BAB 2
LANDASAN TEORI

2.1 Penelitian Terkait

Berdasarkan penelitian yang dilakukan oleh Tîrziu et al. [13], salah satu permasalahan utama yang teridentifikasi pada Algoritma Deteksi Jatuh adalah ketiadaan pengujian terhadap aktivitas olahraga matras, yang berpotensi memicu *false alarm*. Tabel 2.1 merupakan upaya penelitian terkait dalam mengatasi permasalahan. Pendekatan yang dilakukan mencakup *Computer Vision, Object Detection,* dan *Pose Estimation*.

Tabel 2.1. Ringkasan Penelitian Terdahulu yang relevan dengan Sistem Deteksi Jatuh menggunakan

Referensi	**Metodologi**	**Keunggulan**	**Limitations**
Tîrziu et al. [13]	YOLOv7-W6-Pose + Telegram API	1. Akurasi tinggi, *real-time*. 2. Menjaga privasi data.	1. *False positive* aktivitas olahraga. 2. Sensitif pencahayaan, *occlusion*.
Kuzevanov et al. [15]	*Skeletal Structure Recognition* (MediaPipe) + *Deep Learning* (LSTM) untuk Klasifikasi Aksi	1. Klasifikasi aktivitas olahraga. 2. *Robust motion blur*, skala kecil.	1. *Pose estimation* sensitif lingkungan. 2. Tidak deteksi *multi-person*.
Arif et al. [16]	*Human Pose Estimation* (GMM) + *Object Detection* (YOLO, K-means) + *Human Object Interaction* (HOI)	1. Memahami aktivitas olahraga kompleks. 2. Akurasi HOI hingga 53.6%.	1. Deteksi postur/objek kecil rumit. 2. HOI berbasis *appearance features* saja.

Penelitian-penelitian terdahulu menunjukkan beragam pendekatan dalam bidang deteksi jatuh dan klasifikasi aktivitas, dengan berbagai keunggulan dan

limitasi. Salah satu penelitian yang menjadi dasar replikasi pada penelitian ini berhasil mengembangkan sistem deteksi jatuh berbasis YOLOv7-W6-Pose yang akurat dan beroperasi secara *real-time* sambil menjaga privasi pengguna. Namun, sistem tersebut secara eksplisit mengakui adanya limitasi, terutama terkait *false positive* pada aktivitas yang menyerupai jatuh, seperti olahraga di matras, serta sensitivitas terhadap variasi pencahayaan dan *occlusion*. Hal ini menjadi titik tolak utama bagi penelitian ini untuk melakukan pengembangan lebih lanjut [13].

Di sisi lain, beberapa penelitian telah berfokus pada klasifikasi aktivitas olahraga menggunakan *skeletal structure* yang diekstraksi oleh MediaPipe dan model *deep learning* LSTM. Keunggulan pendekatan ini terletak pada kemampuannya menganalisis gerakan olahraga secara detail dengan akurasi sudut yang baik dari data 3D, serta ketahanannya terhadap *motion blur* dan skala *objek* kecil dibandingkan dengan beberapa metode lain. Meskipun demikian, penelitian tersebut juga mencatat bahwa estimasi *pose* tetap sensitif terhadap kondisi lingkungan seperti pencahayaan dan *occlusions*, dan bahwa YOLOV7 memiliki performa yang lebih baik dalam menangani *occlusion* serta gerakan cepat pada gambar resolusi tinggi [15].

Pendekatan lain juga telah mengeksplorasi *Human Object Interaction* (HOI) dalam konteks olahraga, memanfaatkan kombinasi *pose estimation* dan *object detection* dengan model seperti YOLO. Penelitian ini menyoroti kompleksitas deteksi postur langka dan *objek* yang terhalang, serta keterbatasan pendekatan berbasis fitur *appearance* saja dalam memprediksi interaksi [16].

