\theta(x)$, yang secara konseptual dapat dianggap sebagai *binary classification* untuk memutuskan apakah titik tubuh tampak atau tertutup.

Begitu posisi titik-titik tubuh diketahui, langkah berikutnya adalah menghubungkan antar titik untuk menghitung **vektor** dan **sudut sendi**. Misalnya, sudut di lutut kiri dapat dihitung dari tiga titik: pinggul (A), lutut (B), dan pergelangan kaki (C). Dua vektor yang terbentuk adalah:

$$BA = A - B, BC = C - B$$

dan sudut di lutut (θ) dihitung menggunakan rumus *dot product*:

$$\theta = \cos^{-1}\left(\frac{BA \,.\, BC}{\|BA\| \,.\, \|BC\|}\right)$$

Nilai θ inilah yang sering menjadi dasar dalam aplikasi-aplikasi praktis seperti *squat counter*, *push-up tracker*, atau *posture monitor*. Untuk menjaga kestabilan nilai, sering digunakan teknik *moving average*:

$$\bar{\theta}_t = \frac{1}{k}\sum_{i=0}^{k-1} \theta_{t-i}$$

sehingga perubahan mendadak akibat noise dapat teredam. Selain itu, perbedaan ukuran tubuh antar individu diatasi dengan **normalisasi skala**, misalnya menggunakan rasio jarak bahu–pinggul sebagai pembagi panjang segmen tubuh lain.

Teknologi ini telah diaplikasikan secara luas di berbagai bidang. Dalam dunia medis, **Park et al. (2024)** menunjukkan pada studi *"Biomechanical Posture Analysis in Healthy Adults*

using MediaPipe Pose" (PMC11086111) bahwa model ini efektif untuk **rehabilitasi pasien** tanpa memerlukan sensor fisik tambahan. Di bidang **analisis gait**, **Kyeong et al. (2023)** berhasil menggunakan MediaPipe untuk mendeteksi fase *heel-strike* dan *toe-off* dari cara berjalan pasien (PMC10384445). Dalam **ergonomi industri**, riset terbaru di *ACM Digital Library (2024)* memanfaatkan estimasi pose untuk mengukur risiko cedera pekerja dari sudut punggung dan lutut. Bahkan dalam kehidupan sehari-hari, aplikasi seperti *PoseTrack* telah digunakan untuk **memantau postur duduk** di depan komputer menggunakan Raspberry Pi, sementara penelitian *"Validation of the Google MediaPipe Hand"* (ScienceDirect, 2024) menegaskan akurasi tinggi deteksi gerakan jari pada rehabilitasi tangan. Semua penelitian ini menunjukkan bahwa *pose estimation* bukan sekadar eksperimen laboratorium, tetapi telah menjadi teknologi nyata yang menjembatani dunia digital dan tubuh manusia.

Dengan demikian, **pose estimation** pada dasarnya adalah **masalah regresi spasial multivariabel**, dengan tambahan elemen klasifikasi untuk menilai kehadiran atau kepercayaan tiap titik tubuh. Model seperti MediaPipe menggabungkan konsep *computer vision*, *geometri*, dan *machine learning* agar komputer dapat "memahami" tubuh manusia secara matematis — bukan hanya mengenali siapa Anda, tetapi juga **bagaimana Anda bergerak**. Dalam konteks yang lebih sederhana:

- Jika *image classification* bertanya **"gambar ini menggambarkan apa?"**,
- maka *pose estimation* bertanya **"di mana bagian-bagian tubuh manusia berada?"** Jawaban untuk pertanyaan kedua tidak berupa label tunggal, melainkan **sekumpulan koordinat kontinu**.

Itulah sebabnya pendekatan yang digunakan bukan *softmax classifier*, tetapi *regression head* yang mengestimasi posisi numerik secara langsung dari representasi fitur citra.

Aspek	**Jenis Masalah**	**Luaran**	**Mengukur Performa**
Pose Estimation (2D/3D)	**Regresi multivariabel kontinu**	Koordinat landmark (x, y, z)	Mean Squared Error (MSE), L1, Smooth-L1
Heatmap-based Pose	**Probabilistic regression / spatial density estimation**	Peta probabilitas setiap landmark	Cross-entropy, KL-divergence, Gaussian loss
Visibility / Confidence	**Binary classification (optional)**	Skor kehadiran titik	Binary cross entropy (BCE)

D. Langkah Praktikum

Persiapan Lingkungan (sekali saja)

1. **Python 3.10–3.12** terpasang.
2. (Opsional tapi dianjurkan) buat venv:

```
python -m venv .venv
# Windows
.venv\Scripts\activate
# macOS/Linux
source .venv/bin/activate
```

3. Instal paket:

```
pip install mediapipe==0.10.14 opencv-python numpy
```

4. Jika kamera tidak muncul, coba ganti cv2.VideoCapture(0) menjadi 1 atau 2. Tombol umum: tekan **q** untuk keluar jendela.

