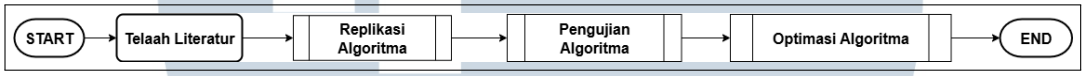
BAB 3
METODOLOGI PENELITIAN

Framework utama dari penelitian pada Gambar 3.1 memperlihatkan seluruh tahapan penelitian yang dilakukan. Penelitian dimulai dengan menelaah literatur dan referensi jurnal, dilanjutkan dengan replikasi algoritma dari jurnal sebelumnya, pengujian algoritma replikasi pada kondisi yang menjadi batasan penelitian sebelumnya, hingga pengembangan dan peningkatan algoritma untuk mengatasi batasan dari penelitian sebelumnya.



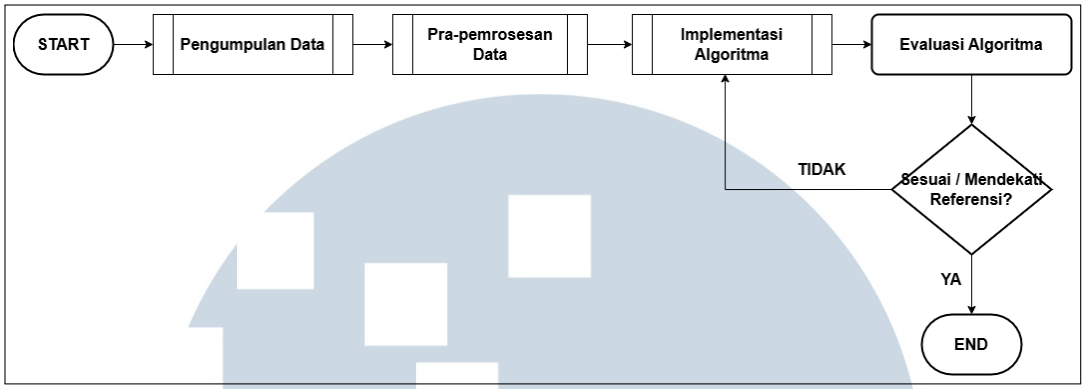
Gambar 3.1. *Framework* Penelitian

3.1 Telaah Literatur

Pada tahapan telaah literatur, hal yang pertama dilakukan adalah mengumpulkan informasi dan menganalisis masalah, tujuan, hasil, dan keterbatasan penelitian sebelumnya. Hal selanjutnya adalah mempelajari teori dan konsep yang akan berguna untuk menyelesaikan masalah pada penelitian sebelumnya dan mengembangkan penelitian. Salah satu batasan dari penelitian sebelumnya adalah sistem deteksi jatuh belum diuji dengan skenario di mana seseorang sedang melakukan matras [13]. Penelitian ini dilakukan untuk menyelesaikan batasan masalah tersebut dan berupaya untuk meningkatkan akurasi sistem deteksi jatuh dalam membedakan kejadian jatuh yang sebenarnya dengan gerakan matras.

3.2 Replikasi Algoritma Deteksi Jatuh

Pada tahapan replikasi algoritma yang dapat dilihat pada Gambar 3.2, hal yang dilakukan pertama kali adalah mengumpulkan *dataset* yang digunakan pada penelitian sebelumnya. *Dataset* yang terkumpul akan memasuki tahap pra-proses sebelum diimplementasikan ke dalam algoritma replika yang dibuat. Setelah algoritma replika berhasil dibuat, algoritma akan dievaluasi sesuai dengan ketentuan dan algoritma penelitian sebelumnya. Apabila hasil dari algoritma replikasi tidak sesuai atau tidak mendekati hasil akurasi algoritma penelitian sebelumnya, maka implementasi algoritma harus diperbaiki agar semakin mendekati hasil yang sesuai.



Gambar 3.2. *Framework* Replikasi algoritma Referensi

3.2.1 Pengumpulan Data untuk Replikasi

Dataset utama yang digunakan untuk proses replikasi algoritma adalah Le2i Fall Dataset. Kumpulan data ini mencakup 130 urutan video yang direkam dari empat lingkungan berbeda, yaitu *Home*, *Coffee Room*, *Office*, dan *Lecture Room*. Setiap video memiliki resolusi 320×240 piksel dan kecepatan 25 *frame* per detik (*FPS*). *Dataset* Le2i Fall secara spesifik terbagi menjadi dua kategori utama, sebagaimana dirinci dalam Tabel 3.1.

Tabel 3.1. Distribusi Kategori Video pada Le2i Fall Dataset

Kategori Video	**Jumlah**
Kejadian Jatuh (*Fall*)	99
Aktivitas Normal (*Non-Fall*)	31
Total	**130**

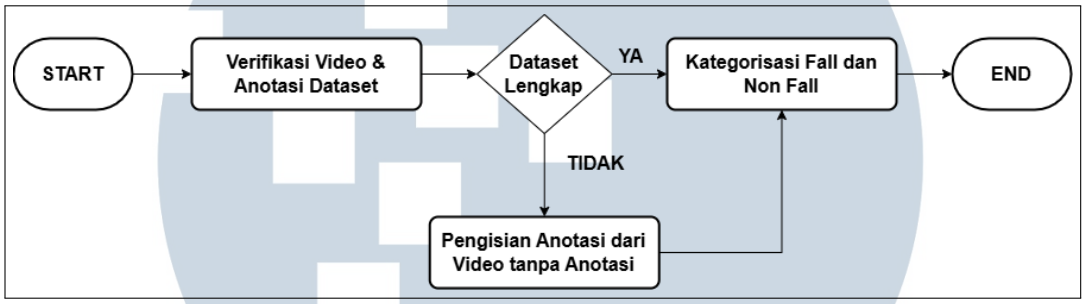
Tautan ke Data Video

Kejadian jatuh dalam *dataset* ini bervariasi, meliputi jatuh ke depan, ke belakang, ke samping, dari berbagai ketinggian, dan dengan tingkat dampak yang berbeda. Sementara itu, aktivitas normal yang direkam mencakup berjalan, duduk atau berdiri dari kursi, membungkuk untuk mengambil objek, mengganti sepatu, dan melakukan peregangan atau latihan. *Dataset* ini juga menyajikan tantangan, seperti kondisi pencahayaan yang bervariasi dan adanya *oklusi*.

3.2.2 Pra-pemrosesan Data untuk Replikasi

Setelah pengumpulan *dataset* dilakukan, tahap berikutnya adalah mempersiapkannya untuk implementasi ke dalam algoritma replika. Proses

pra-pemrosesan ini diawali dengan verifikasi kelengkapan anotasi untuk setiap video dalam *dataset*. Apabila ditemukan video yang tidak memiliki anotasi, pengisian anotasi akan dilakukan secara manual untuk memastikan kelengkapan data. Setelah semua video terverifikasi memiliki anotasi, *dataset* kemudian dikategorikan menjadi dua kelas utama *Fall* dan *Non-Fall*. Alur detail dari tahapan pra-pemrosesan data ini dapat dilihat pada Gambar 3.3.

