



TUGAS AKHIR - EC234801

PENGEMBANGAN KURSI RODA OTONOM DENGAN ESP32-CAM BERBASIS YOLOv11

Aldifahmi Sihotang

NRP 0721 18 4000 0039

Dosen Pembimbing

Dr. Eko Mulyanto Yuniarno, S.T., M.T.

NIP 19680601 1995121 1 009

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.

NIP 19700313 199512 1 001

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025



TUGAS AKHIR - EC234801

**PENGEMBANGAN KURSI RODA OTONOM DENGAN
ESP32-CAM BERBASIS YOLOv11**

Aldifahmi Sihotang

NRP 0721 18 4000 0039

Dosen Pembimbing

Dr. Eko Mulyanto Yuniarno, S.T., M.T.

NIP 19680601 1995121 1 009

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.

NIP 19700313 199512 1 001

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025

[Halaman ini sengaja dikosongkan]



FINAL PROJECT - EC234801

DEVELOPMENT OF AUTONOMOUS WHEELCHAIR WITH ESP32-CAM BASED ON YOLOv11

Aldifahmi Sihotang

NRP 0721 18 4000 0039

Advisor

Dr. Eko Mulyanto Yuniarno, S.T., M.T.

NIP 19680601 1995121 1 009

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.

NIP 19700313 199512 1 001

Undergraduate Study Program of Computer Engineering

Department of Computer Engineering

Faculty of Intelligent Electrical and Informatics Technology

Sepuluh Nopember Institute of Technology

Surabaya

2025

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENGEMBANGAN KURSI RODA OTONOM DENGAN *ESP32-CAM* BERBASIS *YOLOv11*

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Teknik pada
Program Studi S-1 Teknik Komputer
Departemen Teknik Komputer
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh: **Aldifahmi Sihotang**
NRP. 0721 18 4000 0039

Disetujui oleh Tim Penguji Tugas Akhir:

Dr. Eko Mulyanto Yuniarno, S.T., M.T.
NIP: 19680601 1995121 1 009

(Pembimbing I)

.....

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.
NIP: 19700313 199512 1 001

(Pembimbing II)

.....

Dr. Diah Puspito Wulandari, S.T., M.Sc..
NIP: 19801219 200501 2 001

(Penguji I)

.....

Dr. Arief Kurniawan, S.T., M.T..
NIP: 19740907 200212 1 001

(Penguji II)

.....

Arta Kusuma Hernanda, S.T., M.T..
NIP: 1996202311024

(Penguji III)

.....

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

Dr. Arief Kurniawan, S.T., M.T..
NIP. 19740907 200212 1 001

SURABAYA
Januari, 2025

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

DEVELOPMENT OF AUTONOMOUS WHEELCHAIR WITH ESP32-CAM BASED ON YOLOv11

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Engineering at
Undergraduate Study Program of Computer Engineering
Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology

By: **Aldifahmi Sihotang**
NRP. 0721 18 4000 0039

Approved by Final Project Examiner Team:

Dr. Eko Mulyanto Yuniarno, S.T., M.T. (Advisor I)
NIP: 19680601 1995121 1 009

.....

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T. (Co-Advisor II)
NIP: 19700313 199512 1 001

.....

Dr. Diah Puspito Wulandari, S.T., M.Sc.. (Examiner I)
NIP: 19801219 200501 2 001

.....

Dr. Arief Kurniawan, S.T., M.T.. (Examiner II)
NIP: 19740907 200212 1 001

.....

Arta Kusuma Hernanda, S.T., M.T.. (Examiner III)
NIP: 1996202311024

.....