Praktikum D1 — Inisialisasi Kamera dan Akuisisi Citra

Tujuan:
Mahasiswa mampu menginisialisasi kamera dan menampilkan citra secara *real-time* sebagai dasar untuk pengolahan visual selanjutnya.
Deskripsi:
Praktikum ini merupakan tahap awal dalam mengenal sistem visi komputer. Mahasiswa akan belajar membuka perangkat kamera, menampilkan citra secara berkelanjutan, serta memastikan bahwa perangkat keras dan perangkat lunak dapat berkomunikasi dengan baik. Proses ini menjadi fondasi bagi semua eksperimen berikutnya yang menggunakan aliran video sebagai input utama.
Langkah:
1. Jalankan skrip Python menggunakan pustaka `opencv-python`.
2. Inisialisasi kamera dengan `cv2.VideoCapture(0)`.
3. Baca setiap frame dan tampilkan di jendela *preview*.
4. Ukur *frame per second (FPS)* untuk menilai kelancaran akuisisi video.
5. Tekan tombol **q** untuk menghentikan proses.

Indikator Keberhasilan:
Citra tampil dengan stabil, FPS berada di atas 10, dan proses dapat dihentikan dengan perintah keluar tanpa error.

```
import cv2, time

cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise RuntimeError("Kamera tidak bisa dibuka. Coba index 1/2.")

frames, t0 = 0, time.time()
while True:
    ok, frame = cap.read()
    if not ok: break
    frames += 1
    if time.time() - t0 >= 1.0:
        cv2.setWindowTitle("Preview", f"Preview (FPS ~ {frames})")
        frames, t0 = 0, time.time()
    cv2.imshow("Preview", frame)
```

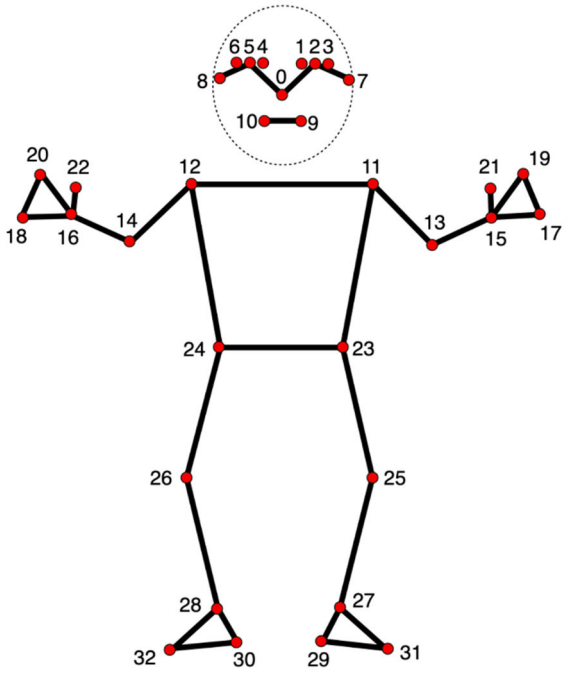
```
    if cv2.waitKey(1) & 0xFF == ord('q'): break

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

Praktikum D2 — Deteksi Pose dan Analisis Sudut Tubuh

Tujuan:
Mahasiswa mampu menerapkan *pose estimation* menggunakan MediaPipe dan menghitung sudut sendi tertentu, misalnya sudut lutut.

Deskripsi:
MediaPipe Pose menyediakan 33 *landmark* tubuh yang dapat digunakan untuk menganalisis posisi dan orientasi manusia. Dalam praktikum ini, mahasiswa akan mengimplementasikan algoritma deteksi pose dan menghitung sudut sendi berdasarkan koordinat tiga titik utama tubuh menggunakan prinsip geometri vektor. Model penanda pose melacak 33 lokasi penanda tubuh, yang mewakili perkiraan lokasi bagian tubuh berikut:




```
0 - nose
1 - left eye (inner)
2 - left eye
3 - left eye (outer)
4 - right eye (inner)
5 - right eye
6 - right eye (outer)
7 - left ear
8 - right ear
9 - mouth (left)
10 - mouth (right)
11 - left shoulder
12 - right shoulder
13 - left elbow
14 - right elbow
15 - left wrist
16 - right wrist
17 - left pinky
18 - right pinky
19 - left index
20 - right index
21 - left thumb
22 - right thumb
23 - left hip
24 - right hip
25 - left knee
26 - right knee
27 - left ankle
28 - right ankle
29 - left heel
30 - right heel
31 - left foot index
32 - right foot index
```

Langkah:

1. Gunakan modul `mediapipe.solutions.pose` untuk mendeteksi pose.
2. Tentukan titik referensi (misalnya pinggul, lutut, dan pergelangan kaki).

3. Hitung sudut di lutut menggunakan rumus *dot product*:

   θ=cos− −1((A−B)·(C−B)∥A−B∥∥C−B∥)\theta = \cos^{-1} \left(\frac{(A-B)\cdot(C-B)}{\|A-B\|\|C-B\|} \right)θ=cos−1(∥A−B∥∥C−B∥(A−B)·(C−B))

4. Tampilkan hasil deteksi kerangka dan nilai sudut di layar secara *real-time*.
5. Uji sistem dengan melakukan gerakan berdiri dan jongkok.

Indikator Keberhasilan:
Kerangka tubuh terdeteksi dengan baik dan nilai sudut berubah sesuai pergerakan.

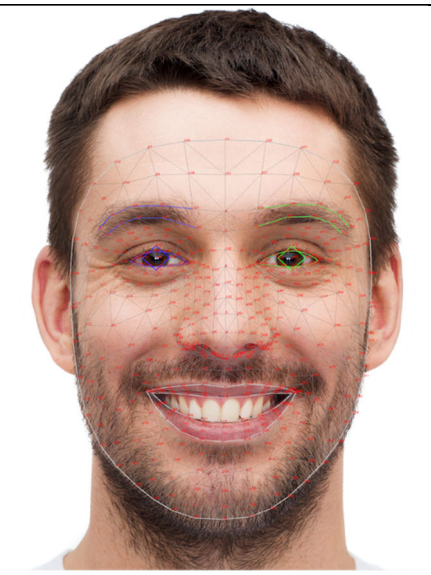
```
01: import cv2, numpy as np
02: from cvzone.PoseModule import PoseDetector
03:
04: cap = cv2.VideoCapture(0)
05: if not cap.isOpened():
06:     raise RuntimeError("Kamera tidak bisa dibuka.")
07:
08: detector = PoseDetector(staticMode=False, modelComplexity=1,
09:                         enableSegmentation=False, detectionCon=0.5,
trackCon=0.5)
10:
11: while True:
12:     # Tangkap setiap frame dari webcam
13:     success, img = cap.read()
14:
15:     # Temukan pose manusia dalam frame
16:     img = detector.findPose(img)
17:
18:     # Temukan landmark, bounding box, dan pusat tubuh dalam frame
19:     # Set draw=True untuk menggambar landmark dan bounding box pada
gambar
20:     lmList, bboxInfo = detector.findPosition(img, draw=True,
bboxWithHands=False)
21:
22:     # Periksa apakah ada landmark tubuh yang terdeteksi
23:     if lmList:
24:         # Dapatkan pusat bounding box di sekitar tubuh
25:         center = bboxInfo["center"]
26:
27:         # Gambar lingkaran di pusat bounding box
28:         cv2.circle(img, center, 5, (255, 0, 255), cv2.FILLED)
29:
30:         # Hitung jarak antara landmark 11 dan 15 dan gambarkan pada
gambar
31:         length, img, info = detector.findDistance(lmList[11][0:2],
32:                                                   lmList[15][0:2],
33:                                                   img=img,
34:                                                   color=(255, 0, 0),
35:                                                   scale=10)
36:
37:         # Hitung sudut antara landmark 11, 13, dan 15 dan gambarkan
pada gambar
38:         angle, img = detector.findAngle(lmList[11][0:2],
39:                                         lmList[13][0:2],
40:                                         lmList[15][0:2],
41:                                         img=img,
42:                                         color=(0, 0, 255),
```

