

BAB IV

IMPLEMENTASI

4.1 Analysis

Pada tahap analisis, penelitian ini berfokus pada pemahaman pemanfaatan MediaPipe dan OpenCV secara optimal dalam mendeteksi kesalahan gerakan saat melakukan latihan squat di gym. MediaPipe, sebagai framework yang menawarkan solusi pemrosesan pose tubuh secara real-time, memungkinkan pengambilan data berupa titik-titik penting tubuh seperti lutut, pinggul, bahu, dan pergelangan kaki. Data landmark tersebut kemudian dianalisis lebih lanjut menggunakan OpenCV, misalnya untuk menghitung sudut antara beberapa titik tubuh. Melalui metode ini, sistem dapat memberikan umpan balik langsung mengenai kesalahan postur atau teknik yang sering terjadi, seperti posisi lutut yang terlalu maju melewati ujung kaki atau punggung yang tidak lurus saat squat. Hasil analisis memperlihatkan bahwa kombinasi kedua teknologi ini berpotensi besar meningkatkan ketepatan deteksi kesalahan gerakan dibandingkan metode manual yang biasanya mengandalkan pengamatan pelatih atau instruktur.

Selain itu, penelitian ini juga mengevaluasi implementasi aplikasi berbasis Streamlit sebagai platform interaktif untuk menampilkan hasil deteksi. Streamlit dipilih karena kemampuannya mengintegrasikan kode Python langsung ke dalam antarmuka web yang ramah pengguna. Melalui aplikasi ini, pengguna dapat memantau video atau tangkapan kamera secara real-time yang kemudian dianalisis oleh model deteksi. Hasil analisis, seperti indikator kesalahan gerakan dan rekomendasi perbaikan, disajikan secara visual dalam bentuk teks, grafik, atau overlay pada video. Diharapkan kedepannya aplikasi ini dapat responsif dan mampu memberikan umpan balik secara cepat, meskipun terdapat beberapa keterbatasan pada akurasi deteksi, terutama pada kondisi pencahayaan yang kurang baik atau saat pengguna mengenakan pakaian dengan warna serupa latar belakang.

4.1.1 State Diagram

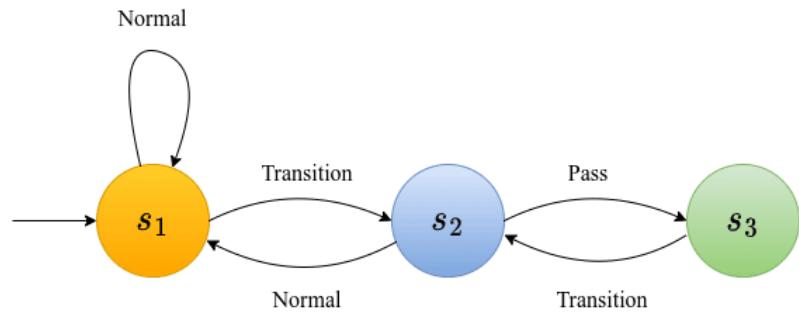


Diagram state transition di atas menggambarkan alur perubahan status dalam proses analisis gerakan squat. Terdapat tiga state utama, yaitu S_1 (Normal), S_2 (Transition), dan S_3 (Pass). Proses dimulai dari state S_1 , yang merepresentasikan kondisi normal. Pada state ini, sistem akan tetap berada di S_1 selama mendeteksi gerakan normal. Jika terdeteksi adanya perubahan atau peralihan gerakan, sistem akan berpindah ke state S_2 (Transition). Dari S_2 , sistem dapat kembali ke S_1 jika gerakan kembali normal, atau melanjutkan ke S_3 (Pass) apabila gerakan telah melewati fase transisi dengan benar. Selain itu, dari S_3 , sistem dapat kembali ke S_2 jika terjadi transisi ulang. Diagram ini membantu memvisualisasikan bagaimana sistem mendeteksi dan mengklasifikasikan setiap fase gerakan squat secara berurutan dan dinamis.

4.1.2 Perhitungan Manual Sudut dan Thresholds

Analisis postur pada sistem ini dilakukan dengan cara mengukur sudut-sudut utama pada tubuh yang dibentuk dari susunan kerangka landmark. Cara ini memungkinkan penilaian pergerakan secara objektif dan terukur. Penghitungan sudut didasarkan pada koordinat (x, y) dari setiap landmark sendi yang diperoleh dari model MediaPipe Pose. Sudut-sudut yang dihasilkan kemudian dibandingkan dengan rentang nilai threshold yang telah ditentukan untuk memutuskan apakah suatu gerakan tergolong “benar” atau “salah”.

Fungsi inti yang digunakan adalah $find_angle(p1, p2, ref_pt)$. Fungsi ini menghitung sudut antara dua vektor:

1. Vektor \vec{A} , yang membentang dari titik referensi (ref_pt) ke titik pertama ($p1$).

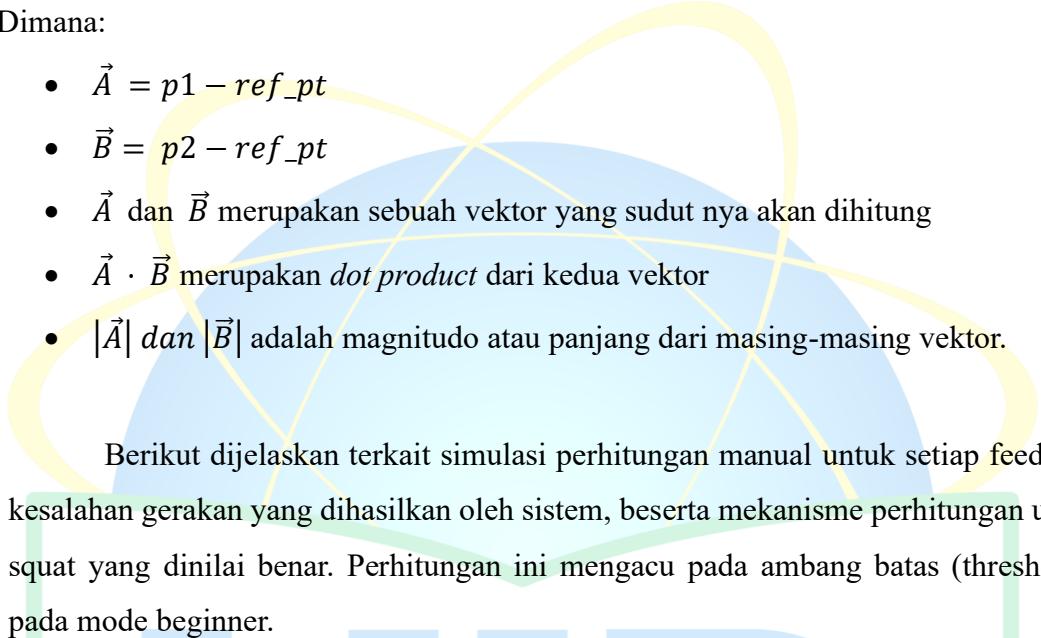
2. Vektor \vec{B} , yang membentang dari titik referensi (*ref_pt*) ke titik kedua (*p2*).

Rumus utama untuk menghitung sudut antara dua vektor adalah:

$$\theta = \arccos\left(\frac{\vec{A} \cdot \vec{B}}{|\vec{A}| |\vec{B}|}\right)$$

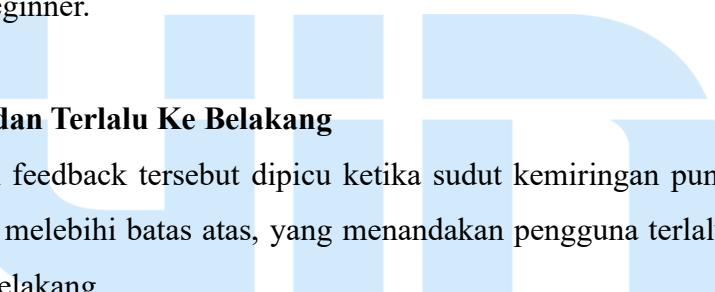
Dimana:

- $\vec{A} = p1 - ref_pt$
- $\vec{B} = p2 - ref_pt$
- \vec{A} dan \vec{B} merupakan sebuah vektor yang sudutnya akan dihitung
- $\vec{A} \cdot \vec{B}$ merupakan *dot product* dari kedua vektor
- $|\vec{A}|$ dan $|\vec{B}|$ adalah magnitudo atau panjang dari masing-masing vektor.



Berikut dijelaskan terkait simulasi perhitungan manual untuk setiap feedback kesalahan gerakan yang dihasilkan oleh sistem, beserta mekanisme perhitungan untuk squat yang dinilai benar. Perhitungan ini mengacu pada ambang batas (thresholds) pada mode beginner.

Skenario: Badan Terlalu Ke Belakang



Untuk feedback tersebut dipicu ketika sudut kemiringan punggung terhadap garis vertikal melebihi batas atas, yang menandakan pengguna terlalu bersandar dan condong ke belakang.

- Sudut Relevan: *hip_vertical_angle*
- Nilai Thresholds: *HIP_THRESH* adalah [10, 50], sehingga kondisi pemicu adalah jika sudut $> 50^\circ$

Contoh Perhitungan:

1. Koordinat Landmark (Hipotetis):

- Bahu (*shldr_coord*): (280, 400)
- Pinggul (*hip_coord*): (350, 410)

2. Perhitungan Vektor:

- \vec{A} (Vektor Pinggul ke Bahu): $(280 - 350, 400 - 410) = (-70, -10)$
- \vec{B} (Vektor Vertikal dari Pinggul): $(350 - 350, 0 - 410) = (0, -410)$

3. Perhitungan Dot Product:

- $\vec{A} \cdot \vec{B} = (-70 \times 0) + (-10 \times -410) = 4100$

4. Perhitungan Magnitudo:

- $|\vec{A}| = \sqrt{(-70)^2 + (-10)^2} = \sqrt{4900 + 100} = \sqrt{5000} \approx 70.71$

- $|\vec{B}| = \sqrt{0^2 + (-410)^2} = 410$

5. Perhitungan Sudut Akhir:

- $\theta = \arccos\left(\frac{4100}{70.71 \times 410}\right) = \arccos\left(\frac{4100}{28991.1}\right) = \arccos(0.1414) \approx 81.8^\circ$

Hasilnya sudut yang dihitung adalah 81.8° . Karena $81.8^\circ > 50^\circ$, kondisi terpenuhi dan sistem akan menampilkan umpan balik “BADAN TERLALU KE BELAKANG”

Skenario: Badan Terlalu Ke Depan

Feedback tersebut dipicu ketika sudut kemiringan tubuh terhadap garis vertikal berada di bawah batas minimum, menandakan punggung pengguna terlalu bungkuk ke depan.

- Sudut relevan: *hip_vertical_angle*
- Nilai Thresholds: *HIP_THRESH* adalah $[10, 50]$, sehingga kondisi pemicu adalah jika sudut $< 10^\circ$.

Contoh Perhitungan:

A. Koordinat Landmark (Hipotetis):

- Bahu (*shldr_coord*): $(355, 250)$
- Pinggul (*hip_coord*): $(350, 410)$

A. Pembentukan Vektor:

- $\vec{A} = (355 - 350, 250 - 410) = (5, -160)$
- $\vec{B} = (0, -410)$

A. Perhitungan Dot Product:

- $\vec{A} \cdot \vec{B} = (5 \times 0) + (-160 \times -410) = 65600$

A. Perhitungan Magnitudo:

- $|\vec{A}| = \sqrt{5^2 + (-160)^2} = \sqrt{25 + 25600} = \sqrt{25625} \approx 160.08$
- $|\vec{B}| = 410$

A. Perhitungan Sudut Akhir:

$$\theta = \arccos\left(\frac{65600}{160.08 \times 410}\right) = \arccos\left(\frac{65600}{65632.8}\right) = \arccos(0.9995) \approx 1.8^\circ$$

Hasilnya sudut yang dihitung adalah 1.8° . Karena $1.8^\circ < 10^\circ$, kondisi terpenuhi dan sistem akan menampilkan feedback “BADAN TERLALU KE DEPAN”.

