BAB 2
LANDASAN TEORI

2.1 Penelitian Terkait

Salah satu batasan masalah dari penelitian yang dilakukan oleh Tîrziu et al. [11] adalah algoritma deteksi jatuh yang belum dioptimalkan pada kondisi terhalang oleh objek lain. Algoritma yang belum optimal pada kondisi tersebut dapat memberikan peringatan palsu dan berpotensi untuk merugikan berbagai pihak, baik individu yang diawasi maupun tenaga medis yang harus siap siaga. Tabel 2.1 merupakan beberapa referensi penelitian terkait yang turut berperan dalam pengembangan deteksi jatuh.

Tabel 2.1. Penelitian Terkait Deteksi Jatuh dalam Kondisi Terhalang Objek

Referensi	**Metodologi**	**Keunggulan**	**Keterbatasan**
Tîrziu et al. [11]	YOLOv7-W6-Pose + *Telegram API*	• Akurasi 95,7%. • Privasi terjaga. • Algoritma *Custom*.	• Sensitif *occlusion*. • *False positive* ketika tubuh terhalang objek.
Qin et al. [12]	ESD-YOLO (YOLOv8 + Dynamic Conv + DyHead + EASlideloss)	• Peningkatan performa: +1,9% *precision*, +4,1% *recall*, +4,3% mAP@0.5.	• Model yang kompleks sehingga butuh GPU mumpuni untuk inferensi *real-time*. • Belum diuji di berbagai kondisi dunia nyata.
Zheng et al. [13]	YOLOv8 + FasterNet	• Beban komputasi model lebih ringan dari model YOLOv8 standar. • Akurasi mAP@0.5 meningkat menjadi 96,2% dari 94,9%.	• Belum menggunakan *keypoints* atau *skeleton*. • Belum diuji pada kondisi *occlusion* berat.

Tîrziu et al. [11] mengembangkan sistem deteksi jatuh yang *real-time* berbasis YOLOv7-W6-Pose dengan menggabungkan deteksi objek dan estimasi *pose* (kerangka tubuh) untuk mendeteksi kejadian jatuh khususnya lansia. Sistem ini memproses titik-titik atau *keypoints* dari bagian tubuh yang akan berwujud kerangka tanpa menyimpan data personal, sehingga menjaga privasi pengguna. Pengujian dilakukan pada video simulasi dan video nyata dan memberikan hasil yang menunjukkan performa sistem dalam mendeteksi jatuh secara efisien dengan tingkat akurasi sebesar 95,7%. Namun, sistem deteksi belum dirancang secara eksplisit untuk menangani kondisi *occlusion* yang berat. Pendekatan sistem deteksi jatuh berbasis estimasi *pose* ini membuka potensi untuk integrasi metode penanganan *occlusion* berbasis *pose* pada masa depan.

Qin et al. [12] memperkenalkan model ESD-YOLO yang merupakan pengembangan dari YOLOv8 dengan menambahkan modul *Deformable Convolution* (DCNv3), *Dynamic Attention* (DyHead), dan *loss function* (EASlideloss). Model ini dirancang khusus untuk menangani skenario kompleks seperti *occlusion*, deformasi tubuh, dan variasi ukuran target. Hasil eksperimen menunjukkan peningkatan metrik performa seperti *precision* yang naik sebesar 1,9%, *recall* sebesar 4,1%, dan mAP@0.5 sebesar 4,3% dibandingkan YOLOv8 standar. Kelebihan utama model ini adalah kemampuannya dalam mendeteksi postur jatuh yang tidak ideal dalam lingkungan ramai atau tidak terstruktur dengan kelemahannya yang memerlukan daya komputasi tinggi, sehingga dapat dikatakan bahwa biaya nya besar.

Sementara itu, Zheng et al. [13] mengembangkan model deteksi jatuh FDT-YOLO yang merupakan pengembangan dari model YOLOv8. Pengembangan dilakukan dengan mengganti modul C2f menjadi FasterNet, lalu menambahkan *Deformable Convulution* pada bagian *neck*, dan mengintegrasikan mekanisme *Triplet Attention*. Model ini dirancang untuk menangani berbagai tantangan seperti variasi pencahayaan, deformasi postur tubuh, serta gangguan dari latar belakang atau occlusion ringan. Hasil eksperimen menunjukkan peningkatan akurasi dari mAP@0.5 sebesar 94,9% menjadi 96,2%, serta mAP@0.5–0.95 dari 84,2% menjadi 85,9%, dengan jumlah parameter yang relatif kecil yaitu hanya 9,9 juta, sehingga cocok untuk *deployment* di perangkat terbatas. Namun, meskipun menunjukkan peningkatan performa dalam kondisi kompleks termasuk *occlusion* ringan, model ini belum secara eksplisit diuji pada skenario *occlusion* berat, dan masih bergantung pada citra visual yang jelas.