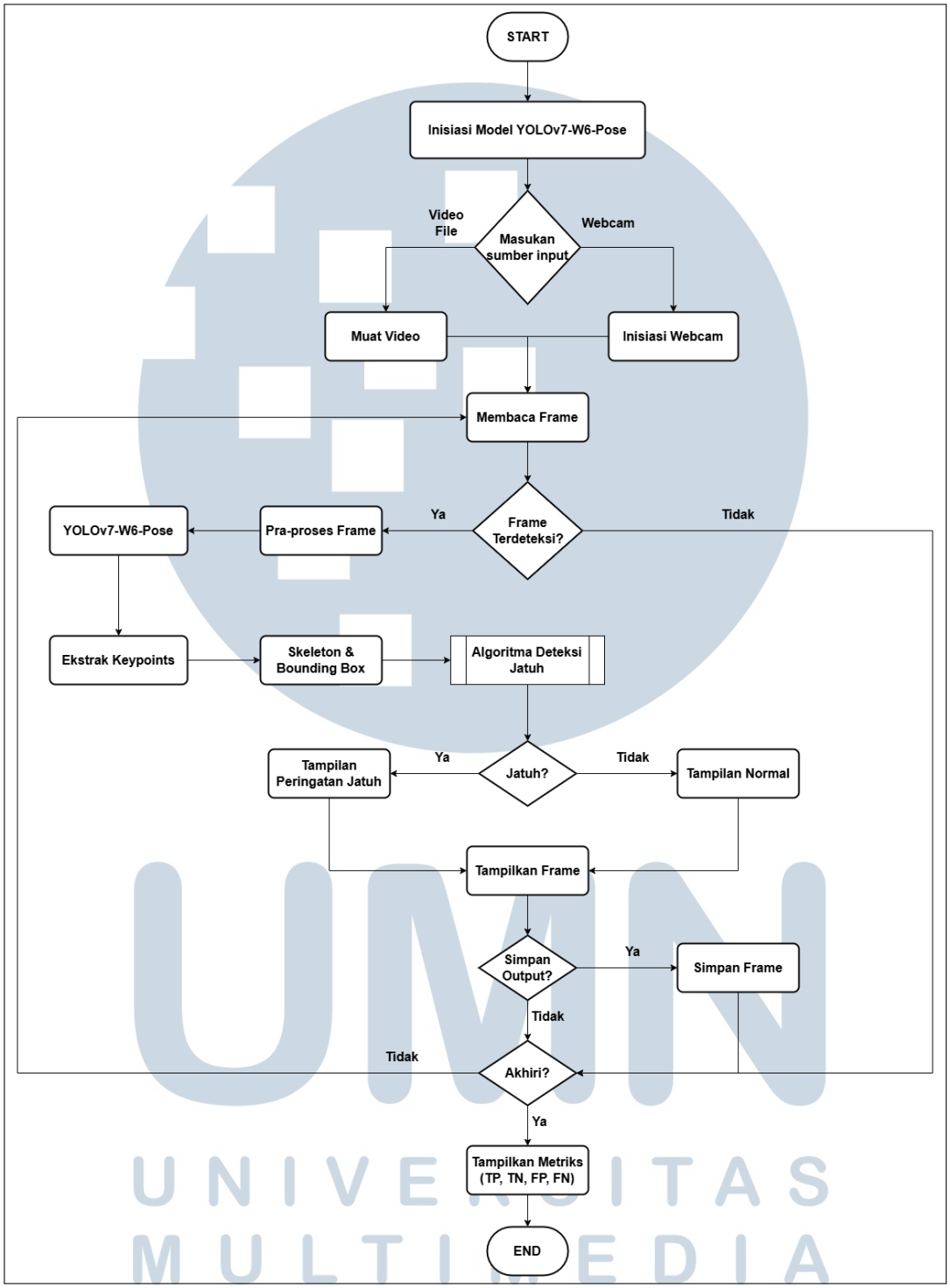


Gambar 3.3. *Flowchart* untuk *Pra-pemrosesan*

3.2.3 Perancangan Algoritma Replikasi

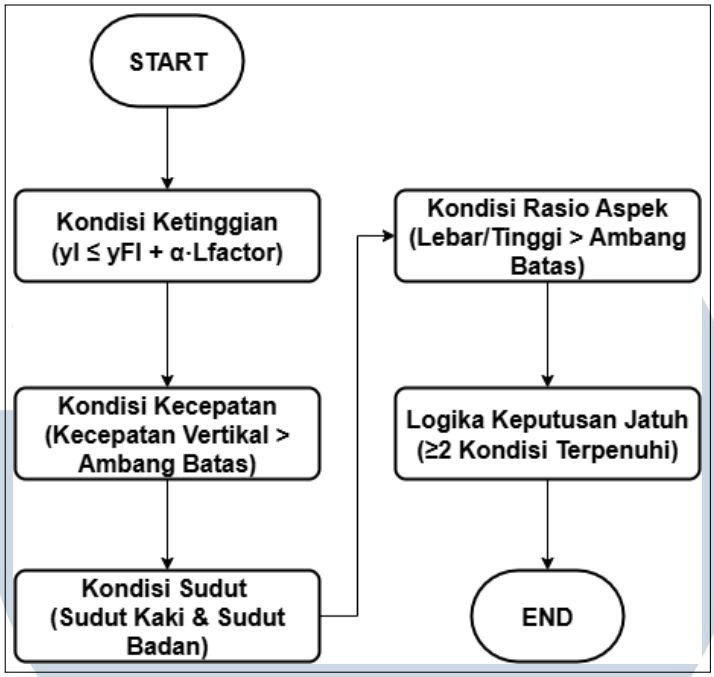
Bagian ini merinci perancangan dan alur kerja dari algoritma replikasi yang dibangun berdasarkan penelitian Tîrziu et al. [13]. Tujuan dari tahap ini adalah untuk menunjukkan tahapan logika deteksi jatuh dari studi yang didapat. Mengingat kode sumber asli tidak tersedia, perancangan ini sepenuhnya didasarkan pada pemahaman terhadap metodologi yang diuraikan dalam jurnal tersebut. Alur kerja dari algoritma replikasi ini, dimulai dari pemrosesan input video hingga evaluasi akhir, dan divisualisasikan secara bertahap dalam *flowchart* pada Gambar 3.4.

Pada Gambar 3.4 menunjukkan detail alur replikasi algoritma deteksi jatuh. Metode dimulai dengan inisialisasi YOLOv7-W6-Pose dan memproses kumpulan data yang akan digunakan. Selanjutnya, sistem menentukan sumber penggunaan input, yaitu antara *webcam* atau video. Selanjutnya, sistem akan memproses *frame* dari video yang sedang diuji. Ini akan mengekstrak *frame* tersebut untuk mendapatkan *keypoints* dan melihat *bounding box*. Jika ada *frame* yang jatuh, sistem akan menampilkan peringatan tentang kejadian jatuh. Setelah proses video selesai, sistem akan menentukan apakah pengujian harus dilanjutkan atau diselesaikan melalui input. Sistem langsung menghitung metrik TP, TN, FP, dan FN jika input dihentikan.



Gambar 3.4. *Flowchart* untuk Proses Pengembangan Algoritma Replikasi

Untuk memberikan pemahaman yang lebih mendalam mengenai proses di dalam proses *module* Algoritma Deteksi Jatuh pada Gambar 3.4, dapat dijelaskan lebih lanjut. Proses ini dapat dilihat pada Gambar 3.5.



Gambar 3.5. *Flowchart* untuk Proses Algoritma Deteksi Jatuh

Berdasarkan pada Gambar 3.5 dapat dilihat proses pertama yaitu sistem memeriksa Kondisi Ketinggian, di mana posisi vertikal bahu dibandingkan dengan posisi kaki yang telah disesuaikan dengan *Length Factor*. Jika kondisi ini mengindikasikan penurunan ketinggian yang tidak wajar, evaluasi dilanjutkan ke Kondisi Kecepatan untuk memeriksa apakah kecepatan vertikal subjek melampaui ambang batas. Selanjutnya, Kondisi Sudut tubuh dianalisis, yang melibatkan pengukuran sudut kaki dan sudut badan untuk mendeteksi postur tubuh yang tidak normal. Secara paralel, Kondisi Rasio Aspek juga dievaluasi untuk menentukan apakah orientasi tubuh (lebar dibanding tinggi) telah melampaui *threshold*, yang mengindikasikan posisi terbaring.

Akhirnya, semua hasil dari kondisi-kondisi tersebut dilakukan secara berurutan dalam Logika Keputusan Jatuh. Sebuah insiden akan dilakukan kategorisasi sebagai jatuh hanya jika minimal dua dari kondisi-kondisi tersebut terpenuhi. Alur logika kondisi ini dirancang untuk meningkatkan keandalan deteksi dan mengurangi potensi alarm palsu dari satu kondisi tunggal.

3.2.4 Implementasi Algoritma Replikasi

Algoritma deteksi jatuh yang dilakukan oleh Tîrziu et al. [13] kemudian diimplementasikan ke dalam bentuk *source code*. Berikut merupakan potongan *source code* untuk implementasi algoritma replikasi.

```python
class FallDetector:
    def __init__(self, poseweights='yolov7-w6-pose.pt', device='0'):
        print(f"Initializing Fall Detector with weights: {poseweights} on device: {device}")

        # Memilih perangkat (CPU atau GPU)
        self.device = select_device(device)
        self.half = self.device.type != 'cpu'

        # Memuat model
        self.model = attempt_load(poseweights, map_location=self.device)
        self.model.eval()

        # Mendefinisikan parameter dan ambang batas
        self.LENGTH_FACTOR_ALPHA = 0.6
        self.VELOCITY_THRESHOLD = 0.8
        self.LEG_ANGLE_THRESHOLD = 50
        self.TORSO_ANGLE_THRESHOLD = 45
        self.ASPECT_RATIO_THRESHOLD = 0.9
        self.CONFIDENCE_THRESHOLD = 0.4
        self.MINIMUM_FALL_FRAMES = 8

        # ... (inisialisasi variabel pelacakan status lainnya)
```

Gambar 3.6. Inisialisasi Sistem dan Pemuatan Model

Pada Gambar 3.6 dapat dilihat tahap pertama dalam alur kerja sistem adalah proses inisialisasi. Pada tahap ini, model YOLOv7-W6-Pose dimuat ke dalam memori, dan seluruh parameter yang relevan, seperti nilai *threshold* untuk deteksi. Proses ini dieksekusi saat `class FallDetector` yang secara otomatis memuat bobot model *(weight file)* dan menyiapkan sistem untuk operasi deteksi.

```python
def run_fall_detection(poseweights='yolov7-w6-pose.pt', source='pose.mp4', ...):
    # ...

    # Parse sumber input
    input_path = source
    if source.isnumeric():
        input_path = int(source) # Menggunakan ID numerik untuk webcam

    # Membuka video capture
    cap = cv2.VideoCapture(input_path)
    if not cap.isOpened():
        # Penanganan error jika video tidak dapat dibuka
        error_msg = f"Error: Could not open video source {source}"
        print(error_msg)
        return
    # ...
```

Gambar 3.7. Pemilihan dan Pemuatan Sumber Video

Pada Gambar 3.7 dapat dilihat setelah tahap inisialisasi model selesai, sistem melanjutkan ke persiapan sumber video. Proses ini dilakukan oleh fungsi

`cv2.VideoCapture()` dari pustaka OpenCV, yang bertanggung jawab untuk membuka dan membaca aliran video. Sesuai dengan implementasi pada fungsi `run_fall_detection`, `cv2.VideoCapture()` dirancang untuk menerima masukan berupa *path* berkas video atau ID numerik dari perangkat *webcam*, berdasarkan pilihan yang diberikan oleh pengguna melalui fungsi `run_interactive()`.

```python
# Di dalam fungsi run_fall_detection()
while cap.isOpened():
    # Membaca Frame
    ret, frame = cap.read()
    if not ret:
        # Jika tidak ada frame lagi, keluar dari loop
        break

    frame_count += 1

    # Pra-Proses Frame (di dalam process_frame)
    # image = cv2.cvtColor(orig_image, cv2.COLOR_BGR2RGB)
    # image = letterbox(image, 640, stride=64, auto=False)[0]
    # image = transforms.ToTensor()(image)
    # ...

    # Memanggil fungsi yang melakukan langkah 4, 5, 6, 7
    processed_frame, is_fall, _, _ = detector.process_frame(frame)

    # ... (lanjutan loop)
```

Gambar 3.8. *Loop* dan Pengambilan *frame*

Pada Gambar 3.8 dapat dilihat setelah sumber video berhasil dilakukan inisialisasi, sistem memasuki *loop* utama untuk memproses video secara *frame-by-frame*. Pada setiap iterasi, fungsi `cap.read()` dipanggil untuk mengambil satu *frame* dari aliran video. Jika sebuah *frame* berhasil diambil (ditandai dengan nilai `ret` yang bernilai *True*), frame tersebut kemudian diteruskan ke fungsi `detector.process_frame()` untuk menjalani tahap pra-pemrosesan, yang meliputi konversi ruang warna, penyesuaian ukuran, dan transformasi ke format *tensor*.

```python
def process_frame(self, frame):
    # Menyalin frame asli agar tidak termodifikasi
    orig_image = frame.copy()
    # Mengonversi warna dari BGR (standar OpenCV) ke RGB (standar Model)
    image = cv2.cvtColor(orig_image, cv2.COLOR_BGR2RGB)

    # Mengubah ukuran gambar ke 640x640 dengan letterboxing untuk menjaga rasio aspek
    frame_height, frame_width = orig_image.shape[:2]
    image = letterbox(image, 640, stride=64, auto=False)[0]

    # Mengonversi gambar ke format Tensor PyTorch dan normalisasi (0-255 -> 0.0-1.0)
    image = transforms.ToTensor()(image)
    image = torch.tensor(np.array([image.numpy()]))
```