2.2 Algoritma Deteksi Jatuh

Algoritma deteksi jatuh pada penelitian Tîrziu et al. [13] menggunakan YOLOv7-W6-Pose untuk melakukan ekstraksi pada tubuh meliputi kepala, hidung, bahu, siku, pinggul, lutut, dan pergelangan kaki. YOLOv7-W6-Pose yang dimanfaatkan dalam algoritma ini adalah model yang telah dilakukan *pretrained* menggunakan *dataset Common Objects in Context* COCO dan dapat mendeteksi 17 titik tubuh manusia dalam satu proses *frame*. Algoritma yang dibangun menganalisis posisi berbagai tubuh dalam menentukan apakah seseorang telah jatuh. Algoritma ini mengevaluasi hubungan antara posisi bahu, torso, dan kaki untuk mengidentifikasi indikasi postur jatuh. Proses ini memerlukan penetapan dan perbandingan koordinat tertentu, serta penentuan *threshold* yang merupakan definisi kejadian jatuh [13]. Berikut merupakan *pseudocode* dari algoritma deteksi

jatuh pada penelitian yang dilakukan oleh Tîrziu et al. [13].

Algorithm 1: Algoritma Deteksi Jatuh

Input: Stream video dari kamera

Output: Sinyal deteksi jatuh

Inisialisasi model YOLOv7-W6-Pose dengan bobot pre-trained

Atur threshold: $speed\_threshold$ dan $angle\_threshold$ (misal, 45°)

while *video stream aktif* **do**

- Tangkap dan pra-proses $current\_frame$ (resize ke 960x960, normalisasi)
- Deteksi pose dan ekstrak koordinat keypoints:
  - Bahu: $S_l(x_l, y_l), S_r(x_r, y_r)$
  - Badan: $T_l(x_{Tl}, y_{Tl}), T_r(x_{Tr}, y_{Tr})$
  - Kaki: $F_l(x_{Fl}, y_{Fl}), F_r(x_{Fr}, y_{Fr})$
- **if** *seseorang terdeteksi* **then**
  - Hitung $L_{factor} = \sqrt{(x_l - x_{Tl})^2 + (y_l - y_{Tl})^2}$
  - Hitung tinggi tubuh: $H_{body} = |y_l - y_{Fl}|$
  - Hitung lebar tubuh: $W_{body} = |x_l - x_r|$
  - **if** $(y_l \leq y_{Fl} + \alpha \cdot L_{factor})$ $(H_{body} < W_{body})$ **then**
    - Hitung kecepatan vertikal keypoints antar frame
    - **if** $kecepatan\ vertikal > speed\_threshold$ **then**
      - Hitung sudut antara badan dan kaki
      - **if** $sudut < angle\_threshold$ **then**
        - Atur $fall\_detected$ = TRUE
      - **else**
        - Atur $fall\_detected$ = FALSE
    - **else**
      - Atur $fall\_detected$ = FALSE
  - **else**
    - Atur $fall\_detected$ = FALSE

Berdasarkan *pseudocode* yang telah dijabarkan dapat dilihat proses algoritma diawali dengan mengekstraksi koordinat dari titik-titik kunci pada tubuh, yang meliputi:

1. Bahu kiri dan kanan: $S_l(x_l, y_l), S_r(x_r, y_r)$

2. Torso (sisi kiri dan kanan): $T_l(x_{Tl}, y_{Tl}), T_r(x_{Tr}, y_{Tr})$

3. Kaki (kiri dan kanan): $F_l(x_{Fl}, y_{Fl}), F_r(x_{Fr}, y_{Fr})$

Setelah koordinat tersebut diperoleh, nilai antara (*intermediate value*) yang dinamakan *length factor* dihitung. Faktor ini didasarkan pada jarak geometris, tepatnya jarak *Euclidean*, antara posisi bahu kiri dan torso kiri. Selanjutnya, *length factor* ini digunakan untuk mengevaluasi hubungan dan posisi relatif antar bagian tubuh [13].

$$L_{factor} = \sqrt{(x_l - x_{Tl})^2 + (y_l - y_{Tl})^2} \tag{2.1}$$

Sumber: Tîrziu et al. [13]

Faktor ini menyesuaikan ambang deteksi sehingga algoritma dapat diterapkan terlepas dari tinggi atau proporsi tubuh seseorang.