Acknowledged,
Head of Computer Engineering Department F-ELECTICS - ITS

Dr. Arief Kurniawan, S.T., M.T..
NIP. 19740907 200212 1 001

SURABAYA
January, 2025

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa / NRP : Aldifahmi Sihotang / 0721 18 4000 0039
Departemen : Teknik Komputer
Dosen Pembimbing / NIP : Dr. Eko Mulyanto Yuniarno, S.T., M.T. / 19680601 1995121 1 009

Dengan ini menyatakan bahwa Tugas Akhir dengan judul *"PENGEMBANGAN KURSI RODA OTONOM DENGAN ESP32-CAM BERBASIS YOLOv11"* adalah hasil karya sendiri, berfsifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, January 2025

Mengetahui
Dosen Pembimbing

Mahasiswa

Dr. Eko Mulyanto Yuniarno, S.T., M.T.
NIP. 19680601 1995121 1 009

Aldifahmi Sihotang
NRP. 0721 18 4000 0039

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : Aldifahmi Sihotang / 0721 18 4000 0039
Department : Computer Engineering
Advisor / NIP : Dr. Eko Mulyanto Yuniarno, S.T., M.T. / 19680601 1995121 1 009

Hereby declared that the Final Project with the title of "*DEVELOPMENT OF AUTONOMOUS WHEELCHAIR WITH ESP32-CAM BASED ON YOLOv11*" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with provisions that apply at Sepuluh Nopember Institute of Technology.

Surabaya, January 2025

Acknowledged

Advisor

Student

Dr. Eko Mulyanto Yuniarno, S.T., M.T.
NIP. 19680601 1995121 1 009

Aldifahmi Sihotang
NRP. 0721 18 4000 0039

[Halaman ini sengaja dikosongkan]

ABSTRAK

Nama Mahasiswa : Aldifahmi Sihotang
Judul Tugas Akhir : PENGEMBANGAN KURSI RODA OTONOM DENGAN *ESP32-CAM* BERBASIS *YOLOv11*
Pembimbing : 1. Dr. Eko Mulyanto Yuniarno, S.T., M.T.
 2. Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.

Penelitian ini bertujuan untuk mengembangkan sistem kendali kursi roda otonom yang dapat mengikuti pergerakan manusia secara real-time. Sistem ini mengintegrasikan algoritma deteksi objek *YOLOv11* dan pelacakan gerakan tubuh menggunakan *MediaPipe Pose*, dengan memanfaatkan sudut pandang optimal sebagai dasar pengambilan data visual. Kamera ditempatkan pada kacamata pengguna untuk mendapatkan sudut pandang yang optimal dalam mendekripsi dan melacak pergerakan pengguna. Sistem ini dirancang agar dapat beroperasi secara nirkabel menggunakan modul *ESP32*, memberikan fleksibilitas dan efisiensi dalam mengendalikan pergerakan kursi roda. Pengujian dilakukan dalam lingkungan terkendali untuk memastikan keakuratan dan kecepatan deteksi serta pelacakan gerakan pengguna. Hasil penelitian menunjukkan bahwa sistem dapat mengikuti pergerakan pengguna dengan akurat dan responsif, memberikan kontribusi signifikan terhadap pengembangan teknologi mobilitas kesehatan yang lebih cerdas dan mandiri.

Kata Kunci: Kursi Roda Otonom, *YOLOv11*, *MediaPipe Pose*, Sudut Pandang Optimal, Deteksi Gerakan, Kendali Nirkabel, Mobilitas Kesehatan

[Halaman ini sengaja dikosongkan]

ABSTRACT

*Name : Aldifahmi Sihotang
Title : DEVELOPMENT OF AUTONOMOUS WHEELCHAIR WITH ESP32-CAM
BASED ON YOLOv11
Advisors : 1. Dr. Eko Mulyanto Yuniaro, S.T., M.T.
2. Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.*

The objective of this study is to develop an autonomous wheelchair control system capable of real-time trajectory tracking. The system integrates the object detection algorithm YOLOv11 and body motion tracking using MediaPipe Pose, with the optimal viewing angle serving as the basis for visual data collection. The camera is positioned on the user's glasses to ascertain the optimal viewing angle for detecting and tracking user movement. This system is designed to operate wirelessly using the ESP32 module, thereby facilitating flexibility and efficiency in controlling wheelchair movement. Testing was conducted in a controlled environment to ensure the accuracy and speed of user movement detection and tracking. The results demonstrated that the system can follow user movement with precision and responsiveness, thereby making a substantial contribution to the advancement of intelligent and independent health mobility technology.