```
43:                                         scale=10)
44:
45:         # Periksa apakah sudut mendekati 50 derajat dengan offset 10
46:         isCloseAngle50 = detector.angleCheck(myAngle=angle,
47:                                              targetAngle=50,
48:                                              offset=10)
49:
50:         # Cetak hasil pemeriksaan sudut
51:         print(isCloseAngle50)
52:
53:     cv2.imshow("Pose + Angle ", img)
54:     if cv2.waitKey(1) & 0xFF == ord('q'): break
55:
56: cap.release()
57: cv2.destroyAllWindows()
58:
```

Praktikum D3 — Deteksi Wajah dan Analisis Kedipan Mata

Tujuan:
Mahasiswa mampu mendeteksi wajah menggunakan MediaPipe Face Mesh dan mengukur perubahan jarak antar *landmark* mata untuk mendeteksi kedipan.

Deskripsi:
Face Mesh memetakan 468 titik pada wajah, termasuk titik di sekitar mata. Praktikum ini memperkenalkan konsep *Eye Aspect Ratio (EAR)*, yaitu rasio antara jarak vertikal dan horizontal mata yang digunakan untuk menentukan apakah mata dalam kondisi terbuka atau tertutup.



Alis (Brow)
- 1 - **browDownLe** (Alis Kiri Bawah)
- 2 - **browDownRight** (Alis Kanan Bawah)
- 3 - **browInnerUp** (Alis Tengah Naik)
- 4 - **browOuterUpLe** (Alis Luar Kiri Naik)
- 5 - **browOuterUpRight** (Alis Luar Kanan Naik)

Pipi (Cheek)
- 6 - **cheekPu** (Pipi Menggembung)
- 7 - **cheekSquintLe** (Pipi Kiri Menyipit)
- 8 - **cheekSquintRight** (Pipi Kanan Menyipit)

Mata (Eye)
- 9 - **eyeBlinkLe** (Mata Kiri Berkedip)
- 10 - **eyeBlinkRight** (Mata Kanan Berkedip)
- 11 - **eyeLookDownLe** (Mata Kiri Melihat Bawah)
- 12 - **eyeLookDownRight** (Mata Kanan Melihat Bawah)
- 13 - **eyeLookInLe** (Mata Kiri Melirik ke Dalam)
- 14 - **eyeLookInRight** (Mata Kanan Melirik ke Dalam)
- 15 - **eyeLookOutLe** (Mata Kiri Melirik ke Luar)
- 16 - **eyeLookOutRight** (Mata Kanan Melirik ke Luar)
- 17 - **eyeLookUpLe** (Mata Kiri Melihat Atas)
- 18 - **eyeLookUpRight** (Mata Kanan Melihat Atas)
- 19 - **eyeSquintLe** (Mata Kiri Menyipit)
- 20 - **eyeSquintRight** (Mata Kanan Menyipit)
- 21 - **eyeWideLe** (Mata Kiri Melebar)
- 22 - **eyeWideRight** (Mata Kanan Melebar)

Rahang (Jaw)
- 23 - **jawForward** (Rahang Maju)
- 24 - **jawLe** (Rahang Geser Kiri)
- 25 - **jawOpen** (Rahang Terbuka)
- 26 - **jawRight** (Rahang Geser Kanan)

Mulut (Mouth)
- 27 - **mouthClose** (Mulut Tertutup)
- 28 - **mouthDimpleLe** (Lesung Pipi Kiri)
- 29 - **mouthDimpleRight** (Lesung Pipi Kanan)
- 30 - **mouthFrownLe** (Cemberut Kiri)
- 31 - **mouthFrownRight** (Cemberut Kanan)
- 32 - **mouthFunnel** (Mulut Bentuk Corong)
- 33 - **mouthLe** (Mulut Geser Kiri)
- 34 - **mouthLowerDownLe** (Bibir Bawah Kiri Turun)
- 35 - **mouthLowerDownRight** (Bibir Bawah Kanan Turun)

	• 36 - **mouthPressLe** (Bibir Kiri Menekan) • 37 - **mouthPressRight** (Bibir Kanan Menekan) • 38 - **mouthPucker** (Mulut Monyong) • 39 - **mouthRight** (Mulut Geser Kanan) • 40 - **mouthRollLower** (Bibir Bawah Menggulung) • 41 - **mouthRollUpper** (Bibir Atas Menggulung) • 42 - **mouthShrugLower** (Bibir Bawah Mengangkat Bahu) • 43 - **mouthShrugUpper** (Bibir Atas Mengangkat Bahu) • 44 - **mouthSmileLe** (Senyum Kiri) • 45 - **mouthSmileRight** (Senyum Kanan) • 46 - **mouthStretchLe** (Mulut Kiri Meregang) • 47 - **mouthStretchRight** (Mulut Kanan Meregang) • 48 - **mouthUpperUpLe** (Bibir Atas Kiri Naik) • 49 - **mouthUpperUpRight** (Bibir Atas Kanan Naik) **Hidung dan Lidah (Nose & Tongue)** • 50 - **noseSneerLe** (Hidung Kiri Mengerut/Mencibir) • 51 - **noseSneerRight** (Hidung Kanan Mengerut/Mencibir) • 52 - **tongueOut** (Lidah Keluar)

Langkah:

1. Gunakan modul `mediapipe.solutions.face_mesh`.
2. Ambil koordinat beberapa *landmark* mata (misalnya 33, 133, 145, dan 159).
3. Hitung *Eye Aspect Ratio (EAR)*:

   EAR=dvertikaldhorizontalEAR = \frac{d_{vertikal}}{d_{horizontal}}EAR=dhorizontaldvertikal

4. Tampilkan nilai EAR di layar; nilai menurun saat mata tertutup.
5. Implementasikan ambang batas untuk mendeteksi kedipan otomatis.

Indikator Keberhasilan:
Sistem mampu menampilkan wajah dengan titik *landmark* yang stabil dan mendeteksi perubahan nilai EAR secara akurat saat mata berkedip.