2.2 Algoritma Deteksi Jatuh

Algoritma yang digunakan pada penelitian Tîrziu et al. [11] dibangun untuk mengekstrak titik-titik yang ditetapkan menjadi kunci dari tubuh seseorang untuk membantu mengindikasikan terjadinya jatuh. Implementasi algoritma pada penelitian Tîrziu et al. [11] dibantu dengan penggunaan YOLOv7-W6-Pose sebagai pendukung sistem deteksi jatuh. YOLOv7-W6-Pose merupakan pengembangan dari YOLO dengan tambahan kemampuan untuk melakukan estimasi pose pada tubuh manusia yang mampu mendeteksi titik kunci (*keypoints)* pada bagian tubuh seperti kepala, bahu, siku, pinggul, lutut, dan pergelangan kaki [11]. Pada penelitian ini, *keypoints* yang ditetapkan sebagai titik kunci terletak pada bahu, *torso*, dan kaki. *Keypoints* didefinisikan ke dalam rumus sebagai berikut:

- Bahu Kiri dan Bahu Kanan: $S_l(x_l, y_l), S_r(x_r, y_r)$
- *Torso* Kiri dan *Torso* Kanan: $T_l(x_{Tl}, y_{Tl}), T_r(x_{Tr}, y_{Tr})$
- Kaki Kiri dan Kaki Kanan: $F_l(x_{Fl}, y_{Fl}), F_r(x_{Fr}, y_{Fr})$

Tahap selanjutnya setelah mendapatkan koordinat dari titik-titik yang telah di-ekstrak adalah menghitung *length factor* yang dihitung berdasarkan pada jarak antara posisi bahu kiri dan badan kiri menggunakan rumus *Eucledian*. *Length Factor* ini digunakan untuk menentukan hubungan antar bagian tubuh dan mengevaluasi posisi relatifnya. Rumus dari *Length Factor* pada penelitian ini dapat dilihat pada Rumus 2.1.

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

Sumber: Tîrziu et al. [11]

Setelah nilai berhasil didapatkan, *length factor* akan menggunakan nilai tersebut untuk menyesuaikan *threshold* dari sistem deteksi sehingga algoritma dapat diterapkan sesuai dengan tubuh individu terlepas perbedaan tinggi dan proporsi tubuh masing-masing individu.

2.2.1 Threshold

1. Tinggi Kedua Bahu Relatif dengan Kaki

   Threshold akan memeriksa koordinat vertikal dari bahu kiri atau bahu kanan lebih rendah dari kaki kiri atau kaki kanan yang disesuaikan oleh *length factor*. Pada kondisi normal, posisi bahu seharusnya berada di atas kaki, namun ketika jatuh, posisi bahu bisa saja berada di posisi yang serupa dengan kaki atau bahkan lebih rendah. Algoritma akan mempertimbangkan kemungkinan jatuh jika posisi bahu turun ke posisi tersebut. *Threshold* didefinisikan seperti pada Rumus 2.2.

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

   Sumber: Tîrziu et al. [11]

   di mana a merupakan faktor penyesuaian kecil untuk memperhitungkan variasi alami tubuh pada saat kondisi normal.

2. Perbedaan Dimensi Tubuh (Lebar dan Tinggi):

   Algoritma akan mengukur perbedaan antara dimensi vertikal (tinggi) dan horizontal (lebar) dari *bounding box* atau kotak pembatas yang mengelilingi tubuh. Apabila tinggi tubuh terdeteksi lebih kecil dari lebar tubuh (menunjukkan posisi tubuh dalam posisi datar atau horizontal), maka hal ini akan menjadi indikasi bahwa seseorang telah terjatuh. Perhitungan dimensi tinggi tubuh, lebar tubuh, dan *threshold* indikasi jatuh dapat dilihat pada Rumus 2.3, Rumus 2.4, dan Rumus 2.5 yang didefinisikan sebagai berikut:

   - Tinggi Tubuh:

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

     Sumber: Tîrziu et al. [11]

   - Lebar Tubuh (jarak antara bahu atau antara sisi dada):

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

     Sumber: Tîrziu et al. [11]

- *Threshold* untuk mendeteksi jatuh:

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

Sumber: Tîrziu et al. [11]

Jika semua kondisi telah terpenuhi, yaitu ketika bahu dan badan berada di posisi jauh lebih rendah dibandingkan dengan dengan posisi kaki, dan dimensi tubuh secara horizontal lebih tinggi dari posisi tubuh secara vertikal, maka algoritma akan menyimpulkan bahwa jatuh telah terjadi. Ketika terjadi jatuh, maka sinyal positif jatuh akan dikembalikan bersamaan dengan posisi *bounding box* yang mengelilingi tubuh. *Threshold* yang menjadi kunci di algoritma yang digunakan adalah tinggi bahu dan badan yang relatif dengan kaki, serta perbedaan antara dimensi tubuh (dimensi horizontal dan vertikal). Algoritma dan logika yang digunakan oleh Tîrziu et al [11] dapat didefinisikan sebagai *Pseudocode* 1.