Skenario: Lutut Melewati Jari Kaki

Feedback ini dipicu ketika sudut kemiringan tulang kering (tibia) terhadap garis vertikal terlalu besar.

- Sudut relevan: *ankle_vertical_angle*
- Nilai Thresholds: *ANKLE_THRESH* adalah 45, sehingga kondisi pemicu adalah jika sudut $> 45^\circ$.

Contoh Perhitungan:

A. Koordinat Landmark (Hipotetis):

- Lutut ('knee_coord'): '(440, 490)'
- Pergelangan Kaki ('ankle_coord'): '(360, 560)'

A. Perhitungan Vektor (diukur dengan referensi pada pergelangan kaki):

- \vec{A} (Vektor Pergelangan Kaki ke Lutut): $(440 - 360, 490 - 560) = (80, -70)$
- \vec{B} (Vektor Vertikal dari Pergelangan Kaki): $(360 - 360, 0 - 560) = (0, -560)$

A. Perhitungan Dot Product:

- $\vec{A} \cdot \vec{B} = (80 \times 0) + (-70 \times -560) = 39200$

A. Perhitungan Magnitudo:

- $|\vec{A}| = \sqrt{80^2 + (-70)^2} = \sqrt{6400 + 4900} = \sqrt{11300} \approx 106.3$
- $|\vec{B}| = 560$

A. Perhitungan Sudut Akhir:

- $\theta = \arccos\left(\frac{39200}{106.3 \times 560}\right) = \arccos\left(\frac{39200}{59528}\right) = \arccos(0.6585) \approx 48.8^\circ$

Didapatkan simulasi hasil sudut yang dihitung adalah 48.8° . Karena $48.8^\circ > 45^\circ$, kondisi terpenuhi dan sistem akan menampilkan feedback “LUTUT MELEWATI JARI KAKI”.

Skenario: Squat Terlalu Dalam

Feedback ini muncul ketika pengguna turun melebihi batas kedalaman squat yang ideal, yang diukur dari sudut kemiringan paha dan juga pinggul.

- Sudut relevan: *knee_vertical_angle*
- Nilai Thresholds: *KNEE_THRESH* adalah [50, 70, 100], sehingga kondisi pemicu adalah jika sudut $> 100^\circ$.

Contoh Perhitungan:

A. Koordinat Landmark (Hipotetis):

- Pinggul (*hip_coor*): (300, 510)
- Lutut (*knee_coord*): (420, 490)

A. Pembentukan Vektor: Sudut diukur dengan referensi pada lutut ('*knee_coord*').

- \vec{A} (Vektor Lutut ke Pinggul): $(300 - 420, 510 - 490) = (-120, 20)$
- \vec{B} (Vektor Vertikal dari Lutut): $(420 - 420, 0 - 490) = (0, -490)$

A. Perhitungan Dot Product:

- $\vec{A} \cdot \vec{B} = (-120 \times 0) + (20 \times -490) = -9800$

A. Perhitungan Magnitudo:

- $|\vec{A}| = \sqrt{(-120)^2 + 20^2} = \sqrt{14400 + 400} = \sqrt{14800} \approx 121.65$

- $|\vec{B}| = 490$

A. Perhitungan Sudut Akhir:

- $\theta = \arccos\left(\frac{-9800}{121.65 \times 490}\right) = \arccos\left(\frac{-9800}{59608.5}\right) = \arccos(-0.1644) \approx 99.45^\circ$

Berdasarkan hasil perhitungan tersebut, sudut yang terbentuk di lutut adalah sekitar 99.45° . Nilai ini mendekati batas bawah threshold yang ditetapkan, misalnya 100° . Jika pengguna melakukan gerakan squat sedikit lebih dalam, misalkan koordinat pinggul berubah menjadi $(290, 515)$, maka sudut yang dihasilkan akan melebihi 100° . Pada kondisi tersebut, sistem akan memberikan umpan balik berupa peringatan “SQUAT TERLALU DALAM” kepada pengguna.

Skenario: Squat Benar (Correct Squat)

Pada kondisi squat yang benar, seluruh sudut utama tubuh berada dalam rentang yang dapat diterima pada tiap fase gerakan (state s1 hingga state s3 atau PASS). Sistem akan mengidentifikasi repetisi squat sebagai valid jika seluruh parameter sudut berada dalam batas yang telah ditentukan.

- Kondisi:
 - *hip_vertical_angle* berada di antara 10° dan 50° .
 - *ankle_vertical_angle* tidak melebihi 45° .
 - *knee_vertical_angle* berada dalam rentang PASS, yaitu antara 70° dan 100° .

Contoh Perhitungan:

A. Koordinat Landmark (Hipotetis):

- Bahu (*shldr_coord*): $(400, 390)$
- Pinggul (*hip_coord*): $(320, 490)$
- Lutut (*knee_coord*): $(420, 500)$
- Pergelangan Kaki (*ankle_coord*): $(400, 570)$

A. Perhitungan ‘*hip_vertical_angle*’:

- Vektor \vec{A} (Pinggul ke Bahu): $(80, -100)$

- Vektor \vec{B} (Vertikal Pinggul): $(0, -490)$
- $\theta_{\text{hip}} = \arccos\left(\frac{49000}{\sqrt{16400 \times 490}}\right) = \arccos(0.781) \approx 38.6^\circ$

Interpretasi: 38.6° berada dalam rentang $(10^\circ, 50^\circ)$, sehingga dinyatakan benar.

A. Perhitungan *ankle_vertical_angle*:

- Vektor \vec{A} (Ankle ke Lutut): $(20, -70)$
- Vektor \vec{B} (Vertikal Ankle): $(0, -570)$
- $\theta_{\text{ankle}} = \arccos\left(\frac{39900}{\sqrt{5300 \times 570}}\right) = \arccos(0.961) \approx 16.0^\circ$

Interpretasi: $16.0^\circ < 45^\circ$, sehingga dinyatakan benar.

A. Perhitungan *knee_vertical_angle*:

- Vektor \vec{A} (Lutut ke Pinggul): $(-100, -10)$
- Vektor \vec{B} (Vertikal Lutut): $(0, -500)$
- $\theta_{\text{knee}} = \arccos\left(\frac{5000}{\sqrt{10100 \times 500}}\right) = \arccos(0.0995) \approx 84.3^\circ$

Interpretasi: 84.3° berada dalam rentang PASS ($70^\circ, 100^\circ$), sehingga dinyatakan benar.

Karena semua sudut berada dalam rentang yang telah ditentukan, sistem menganggap gerakan ini sebagai repetisi squat yang valid dan akan menambah jumlah hitungan squat (*Squat_Count*) ketika pengguna kembali ke posisi awal (*state s1*).

4.1.3 Perhitungan Manual Metrik *Mean Per Joint Position Error* (MPJPE)

Dalam evaluasi performa model estimasi postur, diperlukan sebuah metrik kuantitatif untuk mengukur tingkat akurasi. Salah satu metrik yang sangat umum digunakan baik dalam 2D/3D *Pose Estimation* adalah *Mean Per Joint Position Error* (MPJPE). Pada bagian ini akan dijelaskan secara rinci metodologi perhitungan MPJPE, mulai dari definisi matematis hingga simulasi perhitungan manual pada skenario dengan tingkat akurasi tinggi.

Mean Per Joint Position Error (MPJPE) didefinisikan sebagai rata-rata *Euclidean Distance* antara koordinat persendian (landmark) yang diprediksi oleh model dan koordinat persendian referensi (ground truth). Metrik ini memberikan ukuran kesalahan (error) rata-rata dalam satuan piksel (px). Nilai MPJPE yang lebih rendah mengindikasikan akurasi model yang lebih tinggi.

Perhitungan MPJPE didasarkan pada dua rumus utama berikut:

- *Euclidean Distance (Per Joint Position Error – PJPE)*

Untuk mengetahui seberapa akurat model mendekripsi posisi sendi, pertama-tama dihitung jarak lurus antara titik prediksi $P_p = (x_p, y_p)$ dan titik *ground truth* $P_{gt} = (x_{gt}, y_{gt})$. Di dalam code python rumus ini diimplementasikan dalam kode menggunakan fungsi `np.linalg.norm()`

$$E_{\text{joint}} = \sqrt{(x_p - x_{gt})^2 + (y_p - y_{gt})^2}$$

- *Mean Per Joint Position Error (MPJPE)*

Setelah memperoleh nilai error untuk setiap sendi, langkah berikutnya adalah menghitung rata-rata dari semua error tersebut. Nilai rata-rata inilah yang disebut sebagai MPJPE. Secara matematis, MPJPE dirumuskan sebagai berikut:

$$\text{MPJPE} = \frac{\sum_{i=1}^N E_{\text{joint}_i}}{N_{\text{joints}}}$$

Di mana N_{joints} adalah jumlah total sendi yang dievaluasi. Dengan kata lain, MPJPE adalah total error seluruh sendi dibagi dengan jumlah sendi. Nilai MPJPE yang lebih kecil menunjukkan prediksi model yang semakin mendekati data referensi (ground truth), sehingga akurasi model semakin baik.

Untuk memberikan gambaran yang lebih jelas, berikut adalah contoh simulasi perhitungan manual MPJPE pada satu frame, dengan asumsi bahwa perbedaan antara prediksi dan ground truth relatif kecil (menandakan akurasi tinggi):

1. Penetapan Koordinat Hipotetis:

Misalkan sistem memproses sebuah *frame* dan menghasilkan set koordinat berikut, di mana selisih antara prediksi dan *ground truth* terlihat seperti tabel dibawah ini:

Tabel 4. 1 Hipotesis Kordinat Landmark

Sendi (Joint)	Koordinat Prediksi $P_p = (x_p, y_p)$	Koordinat Ground Truth $P_{gt} = (x_{gt}, y_{gt})$
Shoulder	(150, 210)	(154, 214.5)
Elbow	(180, 280)	(186, 285.3)
Wrist	(200, 350)	(202, 352.2)
Hip	(140, 340)	(147, 345.6)
Knee	(130, 450)	(135, 456.2)
Ankle	(120, 550)	(128, 554.3)
Foot	(140, 580)	(145, 583.3)

2. Perhitungan Per Joint Position Error (PJPE):

Dengan menggunakan rumus *Euclidean Distance*, error untuk setiap sendi dihitung sebagai berikut:

- Shoulder:

$$\begin{aligned} E_{\text{shoulder}} &= \sqrt{(150 - 154)^2 + (210 - 214.5)^2} = \sqrt{(-4)^2 + (-4.5)^2} \\ &= \sqrt{16 + 20.25} = \sqrt{36.25} \approx 6.02 \text{ px} \end{aligned}$$

- Elbow:

$$\begin{aligned} E_{\text{elbow}} &= \sqrt{(180 - 186)^2 + (280 - 285.3)^2} = \sqrt{(-6)^2 + (-5.3)^2} \\ &= \sqrt{36 + 28.09} = \sqrt{64.09} \approx 8.01 \text{ px} \end{aligned}$$