```
01: import cv2, numpy as np
02: from cvzone.FaceMeshModule import FaceMeshDetector
03:
04: # Indeks mata kiri (contoh): vertikal (159,145), horizontal (33,133)
05: L_TOP, L_BOTTOM, L_LEFT, L_RIGHT = 159, 145, 33, 133
06:
07: def dist(p1, p2): return np.linalg.norm(np.array(p1) - np.array(p2))
08:
09: cap = cv2.VideoCapture(0)
10: if not cap.isOpened():
11:     raise RuntimeError("Kamera tidak bisa dibuka.")
12:
13: # Inisialisasi objek FaceMeshDetector
14: # staticMode: Jika True, deteksi hanya terjadi sekali; jika False,
setiap frame
```

```
# maxFaces: Jumlah maksimum wajah yang dideteksi
# minDetectionCon: Ambang kepercayaan deteksi minimum
# minTrackCon: Ambang kepercayaan pelacakan minimum
detector = FaceMeshDetector(staticMode=False, maxFaces=2,
minDetectionCon=0.5, minTrackCon=0.5)

# Variabel untuk menghitung kedipan sederhana
blink_count = 0
closed_frames = 0
CLOSED_FRAMES_THRESHOLD = 3   # jumlah frame berturut-turut untuk
dianggap kedipan
EYE_AR_THRESHOLD = 0.20      # ambang Eye Aspect Ratio (EAR) untuk
menilai mata tertutup
is_closed = False

while True:
    ok, img = cap.read()
    if not ok: break
    img, faces = detector.findFaceMesh(img, draw=True)
    if faces:
        face = faces[0]  # list of 468 (x,y)
        v = dist(face[L_TOP], face[L_BOTTOM])
        h = dist(face[L_LEFT], face[L_RIGHT])
        ear = v / (h + 1e-8)
        cv2.putText(img, f"EAR(L): {ear:.3f}", (20,40),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,255), 2)

        # Contoh ambang kedipan sederhana dan logika counter:
        # jika EAR < EYE_AR_THRESHOLD selama CLOSED_FRAMES_THRESHOLD
frame -> hitung kedipan
        if ear < EYE_AR_THRESHOLD:
            closed_frames += 1
            if closed_frames >= CLOSED_FRAMES_THRESHOLD and not
is_closed:
                blink_count += 1
                is_closed = True
        else:
            closed_frames = 0
            is_closed = False

        # Tampilkan jumlah kedipan pada frame
        cv2.putText(img, f"Blink: {blink_count}", (20,70),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,0), 2)

    cv2.imshow("FaceMesh + EAR", img)
    if cv2.waitKey(1) & 0xFF == ord('q'): break

cap.release()
cv2.destroyAllWindows()

```

Praktikum D4 — Deteksi Tangan dan Penghitungan Jumlah Jari

Tujuan:
Mahasiswa mampu mendeteksi tangan manusia dan menghitung jumlah jari yang terangkat menggunakan *landmark* tangan dari MediaPipe.

Deskripsi:
MediaPipe Hands mendeteksi 21 *landmark* pada setiap tangan. Berdasarkan hubungan antara ujung jari (*tip*) dan sendi bawahnya (*PIP*), dapat ditentukan apakah jari tersebut dalam posisi terangkat atau tidak. Pendekatan ini digunakan untuk menghitung jumlah jari terbuka.

Langkah:

1. Gunakan `mediapipe.solutions.hands` untuk memperoleh koordinat *landmark*.
2. Bandingkan posisi *tip* (ujung jari) terhadap *PIP* (sendi bawah) berdasarkan sumbu vertikal (y).
3. Tentukan logika sederhana: jika ytip<ypipy_{tip} < y_{pip}ytip<ypip, maka jari dianggap terangkat.
4. Tampilkan jumlah jari yang terangkat di layar.
5. Uji sistem dengan berbagai kombinasi jumlah jari.

Indikator Keberhasilan:
Jumlah jari yang terangkat terdeteksi dengan benar (akurasi >80%) pada berbagai kondisi pencahayaan.