Keywords: Autonomous Wheelchair, YOLOv11, MediaPipe Pose, Optimal Viewing Angle, Motion Detection, Wireless Control, Health Mobility.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT, atas rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penelitian ini yang berjudul **"PENGEMBANGAN KURSI RODA OTONOM DENGAN ESP32-CAM BERBASIS YOLOv11"**.

Penelitian ini disusun dalam rangka pemenuhan Tugas Akhir sebagai syarat kelulusan Mahasiswa ITS. Oleh karena itu, penulis mengucapkan banyak terima kasih kepada

1. Bapak Dr.Supeno Mardi Susiko Nugroho, ST.,MT, selaku Kepala Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember
2. Bapak Dr. Eko Mulyanto Yuniarno, S.T., M.T. dan Bapak Dr.Supeno Mardi Susiko Nugroho, ST.,MT, selaku Dosen Pembimbing telah memberikan arahan selama pengerjaan tugas akhir ini.
3. Bapak/ibu selaku dosen penguji I, Bapak/ibu selaku dosen penguji II dan Bapak/ibu selaku dosen penguji III yang telah memberikan saran dan revisi agar pengerjaan Buku Tugas Akhir ini dapat menjadi lebih baik.
4. Bapak-Ibu dosen pengajar Departemen Teknik Komputer, atas ilmu dan pengajaran yang telah diberikan kepada penulis selama ini
5. H. Nauli Sihotang, S.Ag., M.Ag. dan Hj. Jamilah Tanjung, S.Pd., Orang tua saya tercinta yang selalu mendukung dan senantiasa menyayangi saya sedari kecil hingga dewasa.
6. Teman - teman lab B300 dan B201 serta teman - teman Departemen Teknik Komputer lainnya

Akhir kata, semoga penelitian ini dapat memberikan manfaat kepada banyak pihak, penulis menyadari jika skripsi ini masih belum sempurna, dikarenakan keterbatasan ilmu yang dimiliki. Untuk itu penulis mengharapkan saran dan kritik yang bersifat membangun kepada penulis untuk menuai hasil yang lebih baik lagi.

Surabaya, Januari 2025

Aldifahmi Sihotang

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
DAFTAR PROGRAM	xv
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Permasalahan	1
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Manfaat	2
2 TINJAUAN PUSTAKA	3
2.1 Hasil penelitian terdahulu	3
2.1.1 Deteksi Objek Menggunakan YOLO V3	3
2.1.2 Perancangan Sistem Kontrol Motor Kursi Roda	3
2.1.3 Deteksi Objek Berbasis ESP32-CAM	3
2.2 Object Detection	4
2.3 Convolutional Neural Network (CNN)	4
2.3.1 Lapisan Konvolusi (<i>Convolutional Layer</i>)	5
2.3.2 Lapisan Aktivasi (<i>Activation Layer</i>)	5
2.3.3 Lapisan Pooling (<i>Pooling Layer</i>)	5
2.3.4 Lapisan Fully Connected (<i>Fully Connected Layer</i>)	5
2.4 Pose Estimation	6
2.5 YOLO (<i>You Only Look Once</i>)	6