Gambar 3.9. Pra-pemrosesan *frame*

Pada Gambar 3.9 dapat dilihat tahap pra-pemrosesan, setiap *frame* yang diambil dari sumber video harus melalui serangkaian transformasi agar sesuai dengan format input yang dibutuhkan oleh model YOLOv7-W6-Pose. Pertama, *frame* asli yang berformat BGR (standar OpenCV) dikonversi menjadi format RGB, yang merupakan standar yang digunakan oleh model PyTorch untuk analisis warna. Selanjutnya, untuk menyeragamkan ukuran input, gambar diubah ukurannya menjadi 640x640 piksel menggunakan fungsi `letterbox`. Fungsi ini mempertahankan rasio aspek asli gambar dengan menambahkan bantalan *(padding)* jika diperlukan, sehingga mencegah kerusakan tampilan yang dapat mengganggu akurasi deteksi. Langkah terakhir adalah mengubah gambar yang telah diolah menjadi format tensor PyTorch melalui `transforms.ToTensor()`, yang juga secara otomatis menormalkan nilai piksel dari rentang 0-255 menjadi rentang 0.0-1.0, sebuah proses standardisasi untuk komputasi model *deep learning*.

```python
def detect_fall(self, keypoints):
    # Definisi indeks keypoint
    NOSE, LEFT_SHOULDER, RIGHT_SHOULDER = 0, 5, 6
    LEFT_HIP, RIGHT_HIP = 11, 12
    LEFT_KNEE, RIGHT_KNEE = 13, 14
    LEFT_ANKLE, RIGHT_ANKLE = 15, 16

    # 1. Ekstraksi dan Validasi Keypoint
    # ... (Kode untuk mengekstrak koordinat dan memeriksa confidence score) ...

    # 2. Kalkulasi Kondisi-Kondisi Deteksi
    # Kondisi Ketinggian (height_cond)
    height_cond = min_shoulder_y >= (max_feet_y - self.LENGTH_FACTOR_ALPHA * Lfactor)

    # Kondisi Kecepatan (speed_cond)
    speed_cond = avg_speed >= self.VELOCITY_THRESHOLD

    # Kondisi Sudut (leg_angle_cond & torso_cond)
    leg_angle_cond = min(left_leg_angle, right_leg_angle) < self.LEG_ANGLE_THRESHOLD
    torso_cond = torso_angle > self.TORSO_ANGLE_THRESHOLD

    # Kondisi Rasio Aspek (aspect_cond)
    aspect_cond = orientation_ratio > self.ASPECT_RATIO_THRESHOLD

    # 3. Agregasi dan Pengambilan Keputusan
    conditions_met = sum([height_cond, speed_cond, leg_angle_cond, torso_cond, aspect_cond])
    is_fall = conditions_met >= 2

    self.fall_buffer.append(is_fall)
    final_detection = sum(self.fall_buffer) >= 2 if len(self.fall_buffer) >= 1 else is_fall

    return final_detection, current_state, conditions_info
```

Gambar 3.10. Eksekusi Algoritma Inti Deteksi Jatuh

Pada Gambar 3.10 dapat dilihat tahap selanjutnya merupakan langkah inti dalam alur kerja ini, yang merepresentasikan proses ”Algoritma Deteksi Jatuh” pada *flowchart*. Pada tahap ini, metode `detect_fall()` dieksekusi, yang berfungsi sebagai inti pemrosesan dari sistem. Metode ini menerima masukan berupa koordinat *keypoints* dari setiap individu yang terdeteksi, kemudian menganalisis postur dan dinamika gerakan berdasarkan serangkaian aturan yang telah didefinisikan untuk menghasilkan keputusan biner (*Fall* atau *Non Fall*) pada *frame* tersebut.

```python
# Di dalam method FallDetector.process_frame(), setelah ekstraksi keypoint
# Loop untuk setiap orang yang terdeteksi
for idx in range(output.shape[0]):
    key_points = output[idx, 7:]

    # Memanggil algoritma deteksi jatuh
    person_fall, person_state, person_conditions = self.detect_fall(key_points)

    # Visualisasi berdasarkan hasil
    if person_fall:
        is_fall = True
        status_text = "FALL DETECTED!"
        cv2.putText(img, status_text, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        cv2.rectangle(img, (int(xmin), int(ymin)), (int(xmax), int(ymax)), (0, 0, 255), 2)
    else:
        status_text = f"State: {person_state}"
        cv2.putText(img, status_text, (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1)
        cv2.rectangle(img, (int(xmin), int(ymin)), (int(xmax), int(ymax)), (0, 255, 0), 1)

    # Menggambar skeleton
    plot_skeleton_kpts(img, output[idx, 7:].T, 3)
```

Gambar 3.11. Analisis Jatuh dan Visualisasi Hasil

Pada Gambar 3.11 dapat dilihat untuk setiap individu yang terdeteksi dalam *frame*, koordinat *keypoints*-nya akan dianalisis oleh metode `self.detect_fall()`. Hasil analisis yang berupa keputusan biner (`person_fall`) kemudian digunakan untuk menentukan jenis umpan balik visual yang akan digambar pada *frame*. Visualisasi ini mencakup tiga komponen utama yaitu penggambaran kerangka tubuh melalui `plot_skeleton_kpts`, penambahan *bounding box* dengan warna yang sesuai (merah untuk kondisi jatuh dan hijau untuk kondisi normal), serta penyematan teks yang mengindikasikan status deteksi.

```python
# Di dalam loop 'while' pada fungsi run_fall_detection()
    # ... (setelah memanggil detector.process_frame())

    # Menampilkan frame ke layar
    if display:
        cv2.imshow('Fall Detection', processed_frame_resized)

    # Menyimpan frame ke video output
    if save_output and out is not None:
        out.write(processed_frame_resized)

    # Pengecekan kondisi akhir (input 'q' dari user)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break
```

Gambar 3.12. Penayangan, Penyimpanan, dan Iterasi

Pada Gambar 3.12 dapat dilihat setelah proses anotasi visual selesai, *frame* yang telah diberikan informasi deteksi tersebut akan ditampilkan pada layar pengguna. Apabila opsi penyimpanan diaktifkan, *frame* yang sama juga akan disimpan sebagai bagian dari berkas video keluaran *(output)*. Selanjutnya, sistem akan memasuki fase jeda singkat untuk mendeteksi masukan dari papan ketik melalui fungsi `cv2.waitKey(1)`. Jika pengguna menekan tombol 'q', loop pemrosesan akan dihentikan. Namun, jika tidak ada interupsi, siklus akan berlanjut ke iterasi berikutnya, kembali ke tahap pembacaan *frame* selanjutnya.

```python
# Setelah loop 'while' pada fungsi run_fall_detection() berakhir
cap.release()
if save_output and out is not None:
    out.release()
cv2.destroyAllWindows()

# ... (Kode untuk menghitung TP, TN, FP, FN, accuracy, precision, recall, f1_score) ...

# Menampilkan statistik akhir
if frame_count > 0 and not batch_mode and not interactive_mode:
    # ...
    if is_le2i:
        print("\\nDetection Metrics:")
        print(f"True Positives: {true_positives}")
        print(f"True Negatives: {true_negatives}")
        # ... (dan metrik lainnya)
```

Gambar 3.13. Agregasi dan Tampilan Metrik Akhir

Setelah seluruh *frame* video selesai diproses dan *loop* berakhir, sistem akan menjalankan tahap finalisasi. Pada tahap ini, semua sumber daya yang terkait dengan video dilepaskan (`cap.release()`) dan seluruh antarmuka yang dibuka oleh OpenCV akan ditutup. Sebagai langkah terakhir, jika data *ground truth* tersedia, sistem akan menghitung metrik performa secara keseluruhan dan menampilkan hasilnya pada konsol sebelum program menghentikan eksekusi.

3.2.5 Hasil Replikasi Algoritma Deteksi Jatuh

Replikasi algoritma deteksi jatuh dilakukan menggunakan Le2i Fall Dataset. *Dataset* ini terdiri dari total 130 urutan video, yang mencakup 99 video kategori *Fall* dan 31 video kategori *Non-Fall* seperti yang dilakukan pada penelitian Tîrziu et al[13]. Kinerja algoritma dinilai berdasarkan metrik klasifikasi seperti *Accuracy*, *Precision*, *Recall*, dan *F1-score*, yang semuanya dihitung dari *confusion matrix*. Hasil evaluasi ini berfungsi sebagai *benchmark* untuk mengidentifikasi secara kuantitatif area-area limitasi algoritma replikasi, terutama pada skenario yang berpotensi menyebabkan *false positive* atau *false negative*.

Tabel 3.2 berikut menyajikan detail hasil deteksi untuk setiap 130 *video* dari Le2i Fall Dataset.

Tabel 3.2. Hasil Deteksi dari Replikasi Algoritma untuk Dataset Le2i

No	Environment	Nama Video	Kategori	Hasil Deteksi
1	Coffee_room_01	video (1).avi	Fall	Fall (TP)
2	Coffee_room_01	video (2).avi	Fall	Fall (TP)
3	Coffee_room_01	video (3).avi	Fall	Fall (TP)
4	Coffee_room_01	video (4).avi	Fall	Fall (TP)
5	Coffee_room_01	video (5).avi	Fall	Fall (TP)
6	Coffee_room_01	video (6).avi	Fall	Fall (TP)
7	Coffee_room_01	video (7).avi	Fall	Fall (TP)
8	Coffee_room_01	video (10).avi	Fall	Fall (TP)
9	Coffee_room_01	video (11).avi	Fall	Fall (TP)
10	Coffee_room_01	video (12).avi	Fall	Fall (TP)
11	Coffee_room_01	video (13).avi	Fall	Fall (TP)
12	Coffee_room_01	video (16).avi	Fall	Not Fall (FN)
13	Coffee_room_01	video (20).avi	Fall	Fall (TP)
Lanjut pada halaman berikutnya				

Tabel 3.2 Hasil Deteksi dari Replikasi Algoritma untuk Dataset Le2i (lanjutan)

No	Environment	Nama Video	Kategori	Hasil Deteksi
14	Coffee_room_01	video (21).avi	Fall	Fall (TP)
15	Coffee_room_01	video (22).avi	Fall	Fall (TP)
16	Coffee_room_01	video (23).avi	Fall	Fall (TP)
17	Coffee_room_01	video (25).avi	Fall	Fall (TP)
18	Coffee_room_01	video (26).avi	Fall	Fall (TP)
19	Coffee_room_01	video (27).avi	Fall	Fall (TP)
20	Coffee_room_01	video (28).avi	Fall	Fall (TP)
21	Coffee_room_01	video (31).avi	Fall	Not Fall (FN)
22	Coffee_room_01	video (32).avi	Fall	Fall (TP)
23	Coffee_room_01	video (33).avi	Fall	Fall (TP)
24	Coffee_room_01	video (35).avi	Fall	Fall (TP)
25	Coffee_room_01	video (36).avi	Fall	Fall (TP)
26	Coffee_room_01	video (38).avi	Fall	Fall (TP)
27	Coffee_room_01	video (39).avi	Fall	Fall (TP)
28	Coffee_room_01	video (41).avi	Fall	Fall (TP)
29	Coffee_room_01	video (42).avi	Fall	Fall (TP)
30	Coffee_room_01	video (43).avi	Fall	Fall (TP)
31	Coffee_room_01	video (44).avi	Fall	Fall (TP)
32	Coffee_room_01	video (45).avi	Fall	Fall (TP)
33	Coffee_room_01	video (46).avi	Fall	Fall (TP)
34	Coffee_room_01	video (47).avi	Fall	Fall (TP)
35	Coffee_room_01	video (48).avi	Fall	Fall (TP)
36	Coffee_room_02	video (49).avi	Fall	Fall (TP)
37	Coffee_room_02	video (51).avi	Fall	Fall (TP)
38	Coffee_room_02	video (52).avi	Fall	Fall (TP)
39	Coffee_room_02	video (53).avi	Fall	Fall (TP)
40	Coffee_room_02	video (55).avi	Fall	Fall (TP)
41	Coffee_room_02	video (56).avi	Fall	Fall (TP)
42	Coffee_room_02	video (57).avi	Fall	Fall (TP)
43	Coffee_room_02	video (59).avi	Fall	Not Fall (FN)
44	Coffee_room_02	video (60).avi	Fall	Fall (TP)
45	Coffee_room_02	video (62).avi	Fall	Fall (TP)