- Wrist:

$$\begin{aligned} E_{\text{wrist}} &= \sqrt{(200 - 202)^2 + (350 - 352.2)^2} = \sqrt{(-2)^2 + (-2.2)^2} \\ &= \sqrt{4 + 4.84} = \sqrt{8.84} \approx 2.97 \text{ px} \end{aligned}$$

(Nilai error yang sangat kecil ini menunjukkan akurasi yang sangat tinggi untuk deteksi pergelangan tangan pada frame ini)

- Hip:

$$E_{\text{hip}} = \sqrt{(140 - 147)^2 + (340 - 345.6)^2} = \sqrt{(-7)^2 + (-5.6)^2} \\ = \sqrt{49 + 31.36} = \sqrt{80.36} \approx 8.96 \text{ px}$$

- Knee:

$$E_{\text{knee}} = \sqrt{(130 - 135)^2 + (450 - 456.2)^2} = \sqrt{(-5)^2 + (-6.2)^2} \\ = \sqrt{25 + 38.44} = \sqrt{63.44} \approx 7.96 \text{ px}$$

- Ankle:

$$E_{\text{ankle}} = \sqrt{(120 - 128)^2 + (550 - 554.3)^2} = \sqrt{(-8)^2 + (-4.3)^2} \\ = \sqrt{64 + 18.49} = \sqrt{82.49} \approx 9.08 \text{ px}$$

- Foot:

$$E_{\text{foot}} = \sqrt{(140 - 145)^2 + (580 - 583.3)^2} = \sqrt{(-5)^2 + (-3.3)^2} \\ = \sqrt{25 + 10.89} = \sqrt{35.89} \approx 5.99 \text{ px}$$

3. Perhitungan Nilai Akhir MPJPE:

Nilai MPJPE dihitung dengan merata-ratakan semua nilai PJPE yang diperoleh pada step sebelumnya

- Total Error (Jumlah PJPE):

$$\sum E_{\text{joint}} = 6.02 + 8.01 + 2.97 + 8.96 + 7.96 + 9.08 + 5.99 = 48.99$$

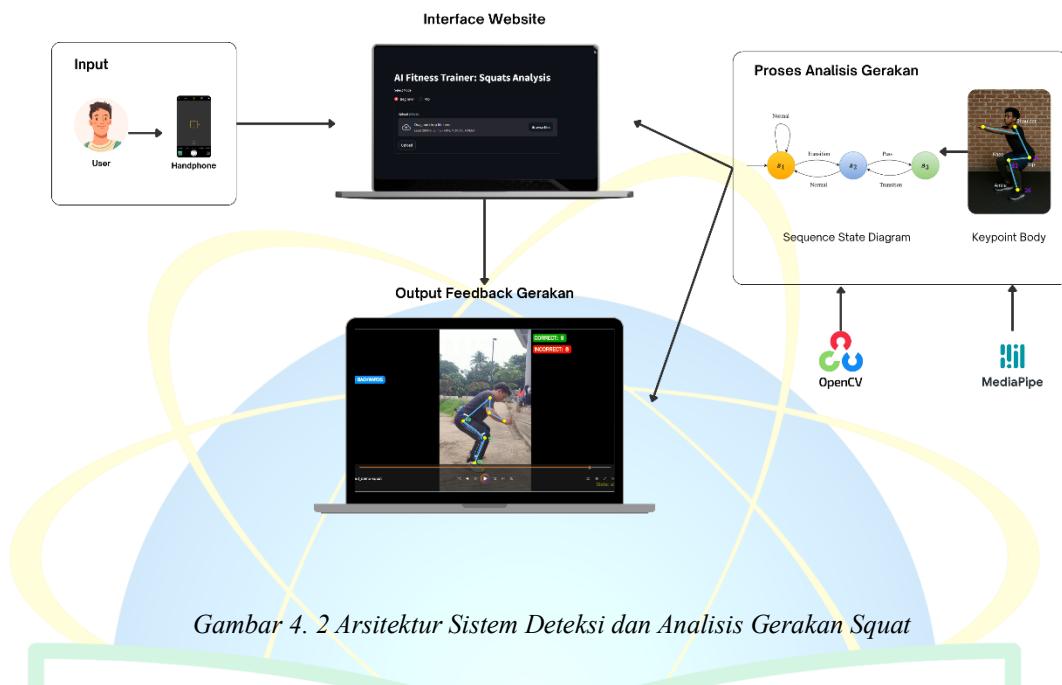
- MPJPE (Rata-rata):

$$\text{MPJPE} = \frac{48.99}{7} \approx 7.00 \text{ px}$$

Nilai MPJPE sekitar 7 piksel ini menunjukkan bahwa rata-rata prediksi posisi sendi model hanya berselisih sekitar 7 piksel dari *ground truth*. Hal ini mengindikasikan bahwa evaluasi analisis video tersebut memiliki tingkat akurasi yang sangat baik pada skenario tersebut.

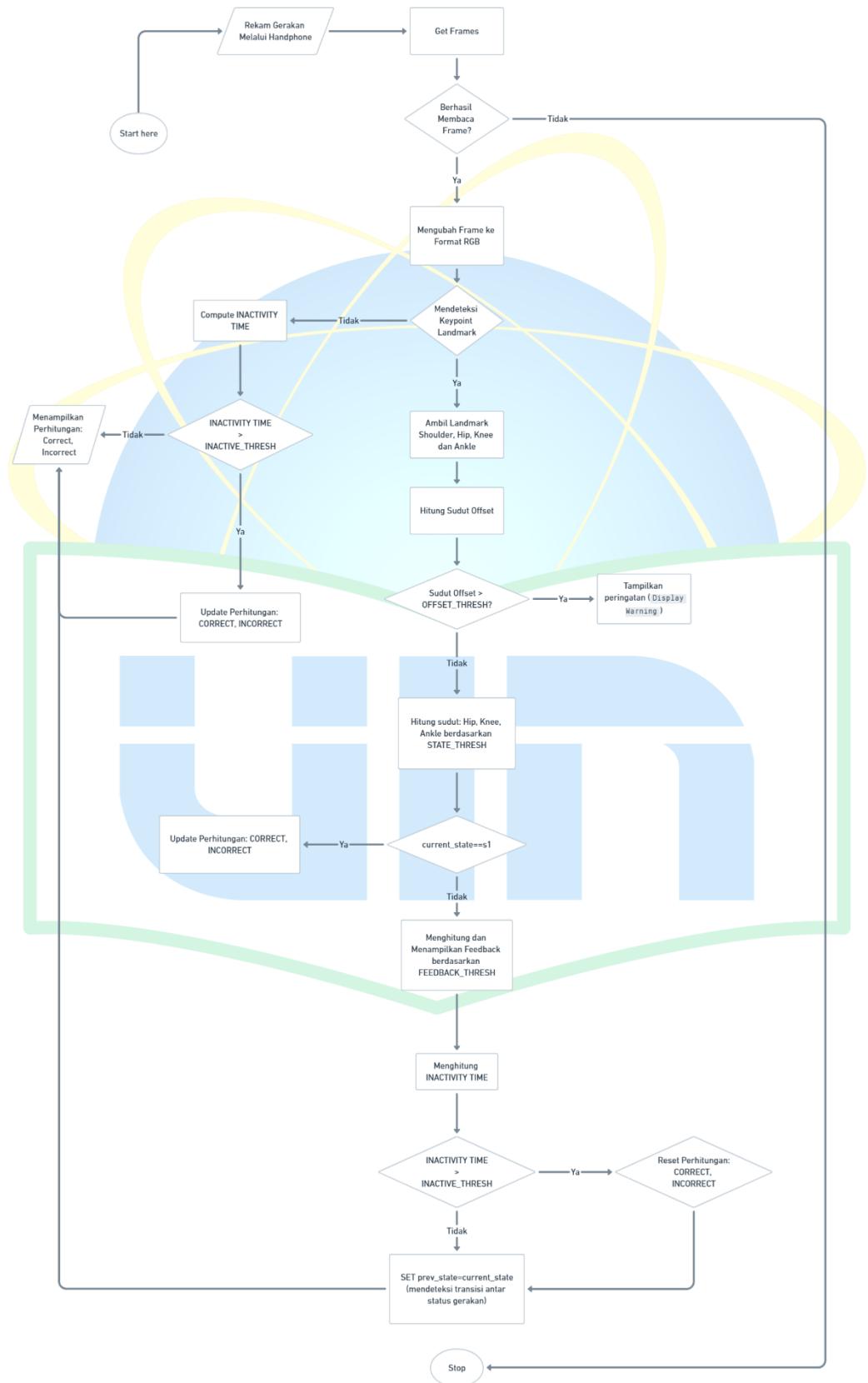
4.2 Design

4.2.1 Desain Arsitektur Workflow Sistem



Desain arsitektur fitness vision memiliki 3 bagian yaitu: input, proses analisis gerakan squat, dan output. Tahapan awal dimulai ketika pengguna merekam aktivitas squat menggunakan kamera, lalu mengunggah video tersebut ke aplikasi web. Setelah proses unggah selesai, sistem memanfaatkan MediaPipe untuk mendeteksi pose tubuh dan menandai titik-titik penting seperti lutut, pinggul, bahu, serta pergelangan kaki. Landmark yang dihasilkan kemudian dianalisis oleh OpenCV untuk menghitung sudut-sudut dan menentukan fase gerakan sesuai dengan state diagram yang telah dibuat (S₁: Normal, S₂: Transition, S₃: Pass). Seluruh hasil analisis ini kemudian divisualisasikan di dalam aplikasi, sehingga pengguna dapat memperoleh umpan balik berupa jumlah gerakan yang benar maupun salah, serta tampilan visual pose tubuh secara real-time. Dengan alur kerja seperti ini, sistem dapat memberikan feedback yang interaktif dan informatif mengenai kualitas gerakan squat yang dilakukan oleh pengguna.

4.2.2 Flowchart Fitness Vision



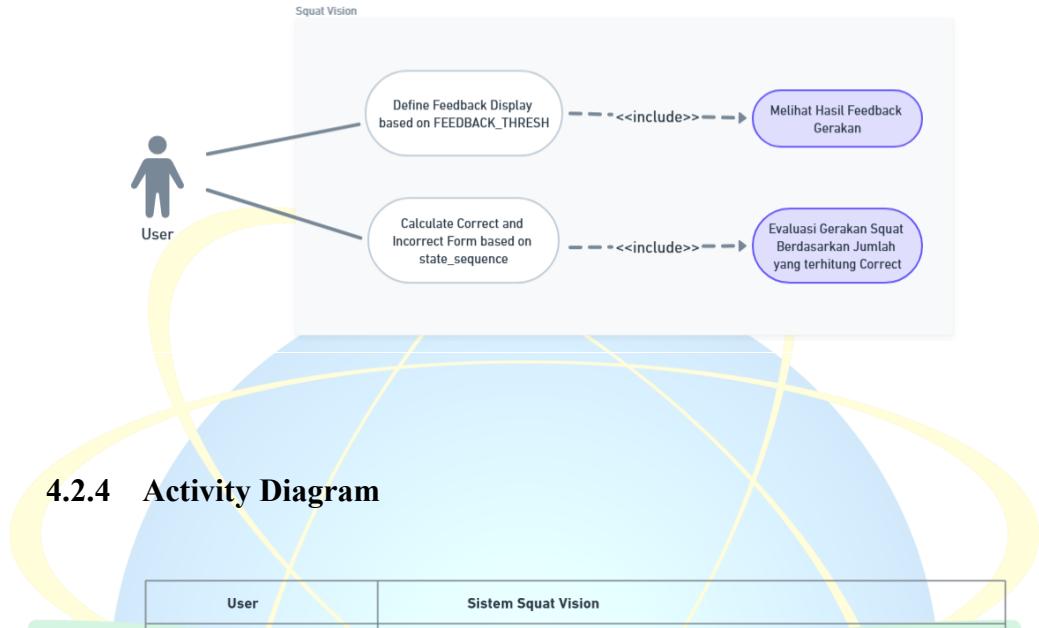
Flowchart Fitness Vision diawali dengan proses pengambilan frame melalui kamera handphone, yang kemudian dievaluasi apakah berhasil membaca frame atau tidak. Jika berhasil, frame akan dikonversi ke format RGB untuk mempermudah pemrosesan. Selanjutnya, sistem melakukan pendekripsi keypoint landmark pada tubuh, seperti bahu (shoulder), pinggul (hip), lutut (knee), dan pergelangan kaki (ankle). Jika deteksi landmark berhasil, sistem menghitung sudut offset antara titik-titik tersebut. Jika nilai sudut offset melebihi threshold (OFFSET_THRESH), maka tampilan peringatan akan ditampilkan kepada pengguna.