2.5.1	YOLOv8	6
2.5.2	YOLOv10	8
2.5.3	YOLOv11	9
2.6	MediaPipe	11
2.6.1	MediaPipe Pose	11
2.7	Classification Performance	12
2.8	Evaluation Metrics	12
2.8.1	Precision	13
2.8.2	Recall	13
2.8.3	Mean Average Precision (mAP)	14
2.8.4	Intersection over Union (IoU)	14
2.9	Tinjauan Pustaka BoT-SORT	15
2.9.1	Kalman Filter	15
2.9.2	<i>Camera Motion Compensation (CMC)</i>	17
2.9.3	Fusi IoU - Re-ID	18
2.10	<i>RoboFlow</i>	19
2.11	OBS Studio	20
2.12	ESP32 Devkit V1	20
2.13	Motor Driver H-Bridge	21
2.14	Kursi Roda Elektrik KY-123	22
3	DESAIN DAN IMPLEMENTASI	23
3.1	Deskripsi Sistem	23
3.1.1	Komponen Sistem	23
3.1.2	Arsitektur Sistem	23
3.2	Perangkat Keras	23
3.2.1	Kamera	24
3.2.2	Unit Kontrol	24
3.2.3	ESP32	25
3.2.4	Driver Motor L298N	26
3.2.5	Skematik Alat	26
3.3	Perangkat Lunak	27
3.3.1	Dataset Citra	28
3.3.2	Labeling	28
3.3.3	Klasifikasi YOLOv11	28

3.3.4	Estimasi Pose MediaPipe	29
3.3.5	Pemrosesan Citra	30
3.3.6	Komunikasi Data	31
3.3.7	Kontrol Motor	31
3.3.8	Kode Program	31
4	PENGUJIAN DAN ANALISIS	35
4.1	Skenario Pengujian	35
4.2	Hasil Pengujian Performa Menggunakan Confusion Matrix	36
4.3	Pengujian Berdasarkan FPS	39
4.4	Pengujian Berdasarkan Regulator Time	40
4.5	Pengujian Keberhasilan Tracking	41
4.6	Pengujian Tingkat Pencahayaan	42
4.7	Pengujian Kesesuaian Jarak Deteksi	43
4.8	Performa Pergerakan Mengikuti Objek	44
4.8.1	Percobaan Dalam Frame	44
4.8.2	Percobaan Bergerak Maju	48
4.8.3	Percobaan Belok Kiri	50
4.8.4	Percobaan Belok Kanan	52
4.8.5	Percobaan Luar Frame	54
4.9	Performa Keberhasilan Mengikuti	55
4.10	Pembahasan Hasil	56
4.10.1	Performa Deteksi Objek	56
4.10.2	Kecepatan Pemrosesan (FPS)	56
4.10.3	Regulator Time	56
4.10.4	Keberhasilan Tracking	56
4.10.5	Kesesuaian Tingkat Pencahayaan	57
4.10.6	Kesesuaian Jarak Deteksi	57
4.10.7	Performa Pergerakan Mengikuti Objek	57
4.10.8	Performa Keberhasilan Mengikuti	58
5	PENUTUP	59
5.1	Kesimpulan	59
5.2	Saran	59
DAFTAR PUSTAKA		61

LAMPIRAN	63
Kode Program	63
BIOGRAFI PENULIS	77

DAFTAR GAMBAR

2.1	An simple CNN architecture.	4
2.2	Arsitektur YOLOv8.	7
2.3	YOLOv8 Pose.	8
2.4	Arsitektur YOLOv10.	9
2.5	Arsitektur Yolov11	10
2.6	Modul Bottleneck, C3, C2F, dan C3K2	10
2.7	MediaPipe 3D.	11
2.8	MediaPipe Pose.	12
2.9	Matrix Konfusi.	12
2.10	Intersection over Union.	15
2.11	Kalman Filter bbox.	16
2.12	Kompensasi Gerakan Kamera	18
2.13	Interface RoboFlow	19
2.14	Interface OBS Studio	20
2.15	Gambar ESP32 Devkit V1	21
2.16	Gambar Motor Driver H-Bridge	21
2.17	Gambar Kursi Roda Elektrik KY-123	22
3.1	Skematik kontrol motor kursi roda	27
3.2	Flowchart program python	32
3.3	Flowchart regulasi arah	32
3.4	Flowchart ESP-CAM	33
3.5	Flowchart ESP Motor	34
4.1	Input Layer Pelatihan Pertama	36
4.2	Loss Plot	36
4.3	Summary of Training Results	37
4.4	Confusion Matrix	37
4.5	F1-Confidence Curve	38
4.6	Inference Results	38
4.7	FPS Trend	39