Lanjut pada halaman berikutnya

Tabel 3.2 Hasil Deteksi dari Replikasi Algoritma untuk Dataset Le2i (lanjutan)

No	Environment	Nama Video	Kategori	Hasil Deteksi
46	Coffee_room_02	video (64).avi	Fall	Fall (TP)
47	Home_01	video (1).avi	Fall	Fall (TP)
48	Home_01	video (2).avi	Fall	Fall (TP)
49	Home_01	video (3).avi	Fall	Fall (TP)
50	Home_01	video (4).avi	Fall	Fall (TP)
51	Home_01	video (5).avi	Fall	Fall (TP)
52	Home_01	video (6).avi	Fall	Fall (TP)
53	Home_01	video (8).avi	Fall	Fall (TP)
54	Home_01	video (9).avi	Fall	Fall (TP)
55	Home_01	video (10).avi	Fall	Fall (TP)
56	Home_01	video (15).avi	Fall	Not Fall (FN)
57	Home_01	video (18).avi	Fall	Fall (TP)
58	Home_01	video (19).avi	Fall	Fall (TP)
59	Home_01	video (20).avi	Fall	Fall (TP)
60	Home_01	video (21).avi	Fall	Fall (TP)
61	Home_01	video (22).avi	Fall	Fall (TP)
62	Home_01	video (23).avi	Fall	Fall (TP)
63	Home_01	video (24).avi	Fall	Fall (TP)
64	Home_01	video (25).avi	Fall	Fall (TP)
65	Home_01	video (26).avi	Fall	Fall (TP)
66	Home_01	video (27).avi	Fall	Fall (TP)
67	Home_01	video (28).avi	Fall	Fall (TP)
68	Home_01	video (29).avi	Fall	Fall (TP)
69	Home_01	video (30).avi	Fall	Fall (TP)
70	Home_02	video (31).avi	Fall	Fall (TP)
71	Home_02	video (32).avi	Fall	Fall (TP)
72	Home_02	video (33).avi	Fall	Fall (TP)
73	Home_02	video (34).avi	Fall	Fall (TP)
74	Home_02	video (35).avi	Fall	Not Fall (FN)
75	Home_02	video (36).avi	Fall	Not Fall (FN)
76	Home_02	video (37).avi	Fall	Fall (TP)
77	Lecture_room	video (2).avi	Fall	Fall (TP)
78	Lecture_room	video (3).avi	Fall	Fall (TP)

Lanjut pada halaman berikutnya

Tabel 3.2 Hasil Deteksi dari Replikasi Algoritma untuk Dataset Le2i (lanjutan)

No	Environment	Nama Video	Kategori	Hasil Deteksi
79	Lecture_room	video (4).avi	Fall	Fall (TP)
80	Lecture_room	video (5).avi	Fall	Fall (TP)
81	Lecture_room	video (6).avi	Fall	Fall (TP)
82	Lecture_room	video (7).avi	Fall	Fall (TP)
83	Lecture_room	video (8).avi	Fall	Fall (TP)
84	Lecture_room	video (10).avi	Fall	Not Fall (FN)
85	Lecture_room	video (11).avi	Fall	Not Fall (FN)
86	Lecture_room	video (12).avi	Fall	Fall (TP)
87	Lecture_room	video (13).avi	Fall	Fall (TP)
88	Lecture_room	video (14).avi	Fall	Fall (TP)
89	Office	video (2).avi	Fall	Fall (TP)
90	Office	video (3).avi	Fall	Not Fall (FN)
91	Office	video (4).avi	Fall	Fall (TP)
92	Office	video (6).avi	Fall	Fall (TP)
93	Office	video (7).avi	Fall	Fall (TP)
94	Office	video (12).avi	Fall	Fall (TP)
95	Office	video (13).avi	Fall	Fall (TP)
96	Office	video (14).avi	Fall	Fall (TP)
97	Office	video (15).avi	Fall	Fall (TP)
98	Office	video (16).avi	Fall	Fall (TP)
99	Office	video (17).avi	Fall	Fall (TP)
100	Coffee_room_02	video (63).avi	Non Fall	Not Fall (TN)
101	Coffee_room_02	video (66).avi	Non Fall	Not Fall (TN)
102	Coffee_room_02	video (67).avi	Non Fall	Not Fall (TN)
103	Coffee_room_02	video (69).avi	Non Fall	Not Fall (TN)
104	Home_02	video (38).avi	Non Fall	Not Fall (TN)
105	Home_02	video (39).avi	Non Fall	Not Fall (TN)
106	Home_02	video (40).avi	Non Fall	Not Fall (TN)
107	Home_02	video (41).avi	Non Fall	Not Fall (TN)
108	Home_02	video (42).avi	Non Fall	Not Fall (TN)
109	Home_02	video (43).avi	Non Fall	Not Fall (TN)
110	Home_02	video (44).avi	Non Fall	Not Fall (TN)
111	Home_02	video (45).avi	Non Fall	Not Fall (TN)

Lanjut pada halaman berikutnya

Tabel 3.2 Hasil Deteksi dari Replikasi Algoritma untuk Dataset Le2i (lanjutan)

No	Environment	Nama Video	Kategori	Hasil Deteksi
112	Home_02	video (46).avi	Non Fall	Not Fall (TN)
113	Home_02	video (47).avi	Non Fall	Not Fall (TN)
114	Lecture_room	video (15).avi	Non Fall	Not Fall (TN)
115	Lecture_room	video (16).avi	Non Fall	Not Fall (TN)
116	Lecture_room	video (18).avi	Non Fall	Not Fall (TN)
117	Lecture_room	video (19).avi	Non Fall	Not Fall (TN)
118	Lecture_room	video (20).avi	Non Fall	Not Fall (TN)
119	Lecture_room	video (21).avi	Non Fall	Not Fall (TN)
120	Lecture_room	video (22).avi	Non Fall	Fall (FP)
121	Lecture_room	video (23).avi	Non Fall	Not Fall (TN)
122	Lecture_room	video (25).avi	Non Fall	Not Fall (TN)
123	Lecture_room	video (26).avi	Non Fall	Not Fall (TN)
124	Office	video (18).avi	Non Fall	Not Fall (TN)
125	Office	video (21).avi	Non Fall	Not Fall (TN)
126	Office	video (22).avi	Non Fall	Not Fall (TN)
127	Office	video (25).avi	Non Fall	Not Fall (TN)
128	Office	video (26).avi	Non Fall	Not Fall (TN)
129	Office	video (28).avi	Non Fall	Not Fall (TN)
130	Office	video (29).avi	Non Fall	Not Fall (TN)

Tautan ke Data Video

Tabel 3.3 merangkum metrik performa keseluruhan dari algoritma replikasi pada *dataset* pada Le2i Fall Dataset.

Tabel 3.3. Hasil Evaluasi Keseluruhan dari Replikasi Algoritma untuk Dataset Le2i

Hasil Keseluruhan	
Metrik	**Nilai**
Total Video	130
True Positive (TP)	90
True Negative (TN)	30
False Positive (FP)	1
False Negative (FN)	9
Accuracy	92.31%
Precision	98.90%
Recall	90.91%
F1 Score	94.74%

Berdasarkan hasil dari replikasi algoritma yang dapat dilihat pada Tabel 3.3, dapat dilakukan perbandingan. Hasil yang dilaporkan oleh Tîrziu et al. [13] pada Le2i Fall Dataset yang sama (dengan *accuracy* 96.15%, *precision* 97%, *recall* 97.98%, dan *f1-score* 97.48%), hasil replikasi ini menunjukkan tingkat kesamaan yang cukup tinggi, meskipun terdapat sedikit variasi. Perbedaan disebabkan oleh keterbatasan informasi terkait dengan parameter dan pilihan dataset yang digunakan untuk pengujian algoritma. Namun, secara keseluruhan, hasil ini mengindikasikan bahwa proses replikasi algoritma telah berhasil mereproduksi kinerja algoritma referensi dengan baik sebagai *benchmark* awal.

3.3 Pengujian Replikasi Algoritma pada Dataset Olahraga Matras

Bagian ini berfokus pada pengujian algoritma hasil replikasi secara lebih spesifik, terutama untuk mengidentifikasi dan mengonfirmasi limitasi kinerja pada skenario yang telah dibahas dalam telaah literatur, yaitu aktivitas olahraga matras yang berpotensi menimbulkan *false alarm*.