```
01: import cv2
02: from cvzone.HandTrackingModule import HandDetector
03:
04: cap = cv2.VideoCapture(0)
05: if not cap.isOpened():
06:     raise RuntimeError("Kamera tidak bisa dibuka.")
07:
08: detector = HandDetector(staticMode=False, maxHands=1,
modelComplexity=1,
09:                         detectionCon=0.5, minTrackCon=0.5)
10:
11: while True:
12:     ok, img = cap.read()
13:     if not ok: break
14:     hands, img = detector.findHands(img, draw=True, flipType=True)  #
flipType untuk mirror UI
15:     if hands:
16:         hand = hands[0]  # dict berisi "lmList", "bbox", dll.
17:         fingers = detector.fingersUp(hand)  # list panjang 5 berisi 0/1
18:         count = sum(fingers)
19:         cv2.putText(img, f"Fingers: {count}  {fingers}", (20,40),
20:                     cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,0), 2)
21:
22:     cv2.imshow("Hands + Fingers", img)
23:     if cv2.waitKey(1) & 0xFF == ord('q'): break
24:
25: cap.release()
26: cv2.destroyAllWindows()
27:
```

Praktikum D5 — Pengenalan Gestur Tangan (Hand Gesture Recognition)

Tujuan:
Mahasiswa mampu mengenali gestur tangan sederhana berdasarkan hubungan geometris antar *landmark*.

Deskripsi:
Gestur tangan merupakan salah satu bentuk komunikasi nonverbal yang dapat diterjemahkan oleh sistem visi komputer. Dalam praktikum ini, mahasiswa akan merancang aturan geometris untuk mengenali gestur dasar seperti “Thumbs Up”, “OK”, dan “Rock–Paper–Scissors”.

Langkah:

1. Gunakan *landmark* kunci seperti ujung ibu jari (4), ujung telunjuk (8), pergelangan tangan (0), dan lainnya.
2. Definisikan kriteria masing-masing gestur, misalnya:
   - **OK:** jarak antara ujung ibu jari dan telunjuk < ambang tertentu.
   - **Thumbs Up:** ibu jari mengarah ke atas dan jauh dari pergelangan.
   - **Rock/Paper/Scissors:** menggunakan rata-rata jarak ujung jari ke pergelangan.
3. Tampilkan label gestur yang dikenali pada jendela kamera.
4. Uji dengan beberapa posisi tangan.

Indikator Keberhasilan:
Sistem dapat mengenali minimal tiga jenis gestur tangan dengan akurasi yang konsisten dan respons waktu <0.3 detik per frame.

```
01: import cv2, numpy as np
02: from cvzone.HandTrackingModule import HandDetector
03:
04: def dist(a,b): return np.linalg.norm(np.array(a)-np.array(b))
05:
06: def classify_gesture(hand):
07:     # hand["lmList"] berisi 21 titik (x,y,z) dalam piksel saat
flipType=True
08:     lm = hand["lmList"]
09:     wrist = np.array(lm[0][:2])
10:     thumb_tip = np.array(lm[4][:2])
11:     index_tip = np.array(lm[8][:2])
12:     middle_tip = np.array(lm[12][:2])
13:     ring_tip = np.array(lm[16][:2])
14:     pinky_tip = np.array(lm[20][:2])
15:     # Heuristik jarak relatif
16:     r_mean = np.mean([dist(index_tip,wrist), dist(middle_tip,wrist),
17:                       dist(ring_tip,wrist), dist(pinky_tip,wrist),
dist(thumb_tip,wrist)])
18:     # Aturan:
19:     if dist(thumb_tip, index_tip) < 35: return "OK"
20:     # Thumbs up: ibu jari tinggi (y kecil), dan jauh dari wrist
21:     if (thumb_tip[1] < wrist[1]-40) and (dist(thumb_tip, wrist) >
0.8*dist(index_tip, wrist)):
22:         return "THUMBS_UP"
23:     if r_mean < 120: return "ROCK"
24:     if r_mean > 200: return "PAPER"
25:     if dist(index_tip,wrist)>180 and dist(middle_tip,wrist)>180 and \
```