2.2.1 Threshold

1. Tinggi Bahu Relatif terhadap Kaki

   Pemeriksaan ini dilakukan untuk melihat apakah koordinat vertikal bahu kiri (atau kanan) berada di bawah koordinat vertikal kaki kiri (atau kanan), yang disesuaikan oleh *length factor*. Dalam postur normal, bahu seharusnya berada di atas kaki, namun saat jatuh, posisinya bisa berakhir lebih rendah atau sejajar dengan kaki. Algoritma mempertimbangkan kemungkinan jatuh jika bahu jatuh ke level yang mendekati kaki [13]. *Threshold* ini dapat didefinisikan sebagai berikut:

   $$y_l \leq y_{Fl} + a \cdot L_{factor}, \tag{2.2}$$

   Sumber: Tîrziu et al. [13]

   di mana a adalah faktor penyesuaian kecil untuk memperhitungkan variasi alami tubuh dalam posisi normal.

2. Perbedaan antara Lebar dan Tinggi Tubuh

   Algoritma menghitung perbedaan antara dimensi vertikal (tinggi) dan horizontal (lebar) dari kotak pembatas yang mengelilingi tubuh. Jika tinggi

menjadi lebih kecil dari lebar (yaitu, tubuh ”terlentang” secara horizontal daripada vertikal), ini mengindikasikan bahwa orang tersebut telah jatuh [13]. Perhitungan dimensi tubuh dapat dilakukan sebagai berikut:

(a) Tinggi tubuh:

$$H_{body} = |y_l - y_{Fl}|, \tag{2.3}$$

Sumber: Tîrziu et al. [13]

(b) Lebar tubuh (jarak antara bahu atau antara sisi dada):

$$W_{body} = |x_l - x_r|, \tag{2.4}$$

Sumber: Tîrziu et al. [13]

(c) *Threshold* untuk mendeteksi jatuh dapat ditetapkan sebagai:

$$H_{body} < W_{body} \tag{2.5}$$

Sumber: Tîrziu et al. [13]

2.2.2 Fall Decision

Jika semua kondisi terpenuhi, bahu dan torso berada pada ketinggian yang sangat rendah dibandingkan kaki dan tubuh memiliki dimensi horizontal yang lebih besar daripada dimensi vertikal, algoritma menyimpulkan bahwa telah terjadi jatuh. Dalam skenario ini, sinyal jatuh positif akan ditampilkan, diikuti dengan koordinat *bounding box* yang mengelilingi tubuh [13]. *Threshold* kunci dalam algoritma ini meliputi:

1. Tinggi bahu dan torso relatif terhadap kaki. *Threshold* ini disesuaikan menggunakan *length factor* untuk memperhitungkan proporsi tubuh individu. Pada dasarnya, ini memeriksa apakah bahu dan torso turun di bawah tingkat yang telah disesuaikan relatif terhadap kaki.
2. Perbedaan antara dimensi tubuh. Jika perbedaan antara tinggi dan lebar tubuh menjadi negatif (tinggi lebih kecil dari lebar), ini menandakan bahwa orang tersebut sedang dalam posisi berbaring.

2.2.3 Diferensiasi Antara Jatuh dan Posisi Berbaring

Dalam membedakan antara insiden jatuh dan gerakan berbaring yang disengaja, sistem ini menganalisis kecepatan vertikal dari pergerakan tubuh. Karena jatuh ditandai oleh perubahan posisi yang cepat, kecepatan vertikal dari *keypoints* seperti bahu, torso, atau kaki diukur antar *frame* yang berurutan. Jika kecepatan ini melampaui sebuah *threshold* yang telah ditentukan, sistem akan mengonfirmasi terjadinya insiden jatuh. Analisis kecepatan ini dilakukan untuk mendeteksi penurunan mendadak yang mengindikasikan jatuh, dan bukan gerakan lambat dan terkontrol seperti berbaring.[13].