4.8	Regulator Time Plot	40
4.9	Tracking Numeral	42
4.10	Detection and Transmission Plots.	47
4.11	Straight Movement Plots	49
4.12	Left Turn Plots	51
4.13	Right Turn Plots	53
4.14	Dokumentasi Jalur	55
4.15	Pie Chart Performa Keberhasilan Sistem	55

DAFTAR TABEL

3.1	Kode Instruksi dari Hasil Klasifikasi	25
3.2	Tabel Keypoint yang digunakan	29
4.1	Spesifikasi Laptop MSI GF63 untuk Unit Kontrol	39
4.2	Data Tracking Success	41
4.3	Lighting Level Evaluation	42
4.4	Data Jarak (<1m) untuk Diam	43
4.5	Data Status Frame (Dalam Frame)	44
4.5	Data Status Frame (Dalam Frame)	45
4.5	Data Status Frame (Dalam Frame)	46
4.6	Summary of Data Transmission and Detection	47
4.7	Data Performa Bergerak Maju	48
4.8	Summary of Data During Straight Movement	49
4.9	Data Performa Belok (Kiri)	50
4.10	Summary of Data During Left Turn	51
4.11	Data Performa Belok (Kanan)	52
4.12	Summary of Data During Right Turn	53
4.13	Data Status Frame (Luar Frame)	54

[Halaman ini sengaja dikosongkan]

DAFTAR PROGRAM

5.1	Program Deteksi Fusi pada Unit Kontrol.	63
5.2	Program Stream Frame dan Forward Kode Instruksi.	68
5.3	Program Pengolahan Kode Instruksi Pada Motor.	72

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi *Internet of Things (IoT)* dan *deep learning* semakin banyak diadopsi dalam pengembangan sistem kendali otonom, khususnya untuk aplikasi kesehatan dan mobilitas. Kursi roda otonom merupakan salah satu solusi yang dapat meningkatkan kualitas hidup penyandang disabilitas dengan memberikan kemandirian dalam bergerak. Salah satu teknologi utama yang dapat mendukung pengembangan kursi roda otonom adalah modul *ESP32*. Berdasarkan penelitian yang dilakukan oleh *Ekatama* (2024), *ESP32* mampu mengendalikan perangkat secara nirkabel dengan memanfaatkan konektivitas *Wi-Fi* dan *Bluetooth*, memberikan fleksibilitas yang lebih besar dalam mengontrol kursi roda. Penggunaan modul ini memungkinkan pengguna untuk mengontrol kursi roda secara efektif tanpa perlu mengandalkan kontrol manual yang terbatas [1].

Selain aspek kendali nirkabel, sistem otonom yang mengikuti pergerakan pengguna membutuhkan teknologi yang mampu mendeteksi gerakan tubuh secara real-time. Penelitian oleh *Wijaya et al.* (2022) telah menunjukkan keandalan algoritma *YOLO V3* dalam mendeteksi objek secara cepat dan akurat, yang dapat digunakan untuk melacak pergerakan manusia. Namun, dalam penelitian ini, algoritma *YOLOv11*, versi terbaru dari *YOLO*, diintegrasikan dengan *MediaPipe Pose*, sebuah framework yang dapat mendeteksi dan melacak posisi tubuh manusia. Kombinasi kedua teknologi ini memungkinkan sistem untuk secara akurat mengikuti gerakan pengguna berdasarkan sudut pandang optimal yang diperoleh dari kamera yang dipasang pada kacamata pengguna. Dengan pendekatan ini, kursi roda dapat mengikuti pergerakan pengguna secara alami, mendukung otonomi yang lebih tinggi [2].