3.3.1 Pengumpulan Data untuk Dataset Olahraga Matras

Pada tahap pengumpulan data untuk menguji limitasi ini, *dataset* dikumpulkan menggunakan *script* yang dibangun menggunakan bahasa *python* dan diambil dari *platform* YouTube. *Dataset* yang dipilih secara spesifik terdiri dari video individu yang melakukan kegiatan olahraga matras. Secara keseluruhan, total

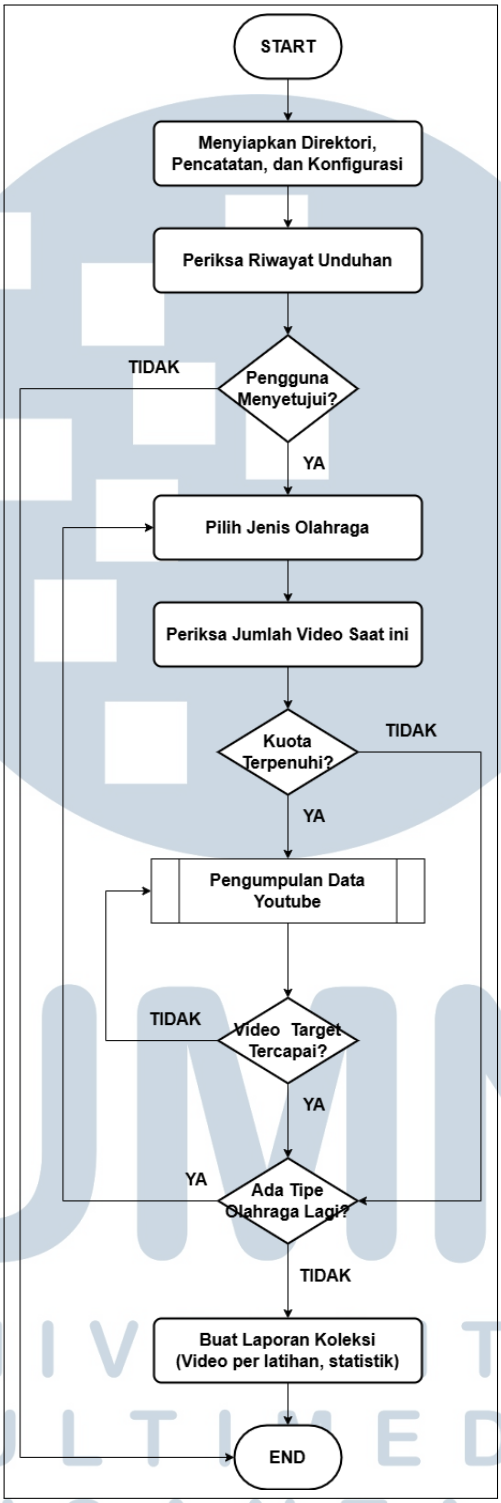
130 video dari berbagai jenis aktivitas olahraga matras yang bisa dilihat pada Tabel 3.4.

Tabel 3.4. Distribusi Jumlah Video per Jenis Aktivitas Olahraga

Jenis Aktivitas Olahraga	**Jumlah Video**
Abdominal curl	14
Bird dog	15
Butterfly back	14
Chest press	15
Hamstring curl	14
Oblique curl	15
Push up	14
Reverse curl	15
Side leg raise	14
Total	**130**

Tautan ke Data Video

Pada Gambar 3.14 dapat dilihat *flowchart* untuk proses pembangunan *script* yang berfungsi untuk mendapatkan video dari YouTube.



Gambar 3.14. *Flowchart* untuk Proses Pembangunan *script* Youtube

Berdasarkan Gambar 3.14 dapat dilihat tahapan untuk membangun *dataset* olahraga matras yang digunakan dalam pengujian limitasi. *Script* ini dikembangkan untuk mengumpulkan data video dari platform YouTube. Alur kerja dari skrip ini

dirancang untuk memastikan proses pengumpulan data berjalan otomatis, terhindar dari duplikasi, dan terdokumentasi dengan baik. Berikut adalah penjelasan tahapan proses berdasarkan *flowchart* yang ditampilkan:

1. Inisialisasi Sistem (START)

   Proses diawali dengan tahap persiapan sistem. Pada langkah ini, *script* secara otomatis menyiapkan direktori atau folder tujuan untuk menyimpan video yang akan diunduh, menginisialisasi berkas pencatatan *(log file)* untuk melacak riwayat unduhan, dan memuat konfigurasi yang diperlukan.

2. Pemeriksaan Riwayat

   Sebelum memulai pengunduhan baru, sistem akan memeriksa riwayat unduhan yang tersimpan dalam *log file*. Langkah ini dilakukan untuk mencegah pengunduhan ulang video yang sudah ada, sehingga memastikan tidak ada data duplikat dalam *dataset*.

3. Konfirmasi Pengguna

   Sistem akan meminta persetujuan dari pengguna untuk melanjutkan proses. Tahap ini memberikan kontrol kepada peneliti untuk memulai atau menghentikan proses pengumpulan data.

4. Pemilihan Kategori Data

   Jika pengguna setuju, sistem akan meminta pengguna untuk memilih jenis olahraga yang ingin dikumpulkan videonya.

5. Pemeriksaan Kuota Awal

   Setelah kategori dipilih, sistem akan memeriksa jumlah video yang sudah ada di direktori untuk kategori tersebut dan membandingkannya dengan kuota atau target yang telah ditentukan sebelumnya.

6. *Loop* Pengumpulan Data

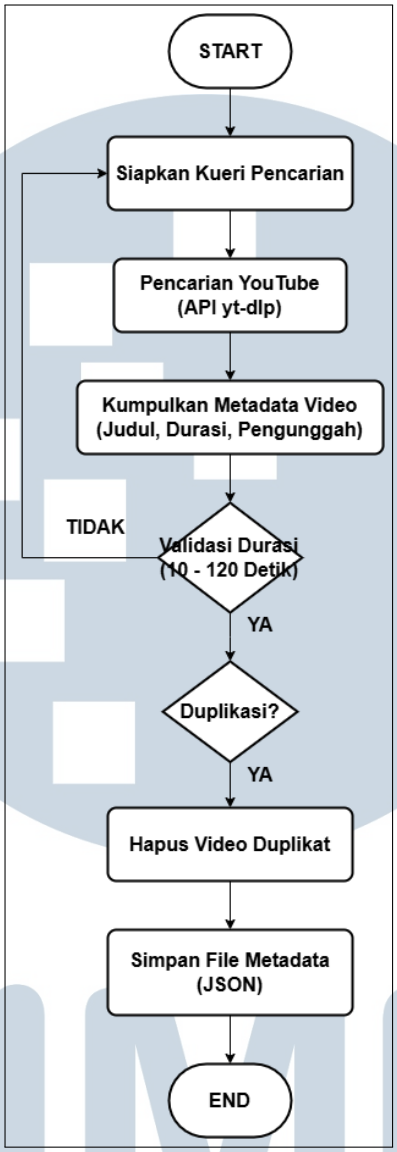
   Jika kuota untuk jenis olahraga yang dipilih belum terpenuhi, sistem akan masuk ke dalam modul utama, yaitu Pengumpulan Data dari YouTube. Modul ini akan mencari, memfilter, dan mengunduh video yang relevan dari YouTube sesuai dengan kategori yang dipilih.

   Setelah satu sesi pengunduhan selesai, sistem akan kembali memeriksa apakah video target telah tercapai. Jika belum, proses pengumpulan data untuk kategori yang sama akan diulang hingga kuota terpenuhi.

7. Pembuatan Laporan (END)

   Setelah semua proses pengumpulan data selesai, sistem akan secara otomatis membuat laporan koleksi data. Laporan ini berisi rincian penting seperti jumlah total video per jenis latihan dan statistik relevan lainnya (misalnya, total durasi, jumlah video yang berhasil diunduh, dll.). Setelah laporan dibuat, proses berakhir.

Proses yang terjadi di dalam tahap ”Pengumpulan Data YouTube” itu sendiri dapat diperinci lebih lanjut. Sub-alur kerja ini, yang diilustrasikan pada Gambar 3.15, memastikan bahwa setiap video yang dikumpulkan relevan, unik, dan terdokumentasi dengan baik. Berikut adalah rincian dari setiap langkah pada sub-alur tersebut.



Gambar 3.15. *Flowchart* untuk Proses Modul dari Pengumpulan Data Youtube

Berikut merupakan penjelasan dari Gambar 3.15 untuk proses modul dari pengumpulan data YouTube:

1. Persiapan Kueri Pencarian *(Prepare Search Queries)*

   Tahap pertama adalah menyiapkan berbagai variasi kueri atau kata kunci pencarian untuk setiap jenis olahraga. Tujuannya adalah untuk memastikan cakupan hasil pencarian video yang luas dan beragam, tidak hanya terpaku pada satu frasa.

2. Eksekusi Pencarian via API

Script kemudian melakukan pencarian di platform YouTube secara otomatis dengan memanfaatkan API dari `yt-dlp`. *Tools* ini memungkinkan *script* untuk berinteraksi dengan YouTube dan mengambil daftar video yang sesuai dengan kueri yang telah disiapkan.

3. Pengumpulan Metadata Video

   Dari hasil pencarian, *script* akan mengumpulkan *metadata* (informasi deskriptif) yang relevan dari setiap video. *Metadata* yang diambil meliputi judul video, durasi, dan nama pengunggah *(uploader)*.

4. Validasi dan Filtrasi Data

   Setiap video yang ditemukan akan melalui dua tahap filtrasi:

   (a) Validasi Durasi: Sistem akan memeriksa apakah durasi video berada dalam rentang yang telah ditentukan, yaitu antara 10 hingga 120 detik. Hal ini bertujuan untuk menyaring video agar hanya klip pendek yang relevan yang diambil, bukan video latihan berdurasi panjang.

   (b) Pemeriksaan Duplikasi: Sistem akan memverifikasi apakah video yang sama sudah pernah diunduh sebelumnya. Jika terdeteksi sebagai duplikat, video tersebut akan dilewati atau dihapus dari antrean unduhan untuk memastikan keunikan setiap data dalam *dataset*.

5. Penyimpanan Metadata (JSON)

   Untuk setiap video yang lolos tahap validasi dan filtrasi, *metadata*-nya akan disimpan dalam sebuah berkas berformat JSON. Penyimpanan ini sangat penting untuk tujuan dokumentasi, analisis, dan memastikan proses pengumpulan data dapat dilihat kembali.

6. Proses Selesai (END)

   Setelah semua langkah tersebut selesai untuk satu siklus pencarian, sub-alur ini berakhir dan akan kembali ke alur utama untuk memeriksa apakah kuota video sudah terpenuhi.

3.3.2 Hasil Replikasi Algoritma Deteksi Jatuh pada Dataset Olahraga Matras

Setelah melakukan replikasi algoritma deteksi jatuh pada Le2i Fall Dataset, langkah selanjutnya adalah menguji algoritma replikasi ini secara spesifik pada

skenario limitasi, yaitu aktivitas olahraga di matras. Pengujian ini bertujuan untuk mengonfirmasi sejauh mana algoritma replikasi dapat membedakan antara gerakan jatuh yang sebenarnya dengan aktivitas yang menyerupai jatuh, yang berpotensi menimbulkan *false alarm*. Total 130 *video* yang disiapkan dari 9 jenis olahraga meliputi *push up, chest press, butterfly back, bird dog, abdominal curl, reverse curl, oblique curl, side leg raise,* dan *hamstring curl* yang diproses untuk pengujian ini.

Tabel 3.5 berikut menyajikan detail hasil deteksi untuk setiap *video* yang diproses dari *dataset* limitasi, termasuk kategori aktivitas, nama *video*, dan hasil deteksi algoritma replikasi.