Setelah itu, sistem menghitung sudut-sudut kunci berdasarkan posisi hip, knee, dan ankle, serta mengevaluasi apakah gerakan squat sesuai dengan kondisi yang telah ditentukan dalam STATE_THRESH. Berdasarkan hasil evaluasi ini, sistem memberikan feedback secara real-time, baik dalam bentuk peringatan kesalahan maupun konfirmasi jika gerakan dilakukan dengan benar. Proses selanjutnya termasuk penghitungan waktu inaktivitas (INACTIVITY_TIME) untuk memantau apakah pengguna tetap aktif dalam latihan. Jika waktu inaktivitas melebihi batas tertentu (INACTIVE_THRESH), sistem akan mereset perhitungan status gerakan. Terakhir, sistem juga mencatat transisi antar status gerakan untuk analisis lebih lanjut. Proses ini berlangsung secara kontinu hingga pengguna menghentikan aplikasi.

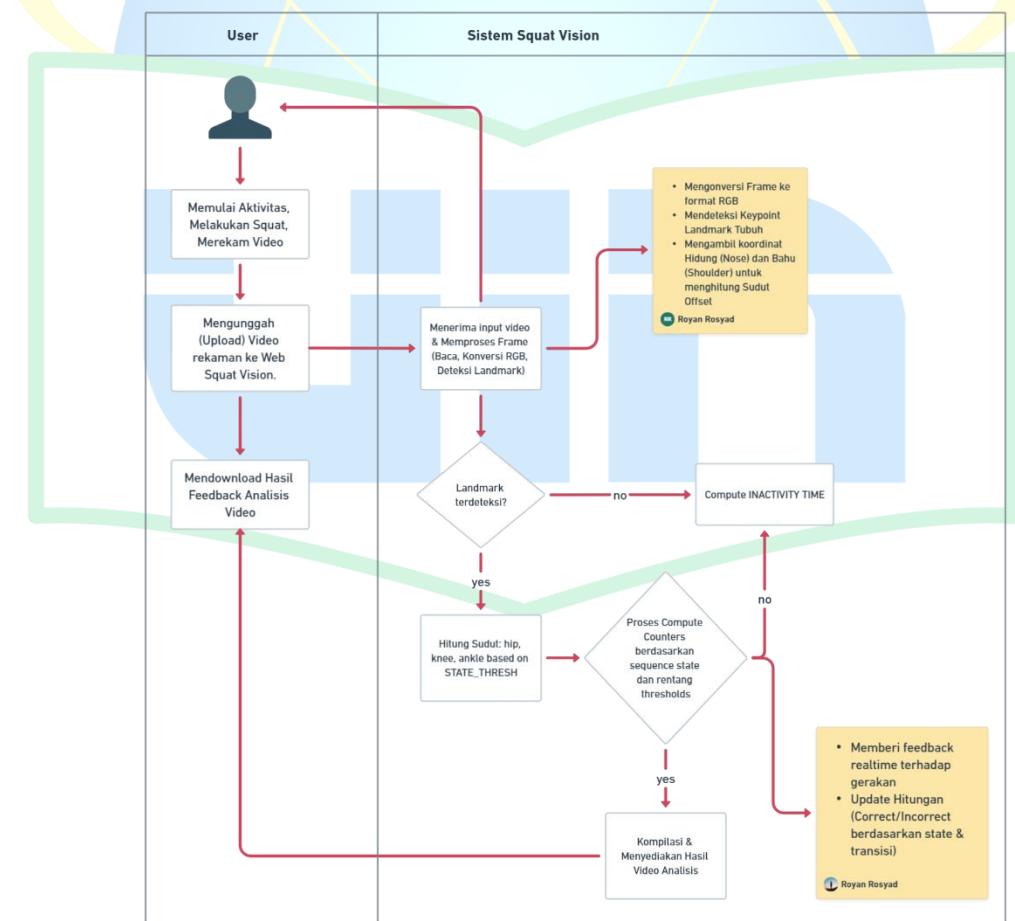
4.2.3 Use Case Diagram

Diagram Use Case ini menggambarkan bagaimana pengguna berinteraksi dengan sistem Fitness Vision, sekaligus menampilkan fungsi-fungsi utama yang tersedia dalam mendekripsi kesalahan gerakan squat. Pengguna berpartisipasi dalam dua use case utama: *Define Feedback Display based on FEEDBACK_THRESH* dan *Calculate Correct and Incorrect Form based on state_sequence*. Pada use case pertama, pengguna dapat menerima umpan balik gerakan secara real-time, berupa peringatan atau konfirmasi yang menunjukkan apakah posisi tubuh saat squat sudah benar atau masih salah. Sementara itu, use case kedua memberikan evaluasi terhadap gerakan squat berdasarkan jumlah repetisi yang dilakukan dengan tepat. Diagram ini juga memperlihatkan hubungan antara kedua use case tersebut, yang

saling melengkapi untuk memberikan gambaran menyeluruh tentang performa latihan pengguna.



4.2.4 Activity Diagram



Activity Diagram pada Gambar 4.5 menggambarkan alur aktivitas antara pengguna dan sistem Fitness Vision dalam proses analisis gerakan squat. Diagram ini dimulai dari pengguna yang melakukan aktivitas squat dan merekam video gerakan, kemudian mengunggah hasil rekaman ke sistem. Sistem menerima input video, mendeteksi pose melalui landmark, menghitung squat, serta membedakan gerakan yang benar dan salah berdasarkan *state_sequence*. Jika terjadi ketidakaktifan, sistem menghitung waktu inaktivitas untuk evaluasi lebih lanjut. Setelah proses selesai, sistem memberikan umpan balik terhadap gerakan pengguna dan menampilkan hasil analisis video yang dapat diunduh oleh pengguna.

4.3 Implementasi

Pada tahap ini proses penulisan code dilakukan menggunakan IDE Visual Studio Code. Bagian ini menjelaskan tahapan implementasi sistem deteksi kesalahan squat, mulai dari penentuan threshold, ekstraksi fitur pose, hingga logika deteksi dan feedback pada pengguna.

4.3.1 Penggunaan Model Mediapipe Pose

Langkah pertama dalam implementasi sistem deteksi kesalahan squat adalah inisialisasi model MediaPipe Pose. MediaPipe Pose merupakan solusi deteksi pose yang dikembangkan oleh Google yang mampu mengestimasi 33 titik landmark pada tubuh manusia. Dalam sistem ini, model MediaPipe Pose menjadi komponen inti yang bertanggung jawab untuk mendeteksi posisi dan orientasi tubuh pengguna saat melakukan gerakan squat.

```
Import cv2
import mediapipe as mp
import numpy as np

def draw_rounded_rect(img, rect_start, rect_end, corner_width,
box_color):

    x1, y1 = rect_start
    x2, y2 = rect_end
    w = corner_width

    # draw filled rectangles
    cv2.rectangle(img, (x1 + w, y1), (x2 - w, y1 + w),
box_color, -1)
    cv2.rectangle(img, (x1 + w, y2 - w), (x2 - w, y2),
box_color, -1)
    cv2.rectangle(img, (x1, y1 + w), (x1 + w, y2 - w),
box_color, -1)
    cv2.rectangle(img, (x2 - w, y1 + w), (x2, y2 - w),
box_color, -1)
```

```

        cv2.rectangle(img, (x1 + w, y1 + w), (x2 - w, y2 - w),
box_color, -1)

    # draw filled ellipses
    cv2.ellipse(img, (x1 + w, y1 + w), (w, w),
                angle = 0, startAngle = -90, endAngle = -180,
color = box_color, thickness = -1)

    cv2.ellipse(img, (x2 - w, y1 + w), (w, w),
                angle = 0, startAngle = 0, endAngle = -90, color
= box_color, thickness = -1)

    cv2.ellipse(img, (x1 + w, y2 - w), (w, w),
                angle = 0, startAngle = 90, endAngle = 180,
color = box_color, thickness = -1)

    cv2.ellipse(img, (x2 - w, y2 - w), (w, w),
                angle = 0, startAngle = 0, endAngle = 90, color
= box_color, thickness = -1)

    return img

def draw_dotted_line(frame, lm_coord, start, end, line_color):
    pix_step = 0

    for i in range(start, end+1, 8):
        cv2.circle(frame, (lm_coord[0], i+pix_step), 2,
line_color, -1, lineType=cv2.LINE_AA)

    return frame

def draw_text(
    img,
    msg,
    width = 8,
    font=cv2.FONT_HERSHEY_SIMPLEX,
    pos=(0, 0),
    font_scale=1,
    font_thickness=2,
    text_color=(0, 255, 0),
    text_color_bg=(0, 0, 0),
    box_offset=(20, 10),
):
    offset = box_offset
    x, y = pos
    text_size, _ = cv2.getTextSize(msg, font, font_scale,
font_thickness)
    text_w, text_h = text_size
    rec_start = tuple(p - o for p, o in zip(pos, offset))
    rec_end = tuple(m + n - o for m, n, o in zip((x + text_w, y
+ text_h), offset, (25, 0)))

```

```

        img = draw_rounded_rect(img, rec_start, rec_end, width,
text_color_bg)

        cv2.putText(
            img,
            msg,
            (int(rec_start[0] + 6), int(y + text_h + font_scale -
1)),
            font,
            font_scale,
            text_color,
            font_thickness,
            cv2.LINE_AA,
        )

    return text_size

def find_angle(p1, p2, ref_pt = np.array([0,0])):
    p1_ref = p1 - ref_pt
    p2_ref = p2 - ref_pt

    cos_theta = (np.dot(p1_ref,p2_ref)) / (1.0 *
np.linalg.norm(p1_ref) * np.linalg.norm(p2_ref))
    theta = np.arccos(np.clip(cos_theta, -1.0, 1.0))

    degree = int(180 / np.pi) * theta

    return int(degree)

def get_landmark_array(pose_landmark, key, frame_width,
frame_height):

    denorm_x = int(pose_landmark[key].x * frame_width)
    denorm_y = int(pose_landmark[key].y * frame_height)

    return np.array([denorm_x, denorm_y])

def get_landmark_features(kp_results, dict_features, feature,
frame_width, frame_height):

    if feature == 'nose':
        return get_landmark_array(kp_results,
dict_features[feature], frame_width, frame_height)

    elif feature == 'left' or 'right':
        shldr_coord = get_landmark_array(kp_results,
dict_features[feature]['shoulder'], frame_width, frame_height)
        elbow_coord = get_landmark_array(kp_results,
dict_features[feature]['elbow'], frame_width, frame_height)
        wrist_coord = get_landmark_array(kp_results,
dict_features[feature]['wrist'], frame_width, frame_height)