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 $fall\_detected = \text{TRUE}$;

        else

          Atur $fall\_detected = \text{FALSE}$;

      else

        Atur $fall\_detected = \text{FALSE}$;

    else

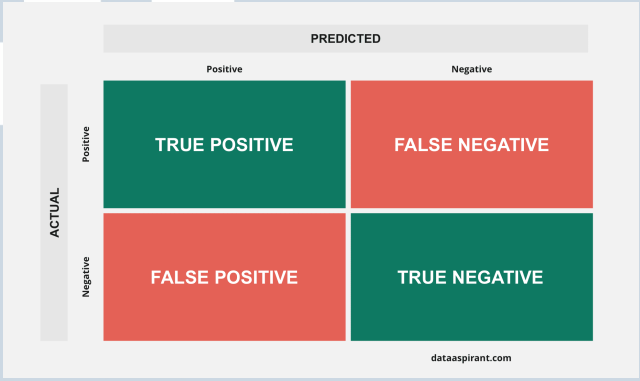
      Atur $fall\_detected = \text{FALSE}$;

2.3 Evaluation Matrix

Matriks evaluasi yang digunakan pada penelitian ini terdiri dari *Confusion Matrix*, *Precision*, *Recall*, *Accuracy*, dan *F1-Score*.

- **Confusion Matrix**

  Confusion Matrix adalah matriks evaluasi yang menunjukkan hasil dari performa sistem deteksi dalam bentuk tabel. *Confusion Matrix* memiliki dua dimensi untuk menunjukkan hasil yang asli dan hasil prediksi dari algoritma sistem. *Confusion Matrix* dapat dilihat pada Gambar 2.1.



Gambar 2.1. *Confusion Matrix*

Sumber: Data Aspirant [14]

Pada penelitian ini, *True Positive* menandakan bahwa kejadian jatuh terjadi dan sistem berhasil mendeteksi kejadian jatuh tersebut. *True Negative* menandakan bahwa kejadian jatuh tidak terjadi dan sistem tidak mendeteksi kejadian jatuh apapun. *False Positive* menandakan bahwa kejadian jatuh tidak terjadi namun sistem mendeteksi adanya kejadian jatuh. Lalu terakhir, *False Negative* menandakan bahwa kejadian jatuh terjadi namun sistem gagal mendeteksi adanya kejadian jatuh.

- **Precision**

  Precision mengukur tingkat keakuratan sistem dalam memprediksi hasil deteksi jatuh dengan nilai positif, yang artinya *Precision* mengukur berapa banyak jatuh yang benar-benar terjadi dari seluruh kemungkinan jatuh yang diprediksi. Rumus untuk menghitung nilai *precision* dapat dilihat pada Rumus 2.6.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

- **Recall**

  Recall mengukur tingkat keakuratan sistem dalam mengenali kejadian jatuh yang benar-benar terjadi. Nilai dari *Recall* yang besar menunjukkan bahwa sistem berhasil mendeteksi kejadian jatuh secara akurat. Sebaliknya, nilai *Recall* yang kecil menunjukkan bahwa sistem kurang mampu untuk mendeteksi kejadian jatuh. Rumus untuk menghitung nilai *Recall* dapat dilihat pada Rumus 2.7.

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

- **Accuracy**

  Accuracy mengukur seberapa banyak prediksi yang tepat dibandingkan dengan jumlah prediksi yang dilakukan oleh sistem, baik prediksi jatuh maupun tidak jatuh. Rumus untuk menghitung nilai *Accuracy* dapat dilihat pada Rumus 2.8.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.8}$$

- **F1-Score**

  F1-Score mengukur keseimbangan antara *Precision* dan *Recall*. Nilai *F1-Score* yang tinggi menunjukkan bahwa sistem memiliki keseimbangan antara *Precision* dan *Accuracy* yang baik. Rumus untuk menghitung nilai *F1-Score* dapat dilihat pada Rumus 2.9.

$$F1\text{-}Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.9}$$