```
26:         dist(ring_tip,wrist)<160 and dist(pinky_tip,wrist)<160:
27:          return "SCISSORS"
28:     return "UNKNOWN"
29:
30: cap = cv2.VideoCapture(0)
31: if not cap.isOpened():
32:     raise RuntimeError("Kamera tidak bisa dibuka.")
33:
34: detector = HandDetector(staticMode=False, maxHands=1,
modelComplexity=1,
35:                         detectionCon=0.5, minTrackCon=0.5)
36:
37: while True:
38:     ok, img = cap.read()
39:     if not ok: break
40:     hands, img = detector.findHands(img, draw=True, flipType=True)
41:     if hands:
42:         label = classify_gesture(hands[0])
43:         cv2.putText(img, f"Gesture: {label}", (20,40),
44:                     cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,255), 2)
45:
46:     cv2.imshow("Hand Gestures (cvzone)", img)
47:     if cv2.waitKey(1) & 0xFF == ord('q'): break
48:
49: cap.release()
50: cv2.destroyAllWindows()
51:
```

Praktikum D6 — Analisis Gerakan Tubuh dan Penghitung Aktivitas

Tujuan:
Mahasiswa mampu membuat sistem penghitung aktivitas fisik (seperti *squat* dan *push-up*) menggunakan data *landmark* pose dan logika kondisi gerakan.

Deskripsi:
Praktikum ini mengintegrasikan kemampuan deteksi pose dengan logika analisis gerakan. Mahasiswa akan menerapkan penghitungan sudut dan rasio jarak untuk mengenali dua posisi utama (naik dan turun), serta menerapkan teknik *debounce* untuk menghindari penghitungan ganda.

Langkah:

1. Gunakan MediaPipe Pose untuk mendeteksi titik tubuh yang relevan.
2. **Untuk squat:** hitung sudut lutut kiri dan kanan, tentukan kondisi *up* ($\theta > 160°$) dan *down* ($\theta < 80°$).
3. **Untuk push-up:** gunakan rasio jarak bahu–pergelangan terhadap bahu–pinggul.
4. Implementasikan logika transisi `down → up` sebagai satu repetisi.
5. Tampilkan jumlah repetisi dan status posisi di layar secara real-time.

Indikator Keberhasilan:
Sistem mampu menghitung jumlah gerakan dengan kesalahan di bawah 10% untuk 10–20 repetisi.

```
01: import cv2, numpy as np
02: from collections import deque
03: from cvzone.PoseModule import PoseDetector
04:
05: MODE = "squat"   # tekan 'm' untuk toggle ke "pushup"
06: KNEE_DOWN, KNEE_UP = 80, 160     # ambang squat (deg)
07: DOWN_R, UP_R     = 0.85, 1.00    # ambang push-up (rasio)
08: SAMPLE_OK        = 4             # minimal frame konsisten sebelum
ganti state
09:
10: cap = cv2.VideoCapture(0)
11: if not cap.isOpened():
12:     raise RuntimeError("Kamera tidak bisa dibuka.")
13:
14: detector = PoseDetector(staticMode=False, modelComplexity=1,
15:                          enableSegmentation=False, detectionCon=0.5,
trackCon=0.5)
16:
17: count, state = 0, "up"
18: debounce = deque(maxlen=6)
19:
20: def ratio_pushup(lm):
21:     # gunakan kiri: 11=shoulderL, 15=wristL, 23=hipL
22:     sh = np.array(lm[11][1:3])
23:     wr = np.array(lm[15][1:3])
24:     hp = np.array(lm[23][1:3])
25:     return np.linalg.norm(sh - wr) / (np.linalg.norm(sh - hp) + 1e-8)
26:
27: while True:
28:     ok, img = cap.read()
```

```
    if not ok: break
    img = detector.findPose(img, draw=True)
    lmList, _ = detector.findPosition(img, draw=False)  #
[(id,x,y,z,vis), ...]
    flag = None

    if lmList:
        if MODE == "squat":
            # rata-rata sudut lutut kiri & kanan
            # angL, _ = detector.findAngle(img, 23, 25, 27, draw=False)
            # angR, _ = detector.findAngle(img, 24, 26, 28, draw=False)


            angL, img = detector.findAngle(lmList[23][0:2],
                                           lmList[25][0:2],
                                           lmList[27][0:2],
                                           img=img,
                                           color=(0, 0, 255),
                                           scale=10)

            angR, img = detector.findAngle(lmList[24][0:2],
                                           lmList[26][0:2],
                                           lmList[28][0:2],
                                           img=img,
                                           color=(0, 255, 0),
                                           scale=10)