```

```

        hip_coord    = get_landmark_array(kp_results,
dict_features[feature]['hip'], frame_width, frame_height)
        knee_coord   = get_landmark_array(kp_results,
dict_features[feature]['knee'], frame_width, frame_height)
        ankle_coord  = get_landmark_array(kp_results,
dict_features[feature]['ankle'], frame_width, frame_height)
        foot_coord   = get_landmark_array(kp_results,
dict_features[feature]['foot'], frame_width, frame_height)

    return shldr_coord, elbow_coord, wrist_coord, hip_coord,
knee_coord, ankle_coord, foot_coord

else:
    raise ValueError("feature needs to be either 'nose',
'left' or 'right')

def get_mediapipe_pose(
    static_image_mode = False,
    model_complexity = 1,
    smooth_landmarks = True,
    min_detection_confidence = 0.5,
    min_tracking_confidence = 0.5
):
    pose = mp.solutions.pose.Pose(
        static_image_mode,
        model_complexity,
        smooth_landmarks,
        min_detection_confidence,
        min_tracking_confidence
    )
    return pose

```

4.3.2 Penggunaan Landmark yang dibutuhkan

Setelah model MediaPipe Pose berhasil diinisialisasi dan digunakan untuk mendekripsi pose pada setiap frame, langkah berikutnya adalah mengekstraksi titik-titik kunci (landmark) tubuh yang relevan untuk analisis gerakan squat. Proses ini sangat penting karena setiap landmark mewakili posisi anatomi tertentu pada tubuh, seperti bahu, siku, pinggul, lutut, pergelangan kaki, dan kaki.

Untuk memudahkan proses ekstraksi dan pemetaan, sistem menggunakan struktur dictionary yang mengelompokkan indeks landmark berdasarkan sisi tubuh (kiri dan kanan) serta titik pusat (nose). Dengan pendekatan ini, pengambilan koordinat landmark dari hasil deteksi MediaPipe menjadi lebih terstruktur dan efisien, sehingga dapat langsung digunakan untuk perhitungan sudut maupun analisis postur.

Berikut adalah kode yang digunakan untuk mendefinisikan dan mengelompokkan indeks landmark utama:

```
# Dictionary to maintain the various landmark features.
self.dict_features = {}
self.left_features = {
    'shoulder': 11,
    'elbow': 13,
    'wrist': 15,
    'hip': 23,
    'knee': 25,
    'ankle': 27,
    'foot': 31
}

self.right_features = {
    'shoulder': 12,
    'elbow': 14,
    'wrist': 16,
    'hip': 24,
    'knee': 26,
    'ankle': 28,
    'foot': 32
}

self.dict_features['left'] = self.left_features
self.dict_features['right'] = self.right_features
self.dict_features['nose'] = 0
```

4.3.3 Perhitungan Sudut antar Landmark dan State Sequence Squat

Pada tahap ini, sistem menghitung sudut antar titik kunci (landmark) tubuh seperti bahu, pinggul, lutut, dan pergelangan kaki. Sudut-sudut ini digunakan untuk menentukan fase gerakan squat dan melakukan state tracking.

Angle Calculation:

```
def find_angle(p1, p2, ref_pt = np.array([0,0])):
    p1_ref = p1 - ref_pt
    p2_ref = p2 - ref_pt
```

```

        cos_theta = (np.dot(p1_ref,p2_ref)) / (1.0 *
np.linalg.norm(p1_ref) * np.linalg.norm(p2_ref))
        theta = np.arccos(np.clip(cos_theta, -1.0, 1.0))

        degree = int(180 / np.pi) * theta

    return int(degree)

```

Urutan State:

```

def _get_state(self, knee_angle):
    knee = None
    if self.thresholds['HIP_KNEE_VERT'][‘NORMAL’][0] <=
knee_angle <= self.thresholds['HIP_KNEE_VERT'][‘NORMAL’][1]:
        knee = 1
    elif self.thresholds['HIP_KNEE_VERT'][‘TRANS’][0] <=
knee_angle <= self.thresholds['HIP_KNEE_VERT'][‘TRANS’][1]:
        knee = 2
    elif self.thresholds['HIP_KNEE_VERT'][‘PASS’][0] <=
knee_angle <= self.thresholds['HIP_KNEE_VERT'][‘PASS’][1]:
        knee = 3
    return f’s{knee}’ if knee else None

def _update_state_sequence(self, state):
    if state == ‘s2’:
        if ((‘s3’ not in self.state_tracker[‘state_seq’]) and
(self.state_tracker[‘state_seq’].count(‘s2’))==0) or \
        ((‘s3’ in self.state_tracker[‘state_seq’]) and
(self.state_tracker[‘state_seq’].count(‘s2’)==1)):
            self.state_tracker[‘state_seq’].append(state)
    elif state == ‘s3’:
        if (state not in self.state_tracker[‘state_seq’]) and
‘s2’ in self.state_tracker[‘state_seq’]:
            self.state_tracker[‘state_seq’].append(state)

```

Fungsi `find_angle()` menggunakan trigonometri dan aljabar vektor untuk menghitung sudut antara dua titik landmark relatif terhadap titik referensi. Proses perhitungannya dapat diuraikan dalam beberapa tahap:

6. Menghitung Vektor Relatif:

Fungsi pertama-tama menghitung vektor relatif dari kedua titik terhadap titik referensi. Misalnya untuk menghitung sudut lutut

- Hip = [100, 150]
- Knee = [100, 250]
- Titik Vertikal = [100, 0]

Sehingga Vektor Relatif nya:

- $p1_ref = \text{Hip} - \text{Knee} = [100, 150] - [100, 250] = [0, -100]$
- $p2_ref = \text{Titik_vertikal} - \text{Knee} = [100, 0] - [100, 250] = [0, -250]$

7. Menghitung Dot Product

Dot product dari kedua vektor dihitung menggunakan rumus:

- $p1_ref \cdot p2_ref = (p1_ref_x \times p2_ref_x) + (p1_ref_y \times p2_ref_y)$
- $\text{dot_product} = (0 \times 0) + (-100 \times -250) = 25000$

A. Menghitung Panjang Vektor:

Panjang (magnitudo) setiap vektor dihitung menggunakan rumus Pythagoras:

$$|v| = \sqrt{(v_x^2 + v_y^2)}$$

Untuk $p1_ref$:

$$|p1_ref| = \sqrt{(0^2 + (-100)^2)} = 100$$

Untuk $p2_ref$:

$$|p2_ref| = \sqrt{(0^2 + (-250)^2)} = 250$$

B. Perhitungan Sudut Cosinus:

Perhitungan nilai kosinus dari sudut (θ) antara dua vektor dapat diperoleh menggunakan persamaan berikut:

$$\cos(\theta) = (p1_ref \cdot p2_ref) / (|p1_ref| \times |p2_ref|)$$

Sebagai ilustrasi perhitungan:

$$\cos(\theta) = 25000 / (100 \times 250) = 1,0$$

4.3.4 Implementasi Thresholds Transisi Gerakan

Threshold digunakan untuk membedakan antara gerakan squat yang benar dan salah berdasarkan sudut-sudut yang telah dihitung. Setiap threshold diatur sesuai mode (Beginner/Pro) dan digunakan dalam logika deteksi kesalahan.

```
# Get thresholds for beginner mode
def get_thresholds_beginner():
```

```

_ANGLE_HIP_KNEE_VERT = {
    'NORMAL' : (0, 32),
    'TRANS' : (35, 65),
    'PASS' : (70, 95)
}

thresholds = {
    'HIP_KNEE_VERT' : _ANGLE_HIP_KNEE_VERT,
    'HIP_THRESH' : [10, 50],
    'ANKLE_THRESH' : 45,
    'KNEE_THRESH' : [50, 70, 95],
    'OFFSET_THRESH' : 35.0,
    'INACTIVE_THRESH' : 15.0,
    'CNT_FRAME_THRESH' : 50,
    # MPJPE visualization settings
    'VISUALIZE_MPJPE_COMPARISON' : False
}

return thresholds
}

# Get thresholds for beginner mode
def get_thresholds_pro():
    _ANGLE_HIP_KNEE_VERT = {
        'NORMAL' : (0, 32),
        'TRANS' : (35, 65),
        'PASS' : (80, 95)
    }

    thresholds = {
        'HIP_KNEE_VERT' : _ANGLE_HIP_KNEE_VERT,
        'HIP_THRESH' : [15, 50],
        'ANKLE_THRESH' : 30,
        'KNEE_THRESH' : [50, 80, 95],
        'OFFSET_THRESH' : 35.0,
        'INACTIVE_THRESH' : 15.0,
        'CNT_FRAME_THRESH' : 50,
        # MPJPE visualization settings
        'VISUALIZE_MPJPE_COMPARISON' : False
    }

    return thresholds
}

```

Penggunaan Threshold dalam Deteksi:

```
else:  
    if hip_vertical_angle > self.thresholds['HIP_THRESH'][1]:  
        self.state_tracker['DISPLAY_TEXT'][0] = True  
  
    elif hip_vertical_angle < self.thresholds['HIP_THRESH'][0]  
and \  
        self.state_tracker['state_seq'].count('s2') == 1:  
        self.state_tracker['DISPLAY_TEXT'][1] = True  
  
    if self.thresholds['KNEE_THRESH'][0] < knee_vertical_angle <  
self.thresholds['KNEE_THRESH'][1] and \  
        self.state_tracker['state_seq'].count('s2') == 1:  
        self.state_tracker['LOWER_HIPS'] = True  
  
    elif knee_vertical_angle >  
self.thresholds['KNEE_THRESH'][2]:  
        self.state_tracker['DISPLAY_TEXT'][3] = True  
        self.state_tracker['INCORRECT_POSTURE'] = True  
  
    if (ankle_vertical_angle > self.thresholds['ANKLE_THRESH']):  
        self.state_tracker['DISPLAY_TEXT'][2] = True  
        self.state_tracker['INCORRECT_POSTURE'] = True
```

4.3.5 Perhitungan Repetisi dan Feedback Visual

Sistem menghitung jumlah repetisi squat yang benar dan salah berdasarkan urutan state yang terdeteksi. Selain itu, feedback visual diberikan secara real-time pada frame video untuk membantu pengguna memperbaiki postur.

```
If current_state == 's1':  
    if len(self.state_tracker['state_seq']) == 3 and not  
self.state_tracker['INCORRECT_POSTURE']:  
        self.state_tracker['SQUAT_COUNT'] += 1  
        play_sound = str(self.state_tracker['SQUAT_COUNT'])  
  
    elif 's2' in self.state_tracker['state_seq'] and  
len(self.state_tracker['state_seq']) == 1:  
        self.state_tracker['IMPROPER_SQUAT'] += 1  
        play_sound = 'incorrect'  
  
elif self.state_tracker['INCORRECT_POSTURE']:
```