Tabel 3.5. Hasil Deteksi dari Replikasi Algoritma untuk Olahraga Matras

No	Environment	Nama Video	Kategori	Hasil Deteksi
1	Abdominal_curl	video (1).mp4	Non Fall	Fall (FP)
2	Abdominal_curl	video (2).mp4	Non Fall	Fall (FP)
3	Abdominal_curl	video (3).mp4	Non Fall	Fall (FP)
4	Abdominal_curl	video (4).mp4	Non Fall	Fall (FP)
5	Abdominal_curl	video (5).mp4	Non Fall	Fall (FP)
6	Abdominal_curl	video (6).mp4	Non Fall	Fall (FP)
7	Abdominal_curl	video (7).mp4	Non Fall	Fall (FP)
8	Abdominal_curl	video (8).mp4	Non Fall	Fall (FP)
9	Abdominal_curl	video (9).mp4	Non Fall	Fall (FP)
10	Abdominal_curl	video (10).mp4	Non Fall	Fall (FP)
11	Abdominal_curl	video (11).mp4	Non Fall	Fall (FP)
12	Abdominal_curl	video (12).mp4	Non Fall	Fall (FP)
13	Abdominal_curl	video (13).mp4	Non Fall	Fall (FP)
14	Abdominal_curl	video (14).mp4	Non Fall	Fall (FP)
15	Bird_dog	video (1).mp4	Non Fall	Fall (FP)
16	Bird_dog	video (2).mp4	Non Fall	Fall (FP)
17	Bird_dog	video (3).mp4	Non Fall	Fall (FP)
18	Bird_dog	video (4).mp4	Non Fall	Fall (FP)
19	Bird_dog	video (5).mp4	Non Fall	Fall (FP)
20	Bird_dog	video (6).mp4	Non Fall	Fall (FP)
21	Bird_dog	video (7).mp4	Non Fall	Fall (FP)
22	Bird_dog	video (8).mp4	Non Fall	Fall (FP)
23	Bird_dog	video (9).mp4	Non Fall	Fall (FP)

Lanjut pada halaman berikutnya

Tabel 3.5 Hasil Deteksi dari Replikasi Algoritma untuk Olahraga Matras (lanjutan)

No	Environment	Nama Video	Kategori	Hasil Deteksi
24	Bird_dog	video (10).mp4	Non Fall	Fall (FP)
25	Bird_dog	video (11).mp4	Non Fall	Fall (FP)
26	Bird_dog	video (12).mp4	Non Fall	Fall (FP)
27	Bird_dog	video (13).mp4	Non Fall	Fall (FP)
28	Bird_dog	video (14).mp4	Non Fall	Fall (FP)
29	Bird_dog	video (15).mp4	Non Fall	Fall (FP)
30	Butterfly_back	video (1).mp4	Non Fall	Fall (FP)
31	Butterfly_back	video (2).mp4	Non Fall	Not Fall (TN)
32	Butterfly_back	video (3).mp4	Non Fall	Not Fall (TN)
33	Butterfly_back	video (4).mp4	Non Fall	Fall (FP)
34	Butterfly_back	video (5).mp4	Non Fall	Fall (FP)
35	Butterfly_back	video (6).mp4	Non Fall	Fall (FP)
36	Butterfly_back	video (7).mp4	Non Fall	Fall (FP)
37	Butterfly_back	video (8).mp4	Non Fall	Fall (FP)
38	Butterfly_back	video (9).mp4	Non Fall	Fall (FP)
39	Butterfly_back	video (10).mp4	Non Fall	Fall (FP)
40	Butterfly_back	video (11).mp4	Non Fall	Fall (FP)
41	Butterfly_back	video (12).mp4	Non Fall	Not Fall (TN)
42	Butterfly_back	video (13).mp4	Non Fall	Fall (FP)
43	Butterfly_back	video (14).mp4	Non Fall	Fall (FP)
44	Chest_press	video (1).mp4	Non Fall	Fall (FP)
45	Chest_press	video (2).mp4	Non Fall	Fall (FP)
46	Chest_press	video (3).mp4	Non Fall	Fall (FP)
47	Chest_press	video (4).mp4	Non Fall	Fall (FP)
48	Chest_press	video (5).mp4	Non Fall	Fall (FP)
49	Chest_press	video (6).mp4	Non Fall	Fall (FP)
50	Chest_press	video (7).mp4	Non Fall	Fall (FP)
51	Chest_press	video (8).mp4	Non Fall	Fall (FP)
52	Chest_press	video (9).mp4	Non Fall	Fall (FP)
53	Chest_press	video (10).mp4	Non Fall	Fall (FP)
54	Chest_press	video (11).mp4	Non Fall	Fall (FP)
55	Chest_press	video (12).mp4	Non Fall	Fall (FP)
56	Chest_press	video (13).mp4	Non Fall	Fall (FP)

Lanjut pada halaman berikutnya

Tabel 3.5 Hasil Deteksi dari Replikasi Algoritma untuk Olahraga Matras (lanjutan)

No	Environment	Nama Video	Kategori	Hasil Deteksi
57	Chest_press	video (14).mp4	Non Fall	Fall (FP)
58	Chest_press	video (15).mp4	Non Fall	Fall (FP)
59	Hamstring_curl	video (1).mp4	Non Fall	Fall (FP)
60	Hamstring_curl	video (2).mp4	Non Fall	Fall (FP)
61	Hamstring_curl	video (3).mp4	Non Fall	Fall (FP)
62	Hamstring_curl	video (4).mp4	Non Fall	Fall (FP)
63	Hamstring_curl	video (5).mp4	Non Fall	Fall (FP)
64	Hamstring_curl	video (6).mp4	Non Fall	Fall (FP)
65	Hamstring_curl	video (7).mp4	Non Fall	Fall (FP)
66	Hamstring_curl	video (8).mp4	Non Fall	Fall (FP)
67	Hamstring_curl	video (9).mp4	Non Fall	Fall (FP)
68	Hamstring_curl	video (10).mp4	Non Fall	Fall (FP)
69	Hamstring_curl	video (11).mp4	Non Fall	Fall (FP)
70	Hamstring_curl	video (12).mp4	Non Fall	Fall (FP)
71	Hamstring_curl	video (13).mp4	Non Fall	Fall (FP)
72	Hamstring_curl	video (14).mp4	Non Fall	Fall (FP)
73	Oblique_curl	video (1).mp4	Non Fall	Fall (FP)
74	Oblique_curl	video (2).mp4	Non Fall	Fall (FP)
75	Oblique_curl	video (3).mp4	Non Fall	Fall (FP)
76	Oblique_curl	video (4).mp4	Non Fall	Fall (FP)
77	Oblique_curl	video (5).mp4	Non Fall	Fall (FP)
78	Oblique_curl	video (6).mp4	Non Fall	Fall (FP)
79	Oblique_curl	video (7).mp4	Non Fall	Fall (FP)
80	Oblique_curl	video (8).mp4	Non Fall	Fall (FP)
81	Oblique_curl	video (9).mp4	Non Fall	Fall (FP)
82	Oblique_curl	video (10).mp4	Non Fall	Fall (FP)
83	Oblique_curl	video (11).mp4	Non Fall	Fall (FP)
84	Oblique_curl	video (12).mp4	Non Fall	Fall (FP)
85	Oblique_curl	video (13).mp4	Non Fall	Fall (FP)
86	Oblique_curl	video (14).mp4	Non Fall	Fall (FP)
87	Oblique_curl	video (15).mp4	Non Fall	Fall (FP)
88	Push_up	video (1).mp4	Non Fall	Fall (FP)
89	Push_up	video (2).mp4	Non Fall	Fall (FP)

Lanjut pada halaman berikutnya

Tabel 3.5 Hasil Deteksi dari Replikasi Algoritma untuk Olahraga Matras (lanjutan)

No	Environment	Nama Video	Kategori	Hasil Deteksi
90	Push_up	video (3).mp4	Non Fall	Not Fall (TN)
91	Push_up	video (4).mp4	Non Fall	Fall (FP)
92	Push_up	video (5).mp4	Non Fall	Fall (FP)
93	Push_up	video (6).mp4	Non Fall	Fall (FP)
94	Push_up	video (7).mp4	Non Fall	Fall (FP)
95	Push_up	video (8).mp4	Non Fall	Not Fall (TN)
96	Push_up	video (9).mp4	Non Fall	Fall (FP)
97	Push_up	video (10).mp4	Non Fall	Fall (FP)
98	Push_up	video (11).mp4	Non Fall	Fall (FP)
99	Push_up	video (12).mp4	Non Fall	Fall (FP)
100	Push_up	video (13).mp4	Non Fall	Fall (FP)
101	Push_up	video (14).mp4	Non Fall	Fall (FP)
102	Reverse_curl	video (1).mp4	Non Fall	Fall (FP)
103	Reverse_curl	video (2).mp4	Non Fall	Fall (FP)
104	Reverse_curl	video (3).mp4	Non Fall	Not Fall (TN)
105	Reverse_curl	video (4).mp4	Non Fall	Fall (FP)
106	Reverse_curl	video (5).mp4	Non Fall	Not Fall (TN)
107	Reverse_curl	video (6).mp4	Non Fall	Fall (FP)
108	Reverse_curl	video (7).mp4	Non Fall	Fall (FP)
109	Reverse_curl	video (8).mp4	Non Fall	Fall (FP)
110	Reverse_curl	video (9).mp4	Non Fall	Fall (FP)
111	Reverse_curl	video (10).mp4	Non Fall	Fall (FP)
112	Reverse_curl	video (11).mp4	Non Fall	Fall (FP)
113	Reverse_curl	video (12).mp4	Non Fall	Fall (FP)
114	Reverse_curl	video (13).mp4	Non Fall	Fall (FP)
115	Reverse_curl	video (14).mp4	Non Fall	Fall (FP)
116	Reverse_curl	video (15).mp4	Non Fall	Fall (FP)
117	Side_leg_raise	video (1).mp4	Non Fall	Fall (FP)
118	Side_leg_raise	video (2).mp4	Non Fall	Fall (FP)
119	Side_leg_raise	video (3).mp4	Non Fall	Fall (FP)
120	Side_leg_raise	video (4).mp4	Non Fall	Fall (FP)
121	Side_leg_raise	video (5).mp4	Non Fall	Fall (FP)
122	Side_leg_raise	video (6).mp4	Non Fall	Fall (FP)

Lanjut pada halaman berikutnya

Tabel 3.5 Hasil Deteksi dari Replikasi Algoritma untuk Olahraga Matras (lanjutan)

No	Environment	Nama Video	Kategori	Hasil Deteksi
123	Side_leg_raise	video (7).mp4	Non Fall	Not Fall (TN)
124	Side_leg_raise	video (8).mp4	Non Fall	Fall (FP)
125	Side_leg_raise	video (9).mp4	Non Fall	Fall (FP)
126	Side_leg_raise	video (10).mp4	Non Fall	Fall (FP)
127	Side_leg_raise	video (11).mp4	Non Fall	Fall (FP)
128	Side_leg_raise	video (12).mp4	Non Fall	Not Fall (TN)
129	Side_leg_raise	video (13).mp4	Non Fall	Fall (FP)
130	Side_leg_raise	video (14).mp4	Non Fall	Fall (FP)

Tautan ke Data Video

Tabel 3.6 merangkum metrik performa keseluruhan dari algoritma replikasi pada dataset olahraga matras.