            ang = (angL + angR) / 2.0
            if ang < KNEE_DOWN: flag = "down"
            elif ang > KNEE_UP:  flag = "up"
            cv2.putText(img, f"Knee: {ang:5.1f}", (20,70),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,0), 2)
        else:
            # push-up: rasio (shoulder-wrist)/(shoulder-hip)
            r = ratio_pushup(lmList)
            if r < DOWN_R: flag = "down"
            elif r > UP_R:  flag = "up"
            cv2.putText(img, f"Ratio: {r:4.2f}", (20,70),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 2)

        debounce.append(flag)
        if debounce.count("down") >= SAMPLE_OK and state == "up":
            state = "down"
        if debounce.count("up") >= SAMPLE_OK and state == "down":
            state = "up"; count += 1

    cv2.putText(img, f"Mode: {MODE.upper()}  Count: {count}", (20,40),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,255,255), 2)
    cv2.putText(img, f"State: {state}", (20,100),
                cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,255,255), 2)

    cv2.imshow("Pose Counter", img)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'): break
    if key == ord('m'): MODE = "pushup" if MODE == "squat" else "squat"

cap.release()
cv2.destroyAllWindows()

```

E. Hasil Praktikum

Lengkapi hasil tabel praktikum berikut:

No	Nama Praktikum	Hasil Praktikum
1	**D1 — Inisialisasi Kamera dan Akuisisi Citra**	
2	**D2 — Deteksi Pose dan Analisis Sudut Tubuh**	
3	**D3 — Deteksi Wajah dan Analisis Kedipan Mata**	
4	**D4 — Deteksi Tangan dan Penghitungan Jumlah Jari**	
5	**D5 — Pengenalan Gestur Tangan (Hand Gesture Recognition)**	
6	**D6 — Analisis Gerakan Tubuh dan Penghitung Aktivitas**	

F. Penugasan

1. Kumpulkan Laporan Praktikum dari jobsheet ini dalam bentuk Microsoft word sesuai dengan format jobsheet praktikum dan dikumpulkan di web LMS. (JANGAN DALAM BENTUK PDF)
2. Kumpulkan luaran kode praktikum dalam bentuk ipynb yang sudah diunggah pada akun github masing-masing. Lampirkan tautan github yang sudah di unggah melalui laman LMS.

G. Kesimpulan

Melalui enam kegiatan praktikum yang telah dilaksanakan, mahasiswa memperoleh pemahaman dan keterampilan dasar dalam menerapkan teknologi *Computer Vision* berbasis **MediaPipe** dan **cvzone**.Praktikum diawali dengan penguasaan proses akuisisi citra dari kamera, kemudian dilanjutkan dengan penerapan *pose estimation*, *face mesh*, dan *hand tracking* untuk mengenali struktur tubuh manusia secara digital.

Mahasiswa juga telah mempelajari bagaimana mengubah data *landmark* menjadi informasi bermakna melalui analisis geometri, seperti perhitungan sudut, jarak, dan rasio. Proses ini kemudian diintegrasikan dalam sistem sederhana untuk mendeteksi gestur tangan serta menghitung aktivitas fisik seperti *squat* dan *push-up*.

Keseluruhan kegiatan praktikum membekali mahasiswa dengan kemampuan teknis dalam pengolahan citra *real-time*, serta memberikan wawasan mengenai bagaimana teknologi *machine learning* dapat digunakan untuk memahami gerakan dan ekspresi manusia. Dengan penguasaan materi ini, mahasiswa diharapkan mampu mengembangkan aplikasi visi komputer lanjutan yang relevan dengan bidang teknik, kesehatan, industri, maupun sistem interaksi manusia–komputer.

H. Daftar Pustaka

- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11), 2278–2324.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet classification with deep convolutional neural networks*. Advances in Neural Information Processing Systems.

- Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv:1409.1556.
- Szegedy, C., et al. (2015). *Going deeper with convolutions*. IEEE Conference on Computer Vision and Pattern Recognition.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. IEEE Conference on Computer Vision and Pattern Recognition.
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). *Densely connected convolutional networks*. IEEE CVPR.
- Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking model scaling for convolutional neural networks*. ICML.
- Esteva, A., et al. (2017). *Dermatologist-level classification of skin cancer with deep neural networks*. Nature, 542(7639), 115–118.
- Bojarski, M., et al. (2016). *End to End Learning for Self-Driving Cars*. arXiv:1604.07316.
- Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
- Chollet, F. (2015). *Keras: Deep Learning for humans*. GitHub repository.