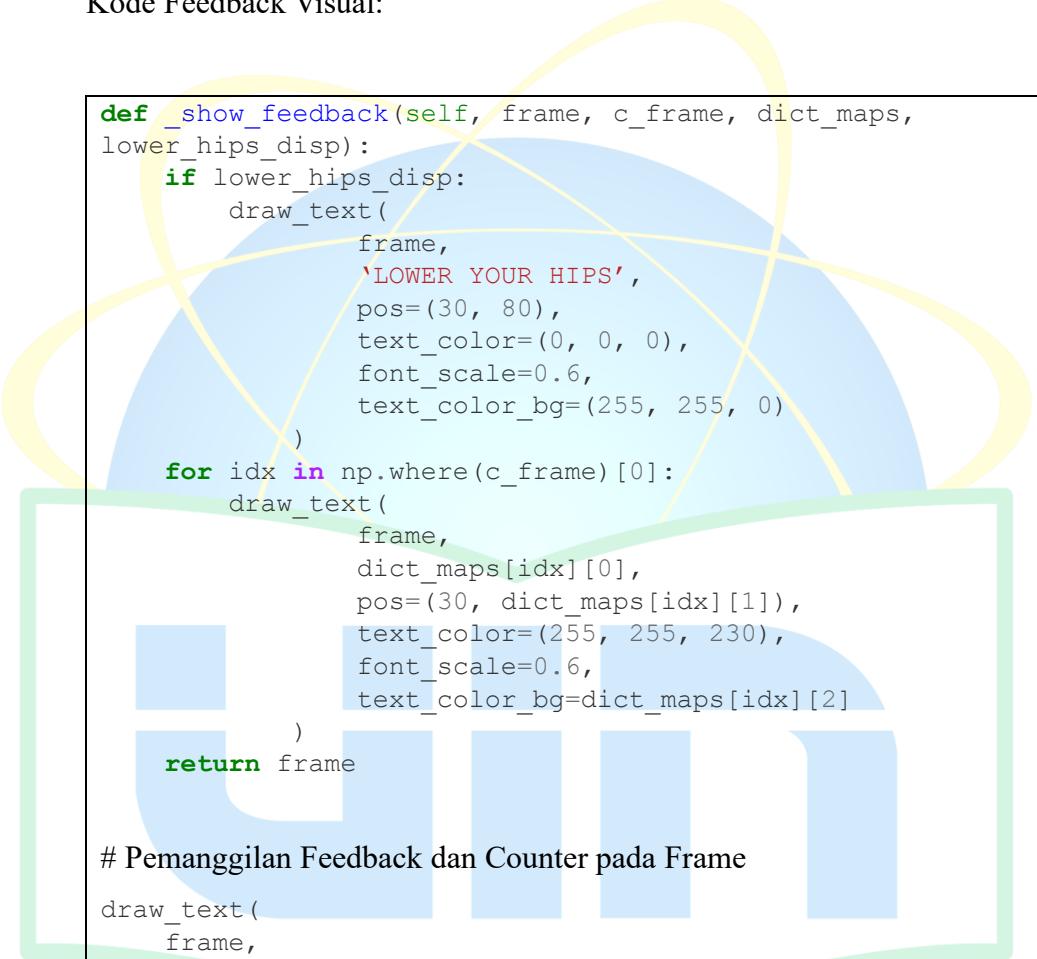
```

        self.state_tracker['IMPROPER_SQUAT']+=1
        play_sound = 'incorrect'

    self.state_tracker['state_seq'] = []
    self.state_tracker['INCORRECT_POSTURE'] = False

```

Kode Feedback Visual:



```

def _show_feedback(self, frame, c_frame, dict_maps,
lower_hips_disp):
    if lower_hips_disp:
        draw_text(
            frame,
            'LOWER YOUR HIPS',
            pos=(30, 80),
            text_color=(0, 0, 0),
            font_scale=0.6,
            text_color_bg=(255, 255, 0)
        )
    for idx in np.where(c_frame)[0]:
        draw_text(
            frame,
            dict_maps[idx][0],
            pos=(30, dict_maps[idx][1]),
            text_color=(255, 255, 230),
            font_scale=0.6,
            text_color_bg=dict_maps[idx][2]
        )
    return frame

# Pemanggilan Feedback dan Counter pada Frame
draw_text(
    frame,
    "CORRECT: " + str(self.state_tracker['SQUAT_COUNT']),
    pos=(int(frame_width*0.68), 30),
    text_color=(255, 255, 230),
    font_scale=0.7,
    text_color_bg=(18, 185, 0)
)

draw_text(
    frame,
    "INCORRECT: " + str(self.state_tracker['IMPROPER_SQUAT']),
    pos=(int(frame_width*0.68), 80),
    text_color=(255, 255, 230),
    font_scale=0.7,
    text_color_bg=(221, 0, 0),
)

# Feedback kesalahan gerakan

```

```

self.FEEDBACK_ID_MAP = {
    0: ('BADAN TERLALU KE BELAKANG', 215, (0, 153, 255)),
    1: ('BADAN TERLALU KE DEPAN', 215, (0, 153, 255)),
    2: ('LUTUT MELEWATI JARI KAKI', 170, (255, 80, 80)),
    3: ('SQUAT TERLALU DALAM', 125, (255, 80, 80))
}

```

4.3.6 Mekanisme Inaktivitas (Reset Counter)

Setelah sistem melakukan perhitungan repetisi squat dan memberikan feedback visual secara real-time, langkah berikutnya adalah mengantisipasi kondisi di mana pengguna berhenti bergerak atau tidak melakukan squat dalam jangka waktu tertentu. Untuk menjaga akurasi penghitungan dan mencegah akumulasi data yang tidak relevan, sistem menerapkan mekanisme reset counter berbasis inaktivitas.

Mekanisme ini bekerja dengan memantau perubahan state gerakan squat. Jika state tidak berubah dalam periode waktu tertentu (berdasarkan threshold INACTIVE_THRESH), maka sistem akan menganggap pengguna sedang tidak aktif dan otomatis mereset penghitung repetisi squat (SQUAT_COUNT) dan squat tidak benar (IMPROPER_SQUAT). Selain itu, sistem juga memberikan notifikasi visual pada frame untuk menginformasikan bahwa counter telah di-reset akibat inaktivitas.

```

# Compute Inactivity

if self.state_tracker['curr_state'] ==
self.state_tracker['prev_state']:
    end_time = time.perf_counter()
    self.state_tracker['INACTIVE_TIME'] += end_time -
self.state_tracker['start_inactive_time']
    self.state_tracker['start_inactive_time'] = end_time

    if self.state_tracker['INACTIVE_TIME'] >=
self.thresholds['INACTIVE_THRESH']:
        self.state_tracker['SQUAT_COUNT'] = 0
        self.state_tracker['IMPROPER_SQUAT'] = 0
        display_inactivity = True

else:
    self.state_tracker['start_inactive_time'] =
time.perf_counter()

```

```

    self.state_tracker['INACTIVE_TIME'] = 0.0

    # Menampilkan notifikasi jika counter di-reset karena
    inaktivitas

    if display_inactivity:
        play_sound = 'reset_counters'
        self.state_tracker['start_inactive_time'] =
time.perf_counter()
        self.state_tracker['INACTIVE_TIME'] = 0.0

```

4.3.7 User Interface Streamlit

Pada sistem ini, antarmuka pengguna dibangun menggunakan Streamlit untuk memudahkan interaksi dan visualisasi hasil analisis squat. Salah satu fitur utama adalah halaman *live analysis* dan upload video, di mana pengguna dapat mengunggah video latihan squat untuk dianalisis secara otomatis.

Tampilan Home:

```

import streamlit as st

st.set_page_config(page_title="Fitness Vision", page_icon="🏋️",
layout="centered")

st.title('Fitness Vision: Analyze Your Squat Technique')
st.markdown("---")
st.markdown("""
### Overview
Fitness Vision adalah aplikasi AI yang membantu Anda
menganalisis teknik squat secara real-time maupun dari video
rekaman. Dengan teknologi pose estimation (MediaPipe), aplikasi
ini memberikan feedback langsung mengenai postur squat Anda dan
mengevaluasi akurasi deteksi pose menggunakan metrik MPJPE (Mean
Per Joint Position Error).

**Fitur utama:**
- Analisis squat secara real-time melalui webcam
- Upload video untuk analisis postur squat
- Evaluasi akurasi pose estimation dengan MPJPE
- Mode Beginner & Pro sesuai kebutuhan
- Visualisasi perbedaan prediksi dan ground truth
""")

st.markdown("""
### Cara Menggunakan
1. Pilih menu **Live Stream** untuk analisis squat secara
langsung menggunakan webcam.
2. Pilih menu **Upload Video** untuk menganalisis video squat
yang sudah direkam.
3. Aktifkan fitur MPJPE untuk melihat evaluasi akurasi pose
estimation.
""")

```

```

4. Kunjungi halaman **MPJPE Analysis** untuk memahami lebih lanjut tentang metrik evaluasi dan interpretasinya.

""")

st.markdown("""
## Quick Navigation
""")
col1, col2, col3 = st.columns(3)
with col1:
    if st.button('📷 Live Stream'):
        st.switch_page('pages/1_📷_Live_Stream.py')
with col2:
    if st.button('📁 Upload Video'):
        st.switch_page('pages/2_📁_Upload_Video.py')
with col3:
    if st.button('📊 MPJPE Analysis'):
        st.switch_page('pages/3_📊_MPJPE_Analysis.py')

st.markdown("""
<sub>Fitness Vision is developed using Mediapipe and OpenCV.</sub>
""", unsafe_allow_html=True)

```

Fitness Vision: Analyze Your Squat Technique

Overview

Fitness Vision adalah aplikasi AI yang membantu Anda menganalisis teknik squat secara real-time maupun dari video rekaman. Dengan teknologi pose estimation (MediaPipe), aplikasi ini memberikan feedback langsung mengenai postur squat Anda dan mengevaluasi akurasi deteksi pose menggunakan metrik MPJPE (Mean Per Joint Position Error).

Fitur utama:

- Analisis squat secara real-time melalui webcam
- Upload video untuk analisis postur squat
- Evaluasi akurasi pose estimation dengan MPJPE
- Mode Beginner & Pro sesuai kebutuhan
- Visualisasi perbedaan prediksi dan ground truth

Cara Menggunakan

- Pilih menu Live Stream untuk analisis squat secara langsung menggunakan webcam.
- Pilih menu Upload Video untuk menganalisis video squat yang sudah direkam.
- Aktifkan fitur MPJPE untuk melihat evaluasi akurasi pose estimation.
- Kunjungi halaman MPJPE Analysis untuk memahami lebih lanjut tentang metrik evaluasi dan interpretasinya.