Sistem kursi roda otonom yang dikembangkan dalam penelitian ini memanfaatkan modul *ESP32* sebagai perangkat keras utama yang mengontrol keseluruhan sistem secara nirkabel. Dalam penelitian yang dilakukan oleh *Narwaria et al.* (2024), *ESP32-CAM* telah terbukti mampu menangkap dan mengolah data visual dengan efisiensi tinggi, yang dapat diterapkan untuk berbagai aplikasi berbasis *IoT*. Sistem yang dikembangkan tidak hanya dirancang untuk mendeteksi objek, tetapi lebih berfokus pada pelacakan gerakan tubuh pengguna melalui teknologi *MediaPipe Pose*. Dengan integrasi ini, sistem dapat memastikan bahwa pergerakan kursi roda tetap sejalan dengan pergerakan pengguna tanpa memerlukan input manual tambahan [3].

1.2 Permasalahan

Dari permasalahan tersebut maka pemantauan manusia pada kursi roda dengan fokus pada kemandirian menghadapi beberapa tantangan utama:

- 1. Keterbatasan Sistem Kursi Roda Otonom Eksisting:** Sistem kursi roda otonom yang ada saat ini mungkin belum mampu mendeteksi dan mengikuti pengguna dengan efisien karena penempatan kamera yang kurang optimal dan penggunaan algoritma deteksi yang kurang canggih.

2. **Tantangan dalam Deteksi dan Pelacakan Gerakan Manusia secara Real-Time:** Kesulitan dalam mendeteksi dan melacak pergerakan manusia secara akurat dan real-time, terutama dalam lingkungan yang dinamis, akibat keterbatasan algoritma dan perangkat keras.
3. **Integrasi Algoritma Deteksi Lanjutan dengan Sudut Pandang Optimal:** Tantangan dalam mengintegrasikan algoritma deteksi seperti *YOLOv11* dan *MediaPipe Pose* dengan sistem pengambilan gambar yang memiliki sudut pandang optimal untuk meningkatkan akurasi dan responsivitas kursi roda.

1.3 Tujuan

Tujuan dari penelitian ini adalah mengembangkan sistem kursi roda otonom yang dapat mengikuti pergerakan pengguna secara real-time, dengan rancangan dan implementasi sistem kendali kursi roda yang dapat mengikuti pergerakan pengguna dengan memanfaatkan sudut pandang optimal dari kamera yang ditempatkan pada kacamata.

1.4 Batasan Masalah

Batasan-batasan dari penelitian ini diharapkan memberikan kontribusi signifikan dalam bidang pemantauan manusia pada kursi roda dengan fokus pada kemandirian dan kontrol otomatis, dengan:

1. **Penempatan Kamera Terbatas pada Sudut Pandang Optimal:** Sistem hanya menggunakan kamera yang ditempatkan pada kacamata pengguna, sehingga deteksi terbatas pada objek dan pergerakan yang berada dalam garis pandang pengguna.
2. **Penggunaan Algoritma dan Perangkat Keras Tertentu:** Sistem deteksi dan pelacakan dibatasi pada penggunaan algoritma *YOLOv11* dan *MediaPipe Pose*, serta perangkat keras yang mendukung implementasi algoritma tersebut.
3. **Lingkungan Operasional Terkendali:** Pengujian dan implementasi sistem dilakukan dalam lingkungan yang terkendali, seperti di dalam ruangan dengan kondisi pencahayaan dan hambatan yang minimal.
4. **Keterbatasan Pemrosesan Waktu Nyata:** Kemampuan pemrosesan real-time sistem dibatasi oleh kapasitas perangkat keras yang digunakan, sehingga mungkin tidak optimal dalam situasi dengan kompleksitas tinggi.