Tabel 3.6. Hasil Evaluasi Keseluruhan dari Replikasi Algoritma untuk Olahraga Matras

Hasil Keseluruhan	
Metrik	**Nilai**
Total Video	130
True Positive (TP)	0
True Negative (TN)	9
False Positive (FP)	121
False Negative (FN)	0
Accuracy	6.92%
Precision	0.00%
Recall	0.00%
F1 Score	0.00%

Berdasarkan hasil dari replikasi algoritma yang dapat dilihat pada Tabel 3.6 menggunakan *dataset* olahraga matras, secara jelas menunjukkan bahwa algoritma deteksi jatuh replikasi memiliki tingkat *false positive* yang sangat tinggi pada *dataset* aktivitas olahraga. Hampir seluruh *video* olahraga salah diklasifikasikan sebagai kejadian jatuh (*false positives*), sementara hanya sebagian kecil (9 *video*) yang berhasil terdeteksi dengan benar sebagai *non-fall* (*true negatives*). *accuracy* keseluruhan yang sangat rendah (6.98%) dan nilai *precision*, *recall*, serta *F1 Score* yang 0.00% (karena tidak ada *true Positives* sama sekali dalam pengujian ini)

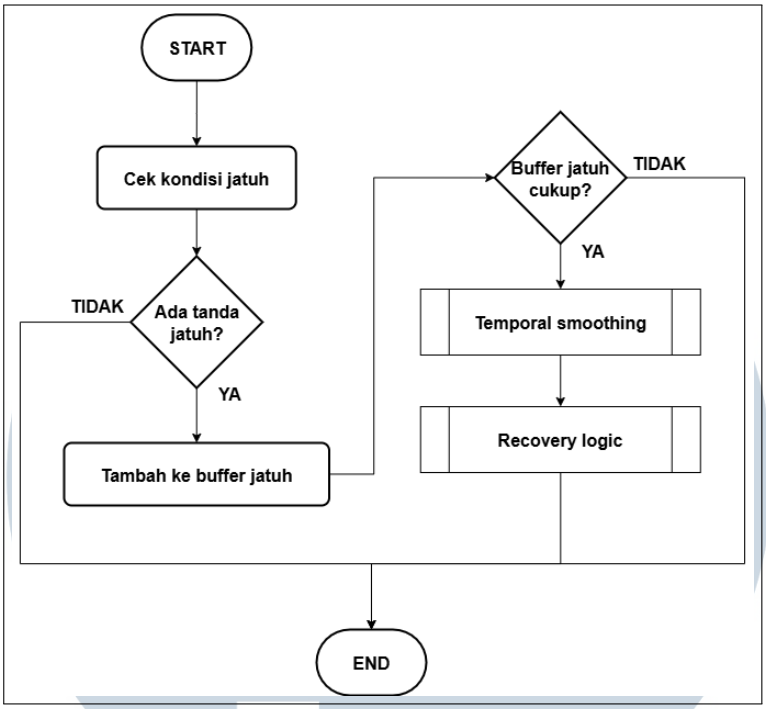
menegaskan bahwa algoritma ini belum mampu membedakan gerakan olahraga dari jatuh. Hal ini mengonfirmasi secara limitasi yang disebutkan dalam literatur sebelumnya dan menjadi dasar untuk upaya pengembangan algoritma pada bagian selanjutnya.

3.4 Optimasi Algoritma Deteksi Jatuh

Berdasarkan temuan bahwa aktivitas olahraga matras memicu tingkat *false alarm* yang tinggi pada algoritma replikasi, tahap ini berfokus pada optimasi untuk mengatasi masalah tersebut. Tujuan utamanya adalah meningkatkan kinerja sistem secara keseluruhan. Proses optimasi ini diterapkan setelah alur deteksi jatuh awal selesai diproses, sesuai dengan *flowchart* yang ditunjukkan pada Gambar 3.4.

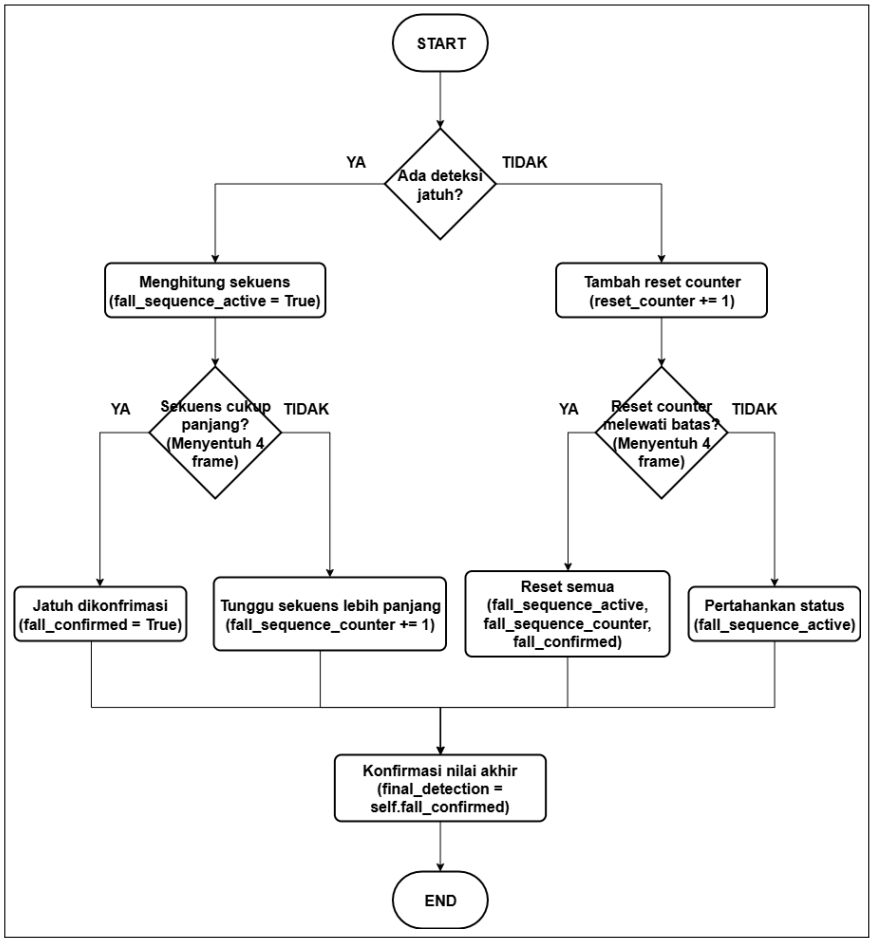
3.4.1 Perancangan Optimasi Algoritma

Pada Gambar 3.16 menjelaskan alur kerja keseluruhan dari algoritma yang telah dioptimalkan, dirancang untuk menambah lapisan validasi sebelum mencapai keputusan akhir. Proses dimulai dengan pengecekan kondisi jatuh dasar pada setiap *frame* video. Jika terdeteksi adanya tanda-tanda jatuh, sistem tidak langsung mengeluarkan alarm, melainkan menambahkan informasi tersebut ke dalam sebuah *buffer* untuk mengumpulkan bukti dari beberapa *frame* secara berurutan. Setelah jumlah deteksi di dalam *buffer* dianggap cukup, barulah kedua logika optimasi utama dijalankan. *Temporal Smoothing* untuk memastikan konsistensi kejadian, diikuti oleh *Recovery Logic* untuk memeriksa adanya pola pemulihan gerak. Proses langkah ini memastikan bahwa keputusan akhir didasarkan pada serangkaian pengamatan dari waktu ke waktu, sehingga secara efektif dapat mengurangi *false positive*.



Gambar 3.16. *Flowchart* keseluruhan untuk proses dari Optimasi Algoritma

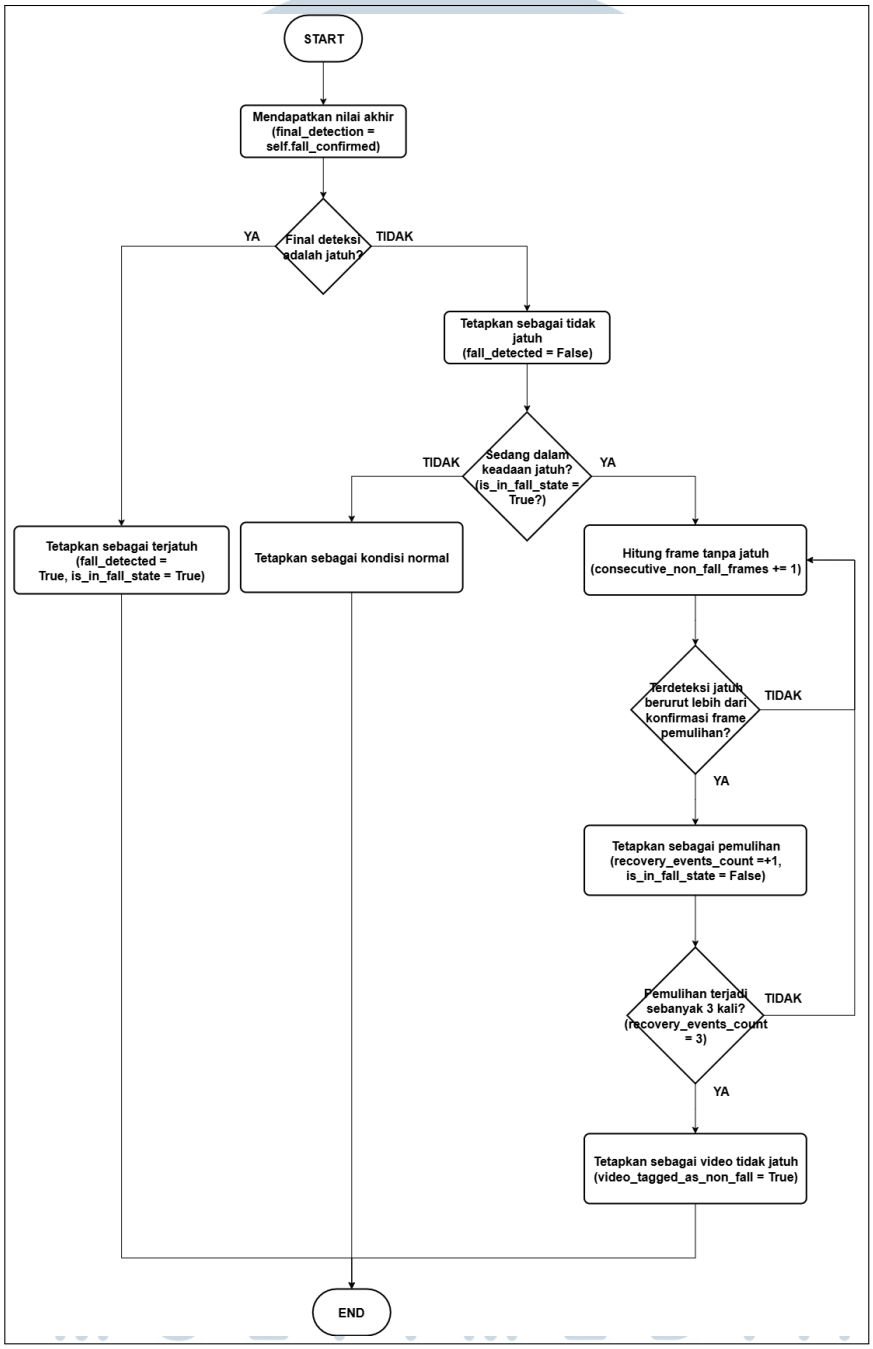
Pada Gambar 3.17 menjelaskan alur logika dari *Temporal Smoothing*. Pertama alur mulai dari melakukan validasi insiden jatuh berdasarkan konsistensi deteksi dari waktu ke waktu, yang alur kerjanya terbagi menjadi dua cabang utama. Jika terdeteksi adanya tanda jatuh, sistem akan mulai menghitung sekuens jatuh (`fall_sequence_active = True`). Apabila panjang sekuens ini telah mencapai ambang batas yang ditentukan (misalnya 4 *frame*), maka insiden jatuh akan ditetapkan (`fall_confirmed = True`) jika belum, sistem akan terus menambah hitungan sekuens. Sebaliknya jika tidak ada tanda jatuh, sistem akan menambah hitungan reset (`reset_counter`). Apabila hitungan reset ini melewati batasnya, maka semua status sekuens jatuh akan dilakukan ulang, yang menandakan bahwa urutan jatuh sebelumnya telah terputus dan dianggap sebagai anomali. Nilai akhir dari proses ini (`final_detection`) ditentukan oleh status `self.fall_confirmed`, sehingga memastikan sebuah insiden jatuh hanya dilakukan validasi jika terdeteksi secara konsisten dalam beberapa *frame*.