Quick Navigation

Live Stream Upload Video MPJPE Analysis

Halaman Live Analysis:

```

import streamlit as st
from streamlit_webrtc import VideoHTMLAttributes,
webrtcStreamer

```

```

from aiortc.contrib.media import MediaRecorder
from utils import get_mediapipe_pose
from process_frame import ProcessFrame
from thresholds import get_thresholds_beginner,
get_thresholds_pro

st.title('AI Fitness Trainer: Squats Analysis')

col1, col2 = st.columns(2)
with col1:
    mode = st.radio('Select Mode', ['Beginner', 'Pro'],
horizontal=True)
with col2:
    enable_mpjpe = st.checkbox('Enable MPJPE Evaluation',
value=False)

if mode == 'Beginner':
    thresholds = get_thresholds_beginner()
else:
    thresholds = get_thresholds_pro()

live_process_frame = ProcessFrame(thresholds=thresholds,
flip_frame=True,
evaluate_mpjpe=enable_mpjpe)

pose = get_mediapipe_pose()

def video_frame_callback(frame):
    frame = frame.to_ndarray(format="rgb24")
    frame, _ = live_process_frame.process(frame, pose)
    return av.VideoFrame.from_ndarray(frame, format="rgb24")

ctx = webrtc_streamer(
    key="Squats-pose-analysis",
    video_frame_callback=video_frame_callback,
    rtc_configuration={"iceServers": [{"urls": [
["stun:stun.l.google.com:19302"]]}]},
    media_stream_constraints={"video": {"width": {"min":480,
'ideal':480}}, "audio": False},
    video_html_attrs=VideoHTMLAttributes(autoPlay=True,
controls=False, muted=False)
)

# Menampilkan hasil MPJPE jika diaktifkan
if ctx.state.playing and enable_mpjpe:
    st.markdown("## MPJPE Over Time")
    if len(live_process_frame.mpjpe_values) > 0:
        st.line_chart(live_process_frame.mpjpe_values,
use_container_width=True)

```



Halaman Upload Video:

```
import os
import sys
import streamlit as st
import cv2
import tempfile
import numpy as np

BASE_DIR = os.path.abspath(os.path.join(__file__, '../..'))
sys.path.append(BASE_DIR)

from utils import get_mediapipe_pose
from process_frame import ProcessFrame
from thresholds import get_thresholds_beginner,
get_thresholds_pro

st.title('AI Fitness Trainer: Squats Analysis')

col1, col2 = st.columns(2)
with col1:
    mode = st.radio('Select Mode', ['Beginner', 'Pro'],
horizontal=True)
with col2:
```

```

enable_mpjpe = st.checkbox('Enable MPJPE Evaluation',
                           value=False,
                           help="Mean Per Joint Position
Error - Evaluates pose estimation accuracy")

# MPJPE visualization options if MPJPE is enabled
if enable_mpjpe:
    col1_mpjpe, col2_mpjpe = st.columns(2)
    with col1_mpjpe:
        show_comparison = st.checkbox('Show Prediction vs Ground
Truth', value=False,
                                       help="Visualize the
difference between predicted and ground truth landmarks")
    with col2_mpjpe:
        display_mpjpe = st.checkbox('Display MPJPE on Video',
                                    value=False,
                                    help="Show MPJPE values
directly on the video frame")
else:
    show_comparison = False
    display_mpjpe = False

thresholds = None

if mode == 'Beginner':
    thresholds = get_thresholds_beginner()

elif mode == 'Pro':
    thresholds = get_thresholds_pro()

upload_process_frame = ProcessFrame(thresholds=thresholds,
                                      evaluate_mpjpe=enable_mpjpe,
                                      visualize_comparison=show_comparison,
                                      display_mpjpe=display_mpjpe)

# Initialize face mesh solution
pose = get mediapipe_pose()

download = None

if 'download' not in st.session_state:
    st.session_state['download'] = False

output_video_file = f'output_recorded.mp4'

if os.path.exists(output_video_file):
    os.remove(output_video_file)

with st.form('Upload', clear_on_submit=True):
    up_file = st.file_uploader("Upload a Video", ['mp4', 'mov',
                                                'avi'])

```

```

uploaded = st.form_submit_button("Upload")

stframe = st.empty()

ip_vid_str = '<p style="font-family:Helvetica; font-weight: bold; font-size: 16px;">Input Video</p>'
warning_str = '<p style="font-family:Helvetica; font-weight: bold; color: Red; font-size: 17px;">Please Upload a Video first!!!</p>'

warn = st.empty()

download_button = st.empty()

if up_file and uploaded:

    download_button.empty()
    tfile = tempfile.NamedTemporaryFile(delete=False)

    try:
        warn.empty()
        tfile.write(up_file.read())

        vf = cv2.VideoCapture(tfile.name)

        # ----- Write the processed video frame. -----
        fps = int(vf.get(cv2.CAP_PROP_FPS))
        width = int(vf.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(vf.get(cv2.CAP_PROP_FRAME_HEIGHT))
        frame_size = (width, height)
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')
        video_output = cv2.VideoWriter(output_video_file,
        fourcc, fps, frame_size)
        #
        # -----
    except:
        pass

    txt = st.sidebar.markdown(ip_vid_str,
    unsafe_allow_html=True)
    ip_video = st.sidebar.video(tfile.name)

    while vf.isOpened():
        ret, frame = vf.read()
        if not ret:
            break

        # convert frame from BGR to RGB before processing it.
        Frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        out_frame, _ = upload_process_frame.process(frame,
pose)
        stframe.image(out_frame)
        video_output.write(out_frame[...,:-1])

```

```

vf.release()
video_output.release()

# Show MPJPE statistics if evaluation was enabled
if enable_mpjpe and upload_process_frame.mpjpe_values:
    st.markdown("### MPJPE Over Time")

# Create a line chart with improved styling
st.line_chart(
    upload_process_frame.mpjpe_values,
    use_container_width=True
)

# Display metrics below the chart
col1, col2, col3 = st.columns(3)

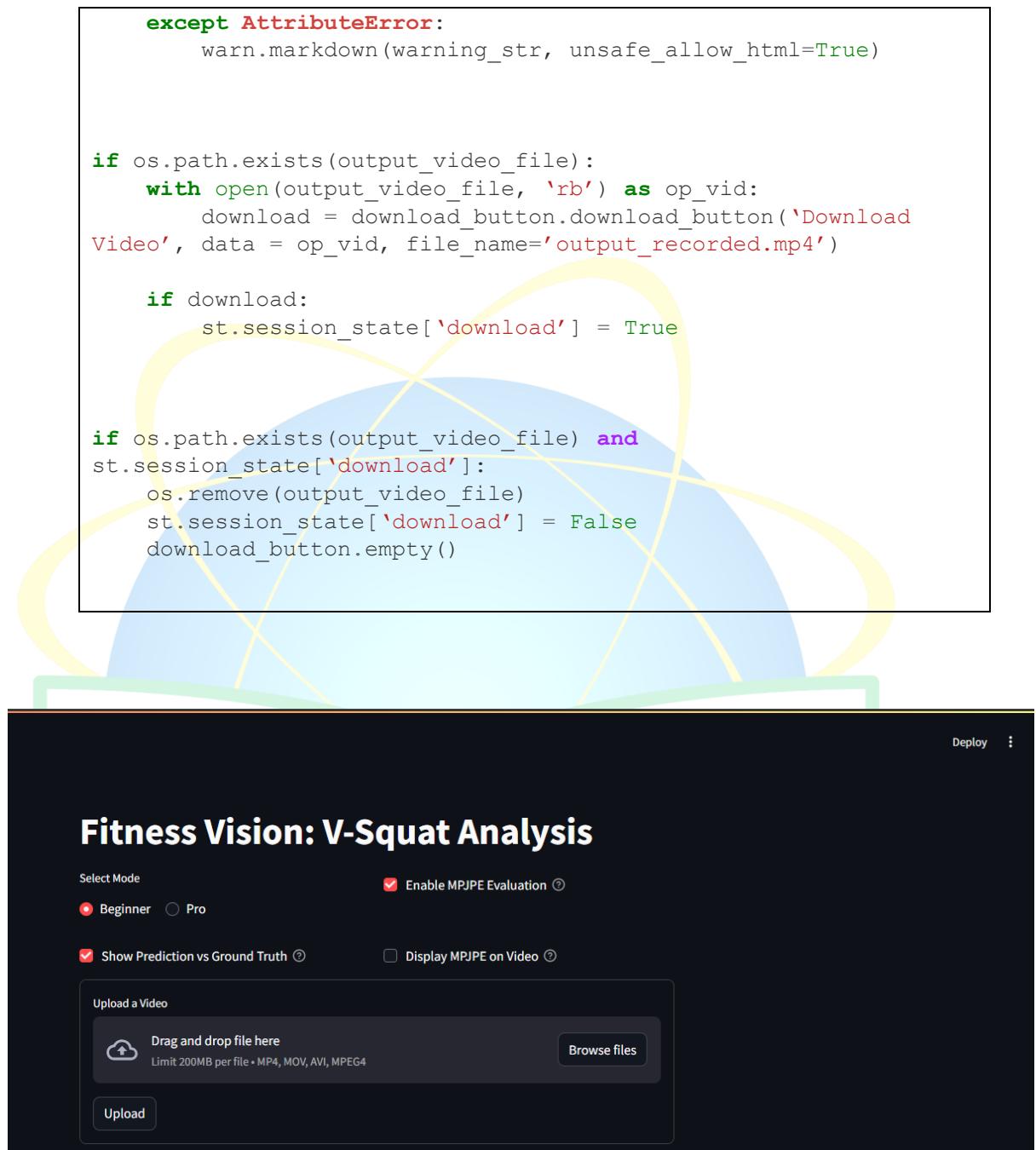
# Calculate statistics
avg_mpjpe =
np.mean(upload_process_frame.mpjpe_values)
min_mpjpe = min(upload_process_frame.mpjpe_values)
if upload_process_frame.mpjpe_values else 0
max_mpjpe = max(upload_process_frame.mpjpe_values)
if upload_process_frame.mpjpe_values else 0

with col1:
    st.metric("Average MPJPE (px)", f"{avg_mpjpe:.2f}")
with col2:
    st.metric("Min MPJPE (px)", f"{min_mpjpe:.2f}")
with col3:
    st.metric("Max MPJPE (px)", f"{max_mpjpe:.2f}")

# Add styling for metrics to match the desired look
st.markdown("""
<style>
    .stMetric {
        background-color: #262730;
        padding: 15px;
        border-radius: 5px;
        margin-bottom: 20px;
    }
    .stMetric label {
        color: #FAFAFA;
    }
    .stMetric .css-1lwivap2 {
        font-size: 42px;
        color: #FFFFFF;
        font-weight: bold;
    }
</style>
""", unsafe_allow_html=True)

stframe.empty()
ip_video.empty()
txt.empty()
tfile.close()

```

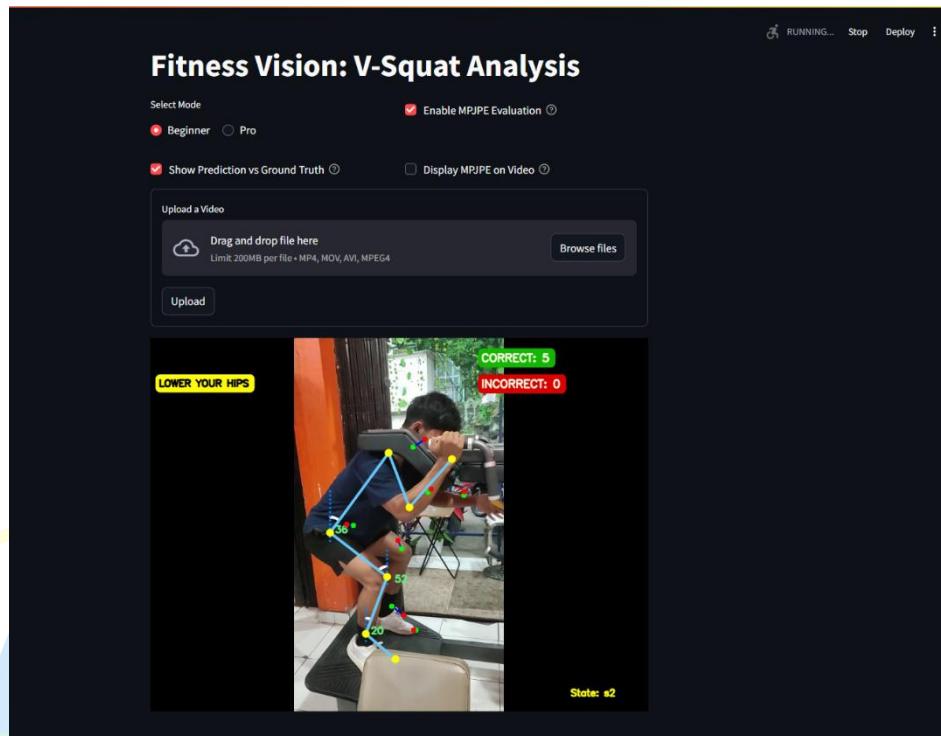


Gambar 4. 8 User Interface Upload Video Fitness Vision

4.4 Testing

Pada tahap ini, dilakukan pengujian menyeluruh terhadap sistem deteksi kesalahan squat yang telah dikembangkan. Pengujian bertujuan untuk memastikan seluruh fitur berjalan dengan baik, memberikan feedback yang akurat, serta mampu beradaptasi pada berbagai kondisi penggunaan.