Analisis sudut tubuh juga diimplementasikan untuk membedakan jatuh dari berbaring. Insiden jatuh yang cepat akan menghasilkan sudut tajam saat torso dan kaki menjadi paralel dengan tanah, berbeda dengan transisi postur yang lebih lambat pada saat berbaring. Sebuah ambang batas 45 derajat pada sudut antara torso dan kaki diterapkan sebagai kriteria deteksi. Penurunan sudut di bawah ambang batas ini mengindikasikan posisi tubuh yang horizontal secara tiba-tiba, yang diklasifikasikan sebagai kejadian jatuh. [13].

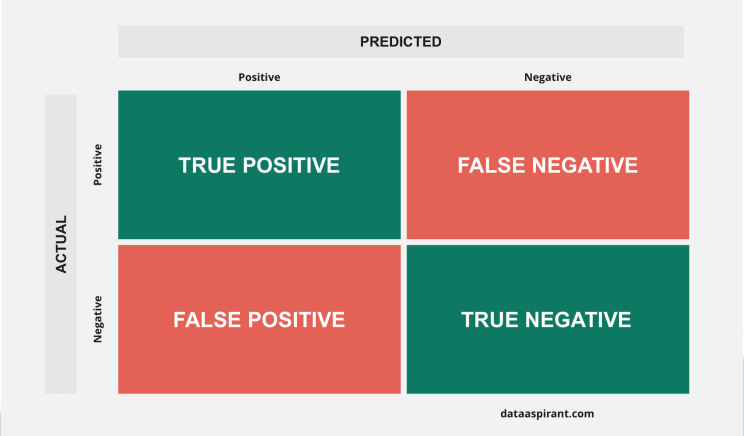
2.3 Metrik Evaluasi Sistem Deteksi Jatuh

Evaluasi kinerja sistem deteksi jatuh YOLOv7-W6-Pose dilakukan dengan menggunakan serangkaian metrik yang meliputi *Confusion Matrix, Accuracy, Recall, Precision,* dan *F1-Score*.

1. *Confusion Matrix*

   Confusion Matrix adalah representasi tabel yang memvisualisasikan kinerja hasil algoritma klasifikasi model. Matriks ini memiliki dua dimensi yang menampilkan hasil aktual dan hasil prediksi dari algoritma klasifikasi model.

   Gambar 2.1 mengilustrasikan nilai *True Positive* (TP), yang menandakan prediksi yang benar bahwa suatu objek bernilai positif, serta nilai *False Negative* (FN), yang menunjukkan prediksi yang salah bahwa suatu objek bernilai negatif. Di sisi lain, nilai *True Negative* (TN) menunjukkan prediksi yang benar bahwa suatu objek bernilai negatif [17].



Gambar 2.1. *Confusion Matrix*

Sumber: Data Aspirant [18]

2. *Accuracy*

   Accuracy digunakan untuk mengukur sejauh mana sistem dapat mengklasifikasikan *frame* secara benar, baik mendeteksi jatuh secara tepat maupun mengabaikan kejadian yang bukan jatuh [19]. Dapat dihitung dengan Rumus 2.6.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

3. *Recall*

   Recall atau sensitivitas mengukur kemampuan sistem dalam mengenali seluruh kejadian jatuh yang sebenarnya. Dapat dihitung dengan Rumus 2.7.

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

4. *Precision*

   Precision mengukur seberapa akurat prediksi jatuh yang dibuat oleh sistem. Artinya, *precision* memperlihatkan proporsi kejadian jatuh yang benar-benar terjadi dari seluruh prediksi jatuh yang dikeluarkan [19]. Dapat dihitung dengan Rumus 2.8.

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

5. *F1-Score*

   F1-Score digunakan untuk mengevaluasi keseimbangan antara *recall* dan *precision* [19]. Dapat dihitung dengan Rumus 2.9.

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.9}$$