Gambar 3.17. *Flowchart* dari proses *Temporal Smoothing*

Pada Gambar 3.18 menjelaskan alur logika dari *Recovery Logic* yang berfungsi sebagai lapisan validasi akhir setelah mendapatkan nilai dari *Temporal Smoothing* (`final_detection`). Alur kerja dirancang untuk membedakan antara insiden jatuh yang sebenarnya dengan aktivitas di mana subjek bangkit kembali. Jika hasil `final_detection` adalah ya (terdeteksi jatuh), sistem akan menetapkan status saat ini sebagai terjatuh dengan mengaktifkan variabel `fall_detected` dan `is_in_fall_state`. Sebaliknya, jika hasilnya tidak, sistem akan memeriksa apakah sebelumnya ia berada dalam keadaan jatuh (`is_in_fall_state`). Jika ya, sistem akan mulai menghitung *frame* non-jatuh secara berurutan. Apabila jumlah *frame* non-jatuh ini telah mencapai ambang batas konfirmasi pemulihan, maka sebuah kejadian pemulihan akan ditetapkan, dan status `is_in_fall_state` akan dinonaktifkan. Puncaknya, jika jumlah kejadian pemulihan ini terjadi sebanyak batas yang ditentukan (misalnya 3 kali), maka seluruh video akan ditetapkan sebagai video tidak jatuh (`video_tagged_as_non_fall = True`), yang secara

efektif membatalkan semua alarm pada video tersebut dan mengatasi *false positive* dari gerakan berulang seperti olahraga.



Gambar 3.18. *Flowchart* dari proses *Recovery Logic*

3.4.2 Implementasi Algoritma Replikasi

A Perubahan pada Inisialisasi Parameter dan Variabel Status

Perbedaan pertama dan paling fundamental terletak pada metode inisialisasi kelas `FallDetector`. Pada versi yang dikembangkan, parameter dan variabel baru ditambahkan untuk mendukung *temporal smoothing* dan logika pemulihan *(recovery logic)*. Berikut dapat dilihat potongan kode pada Gambar 3.19.

```
def __init__(self, poseweights='yolov7-w6-pose.pt', device='0'):
    # ... (inisialisasi model) ...

    # ADJUSTED PARAMETERS
    self.LENGTH_FACTOR_ALPHA = 0.6
    self.VELOCITY_THRESHOLD = 0.8
    self.LEG_ANGLE_THRESHOLD = 50
    self.TORSO_ANGLE_THRESHOLD = 45
    self.ASPECT_RATIO_THRESHOLD = 0.9
    self.CONFIDENCE_THRESHOLD = 0.4
    # DIHAPUS: self.MINIMUM_FALL_FRAMES = 15

    # ... (variabel state tracking dasar) ...

    # RECOVERY tracking variables (BARU)
    self.is_in_fall_state = False
    self.recovery_events_count = 0
    self.consecutive_non_fall_frames = 0
    self.RECOVERY_CONFIRMATION_FRAMES = 5
    self.MAX_RECOVERIES_ALLOWED = 3
    self.video_tagged_as_non_fall = False
    self.fall_to_recovery_transitions = []

    # TEMPORAL FALL SEQUENCE LOGIC (BARU)
    self.reset_counter = 0
    self.fall_sequence_active = False
    self.fall_sequence_counter = 0
    self.FALL_SEQUENCE_REQUIRED = 4
    self.MAX_RESET_INTERVAL = 4
    self.fall_confirmed = False
    self.expected_fall_type = None
```

Gambar 3.19. Tampilan *script* inisialisasi parameter dan variabel status

Berdasarkan Gambar 3.19 dapat dilihat metode `__init__` terjadi perubahan. Parameter `MINIMUM_FALL_FRAMES` dihapus dan digantikan oleh serangkaian parameter dan variabel status baru untuk mengelola *temporal smoothing* dan pemulihan *(recovery)*. Variabel ini memungkinkan objek `FallDetector` untuk dapat mengingat status dari *frame* sebelum membuat keputusan. Penambahan variabel seperti `is_in_fall_state`, `recovery_events_count`, `fall_sequence_counter`, dan `fall_confirmed` mengubah `FallDetector` menjadi sebuah *state machine*. Setelah itu, sistem dapat melacak status ”sedang jatuh”, menghitung ”peristiwa pemulihan”, dan mengonfirmasi jatuh hanya jika terjadi dalam urutan temporal yang valid.

Penentuan nilai untuk parameter baru yang terkait *temporal smoothing* dan *recovery logic* tidak dilakukan secara acak, melainkan berdasarkan serangkaian pengujian berulang pada kedua *dataset*. Nilai `FALL_SEQUENCE_REQUIRED = 4` dipilih karena durasi tersebut cukup untuk memvalidasi konsistensi gerakan jatuh tanpa terlalu lambat merespons. Demikian pula, nilai `RECOVERY_CONFIRMATION_FRAMES = 5` dan `MAX_RECOVERIES_ALLOWED = 3` ditemukan memberikan keseimbangan terbaik antara pengurangan *false positive* pada *dataset* olahraga matras dan tetap mempertahankan sensitivitas tinggi pada *dataset* Le2i. Proses ini bersifat iteratif, di mana beberapa kombinasi nilai diuji untuk menemukan konfigurasi yang paling optimal.

B Penambahan Temporal Smoothing dan Recovery Logic

Pada kode optimasi, setelah keputusan awal (`basic_detection`) dibuat, ia tidak langsung menjadi *output* akhir. Hasil ini dimasukkan ke dalam dua lapisan logika baru yaitu *temporal smoothing* dan *recovery logic* yang bekerja bersamaan untuk menghasilkan keputusan *frame*. Berikut dapat dilihat potongan kode pada Gambar 3.20.

```python
def detect_fall(self, keypoints):

            # ... (setelah perhitungan semua kondisi) ...

            is_fall = conditions_met >= 2
            self.fall_buffer.append(is_fall)
            basic_detection = sum(self.fall_buffer) >= 2 if len(self.fall_buffer) >= 1 else is_fall

            # === TEMPORAL FALL SEQUENCE LOGIC ===
            if basic_detection:
                self.reset_counter = 0  # Reset karena deteksi jatuh
                if not self.fall_sequence_active:
                    self.fall_sequence_active = True
                    self.fall_sequence_counter = 1
                else:
                    self.fall_sequence_counter += 1

                if self.fall_sequence_counter >= self.FALL_SEQUENCE_REQUIRED:
                    self.fall_confirmed = True
            else:
                self.reset_counter += 1
                if self.reset_counter >= self.MAX_RESET_INTERVAL:
                    self.fall_sequence_active = False
                    self.fall_sequence_counter = 0
                    self.fall_confirmed = False

            final_detection = self.fall_confirmed

            # RECOVERY LOGIC
            if final_detection:
                current_state = "fallen"
                self.fall_detected = True
                self.is_in_fall_state = True
                self.consecutive_non_fall_frames = 0
            else:
                self.fall_detected = False
                if self.is_in_fall_state:
                    self.consecutive_non_fall_frames += 1

                    if self.consecutive_non_fall_frames >= self.RECOVERY_CONFIRMATION_FRAMES:
                        self.recovery_events_count += 1
                        current_state = "recovering"
                        self.is_in_fall_state = False
                        # ... (Reset semua state jatuh dan temporal) ...

                        if self.recovery_events_count >= self.MAX_RECOVERIES_ALLOWED:
                            self.video_tagged_as_non_fall = True

            # ... (update variabel tracking) ...

            return final_detection, current_state, conditions_info
```

Gambar 3.20. Tampilan *script* untuk proses dari *Temporal Smoothing* dan *Recovery Logic*

Berdasarkan Gambar 3.20 dapat dilihat alur pemrosesan yang memperkenalkan beberapa lapisan validasi sebelum mencapai keputusan akhir. Hasil deteksi dasar *per frame*, yang disimpan dalam variabel `basic_detection`, tidak lagi langsung digunakan sebagai *output* akhir. Sebaliknya, hasil ini berfungsi sebagai input untuk memperbarui penghitung urutan temporal, yaitu `fall_sequence_counter`. Dengan demikian, keputusan deteksi final untuk sebuah *frame* (`final_detection`) kini sepenuhnya bergantung pada status `self.fall_confirmed`, yang hanya akan bernilai *True* jika urutan *frame* jatuh yang konsisten dan tidak terputus telah terpenuhi. Selanjutnya, nilai `final_detection` ini menjadi pemicu untuk logika pemulihan yang diimplementasikan dalam

kondisional `final_detection` Logika ini mengelola dan mengubah status sistem, seperti `self.is_in_fall_state`, dan bahkan mengambil keputusan pada level video dengan mengatur `self.video_tagged_as_non_fall` menjadi *true*, yang membatalkan semua deteksi jatuh lainnya pada video tersebut.