4.4.1 Pengujian Mode Upload Video (Analisis dari Video Rekaman)



Gambar 4. 9 Preview Pengujian pada menu Upload Video

Pengujian ini bertujuan untuk memastikan bahwa sistem mampu menangani video latihan squat yang diunggah oleh pengguna. Setiap frame dalam video akan dianalisis oleh sistem guna mendeteksi posisi tubuh, menghitung total repetisi, serta menampilkan umpan balik visual mengenai teknik pelaksanaan gerakan.

Pengujian mode upload video menunjukkan bahwa sistem mampu memproses video rekaman dengan efektif, memberikan feedback yang relevan, dan menghasilkan analisis yang akurat. Fitur seperti deteksi pose, perhitungan repetisi, serta umpan balik visual bekerja dengan baik.

4.4.2 Pengujian pada Mode Beginner & Pro

Pengujian dilakukan pada dua mode, yakni Beginner dan Pro, yang masing-masing memiliki *threshold* deteksi berbeda. Mode Beginner dirancang agar lebih toleran terhadap kesalahan postur, sementara mode Pro menetapkan standar yang lebih tinggi dan ketat. Melalui pengujian ini, dipastikan bahwa sistem mampu menyesuaikan tingkat sensitivitas deteksi serta memberikan umpan balik yang relevan sesuai dengan kemampuan pengguna. Pada tahap ini dilakukan pengujian mode Beginner dan Pro pada satu sample video yang sama.

```

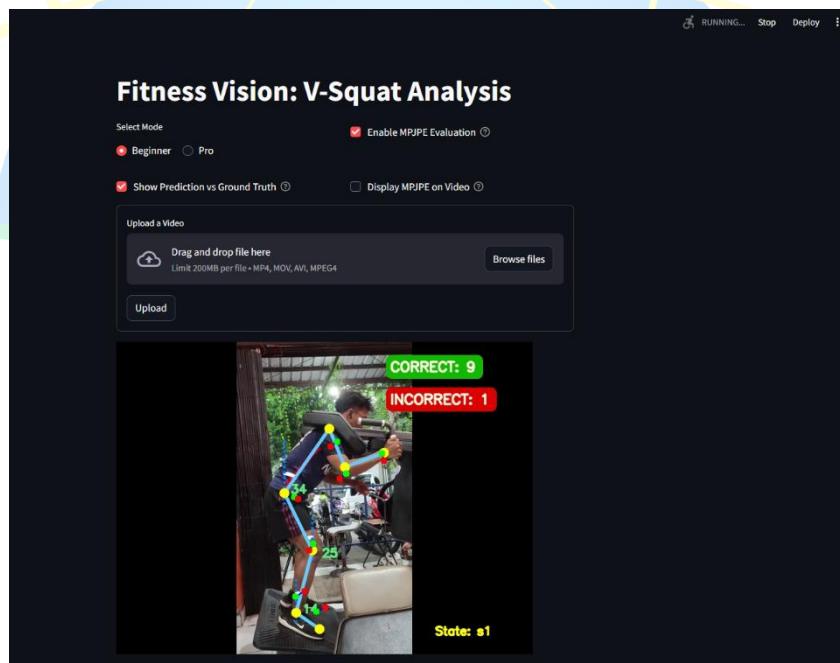
If mode == 'Beginner':
    thresholds = get_thresholds_beginner()
elif mode == 'Pro':
    thresholds = get_thresholds_pro()

upload_process_frame = ProcessFrame(thresholds=thresholds, ...)

```

Kode tersebut menunjukkan pemilihan threshold berdasarkan mode yang dipilih pengguna, sehingga logika deteksi dan feedback akan menyesuaikan.

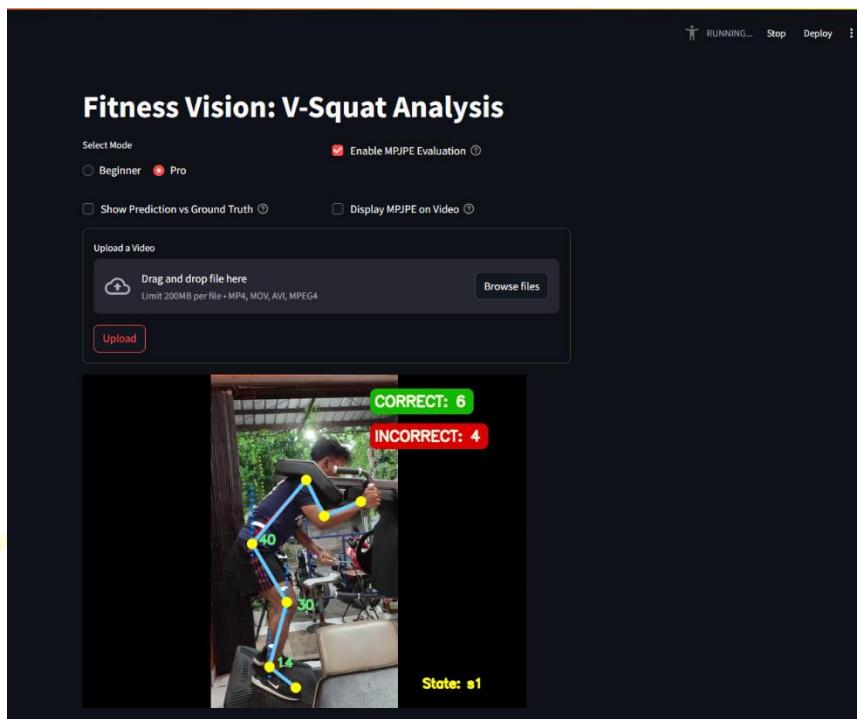
A. Pengujian Mode Beginner :



Gambar 4. 10 Preview Pengujian pada mode Beginner

Dari pengujian mode beginner tersebut didapatkan hasil jumlah gerakan yang benar sejumlah sembilan, dan gerakan yang salah sejumlah satu.

B. Pengujian Mode Pro:



Kemudian dari hasil pengujian mode pro terdapat sebuah perbedaan hasil yakni untuk jumlah gerakan benar yang terhitung sejumlah enam, dan gerakan salah yang terdeteksi sejumlah empat. Hal tersebut menandakan befungsi nya perbedaan penilaian gerakan antara kedua mode tersebut.

4.4.3 Skenario Pengujian pada Berbagai Kondisi

Pengujian dilakukan pada berbagai kondisi lingkungan dan variasi teknik squat untuk memastikan sistem tetap adaptif dan akurat. Berikut rincian variabel yang diuji meliputi pencahayaan, sudut kamera, postur subjek, serta teknik squat benar dan salah

Tabel 4. 2 Tabel Skenario untuk Pengujian Blackbox Testing

No.	Skenario	Feedback Gerakan	Jumlah Pengujian	Ekspektasi
1.	Pendeteksian dengan kondisi cahaya terang dan	“Badan Terlalu ke Belakang”	2 Repetisi	Sistem memberikan feedback sesuai kondisi tersebut jika

	pakaian tidak kontras dengan background			sudut hip-vertical > 45°
		“Badan Terlalu ke Depan”	2 Repetisi	Sistem memberikan feedback sesuai kondisi tersebut jika sudut hip-vertical < 20°
		“Lutut Melewati Jari Kaki”	5 Repetisi	Sistem memberikan feedback sesuai kondisi tersebut jika sudut knee-ankle line > 30°
		“Squat Terlalu Dalam”	5 Repetisi	Sistem memberikan feedback sesuai kondisi tersebut jika sudut hip-knee line > 95°
2.	Pendeteksian dengan kondisi cahaya gelap dan pakaian kontras dengan background	“Badan Terlalu ke Belakang”	2 Repetisi	Sistem memberikan feedback sesuai kondisi tersebut jika sudut hip-vertical > 45°
		“Badan Terlalu ke Depan”	2 Repetisi	Sistem memberikan feedback sesuai kondisi tersebut jika sudut hip-vertical < 20°
		“Lutut Melewati Jari Kaki”	5 Repetisi	Sistem memberikan feedback sesuai kondisi tersebut jika sudut knee-ankle line > 30°

		“Squat Terlalu Dalam”	5 Repetisi	Sistem memberikan feedback sesuai kondisi tersebut jika sudut hip-knee line > 105°
--	--	-----------------------	------------	--

Dengan skenario pengujian ini, diharapkan sistem mampu memberikan feedback yang tepat dan konsisten terhadap empat jenis kesalahan teknik squat pada kedua mode skenario (*Beginner* dan *Pro*). Hasil dari pengujian ini akan menjadi dasar evaluasi akurasi sistem dalam mendeteksi dan mengklasifikasikan setiap jenis kesalahan gerakan.

Tabel 4. 3 Tabel Skenario Pengujian pada Instruktur Ahli

Skenario	Subjek Pengujian	Feedback Gerakan	Jumlah Pengujian	Ekspektasi
Terang	Laki-laki usia 26 tahun	Perfect Squat	10 Repetisi	Dapat menghitung gerakan Correct dan Incorrect dengan baik
	Perempuan usia 24 tahun			
Gelap	Laki-laki usia 23 tahun	Perfect Squat	10 Repetisi	Masih dapat menghitung gerakan Correct dan Incorrect dengan baik
	Perempuan usia 45 tahun			

Berdasarkan tabel skenario pengujian di atas, akan dilakukan pengujian sebanyak 40 kali yang terbagi menjadi 20 kali pengujian untuk masing-masing kondisi pencahayaan terang dan gelap. Pengujian ini melibatkan dua subjek dengan karakteristik usia yang berbeda, yaitu laki-laki usia 25 - 50 tahun dan Perempuan usia 45 tahun, dimana masing-masing akan melakukan 10 repetisi gerakan squat menggunakan *V-Squat Machine*. Pengujian ini bertujuan untuk memvalidasi kehandalan sistem dalam mendeteksi dan memberikan feedback yang akurat pada berbagai kondisi pencahayaan dan karakteristik pengguna yang berbeda.

4.5 Evaluasi

Pada tahap evaluasi, sistem pendekripsi kesalahan pada gerakan squat diuji untuk menilai tingkat akurasi serta konsistensi dalam memberikan *feedback* kepada pengguna. Pengujian dilakukan dengan metode *blackbox testing*, di mana penilaian sistem didasarkan pada respons yang dihasilkan terhadap berbagai skenario gerakan squat yang benar maupun salah. Setiap hasil pengujian dicatat sebagai sesuai atau tidak sesuai dengan harapan, dan tingkat akurasi sistem dihitung berdasarkan persentase keberhasilan dari seluruh skenario yang diuji.

Selain itu, hasil video gerakan juga dianalisis menggunakan metrik Mean Per Joint Position Error (MPJPE) guna menilai ketepatan deteksi pose tubuh. Metrik ini digunakan untuk mengevaluasi seberapa akurat sistem mendekripsi posisi landmark tubuh seperti bahu, pinggul, lutut, dan pergelangan kaki dibandingkan dengan posisi sebenarnya.