1.5 Manfaat

Manfaat pada penelitian ini untuk membuat sistem yang dapat mendeteksi manusia untuk mengontrol gerak dari kursi roda.

BAB II

TINJAUAN PUSTAKA

2.1 Hasil penelitian terdahulu

Pada subbab berikut akan dijabarkan penelitian terdahulu.

2.1.1 Deteksi Objek Menggunakan YOLO V3

Penelitian oleh *Wijaya et al.* (2022) berfokus pada pengembangan sistem deteksi objek yang menggunakan *algoritma YOLO V3* untuk meningkatkan keamanan dalam sistem mobilitas. *Algoritma YOLO V3* merupakan salah satu metode *deep learning* yang dirancang untuk melakukan deteksi objek secara cepat dan akurat pada gambar atau video. Penelitian ini memanfaatkan *YOLO V3* untuk mendeteksi rintangan yang ada di lingkungan sekitar sistem, sehingga dapat membantu dalam menghindari potensi bahaya yang diakibatkan oleh tabrakan dengan objek yang tidak terdeteksi.

Fitur utama dalam penelitian ini adalah implementasi *YOLO V3* sebagai algoritma deteksi objek *real-time*. Algoritma ini memiliki keunggulan dalam kecepatan dan akurasinya, memungkinkan sistem untuk mendeteksi berbagai objek dengan lebih cepat dibandingkan metode deteksi lainnya. *YOLO V3* juga memungkinkan sistem untuk secara simultan mengenali beberapa objek dalam satu *frame* gambar, yang sangat penting dalam lingkungan yang dinamis. Penelitian ini menekankan bahwa penggunaan *YOLO V3* dapat secara signifikan meningkatkan keamanan sistem yang memerlukan deteksi objek secara *real-time* [2].

2.1.2 Perancangan Sistem Kontrol Motor Kursi Roda

Penelitian yang dilakukan oleh *Ekatama* (2024) membahas tentang perancangan sistem kontrol motor kursi roda yang dioperasikan secara nirkabel dengan menggunakan *ESP32*. Dalam penelitian ini, *ESP32* dimanfaatkan sebagai pusat kendali yang mampu mengatur pergerakan motor kursi roda tanpa menggunakan koneksi kabel. Penggunaan koneksi nirkabel ini memungkinkan pengendalian kursi roda dari jarak jauh melalui perangkat yang mendukung *Wi-Fi* atau *Bluetooth*, menawarkan kemudahan dan kenyamanan bagi pengguna dalam mengendalikan kursi roda mereka.

Fitur utama dari penelitian ini melibatkan penggunaan *ESP32* sebagai platform utama untuk mengatur pergerakan motor kursi roda. *ESP32* dipilih karena kemampuannya dalam menyediakan komunikasi nirkabel yang stabil dan efisien. Selain itu, penelitian ini menyoroti bagaimana sistem yang dirancang dapat beroperasi dengan konsumsi daya yang rendah, yang merupakan faktor penting dalam perangkat yang diharapkan memiliki masa penggunaan yang lama. Dengan demikian, sistem ini dioptimalkan untuk menjaga efisiensi daya tanpa mengorbankan kinerja atau responsivitas [1].

2.1.3 Deteksi Objek Berbasis ESP32-CAM

Penelitian yang dilakukan oleh *Narwaria et al.* (2024) mengeksplorasi penggunaan *ESP32-CAM* dalam mengembangkan sistem deteksi objek berbasis *YOLO*. *ESP32-CAM* adalah modul

kamera yang dirancang untuk menangkap gambar secara efisien dan cocok untuk aplikasi *Internet of Things (IoT)*. Penelitian ini menggunakan *ESP32-CAM* bersama dengan algoritma *YOLO* untuk mendeteksi dan mengenali objek secara *real-time*. Sistem ini dirancang untuk menangani kebutuhan deteksi objek di berbagai aplikasi, seperti sistem pengawasan, pengenalan objek, dan visi komputer.

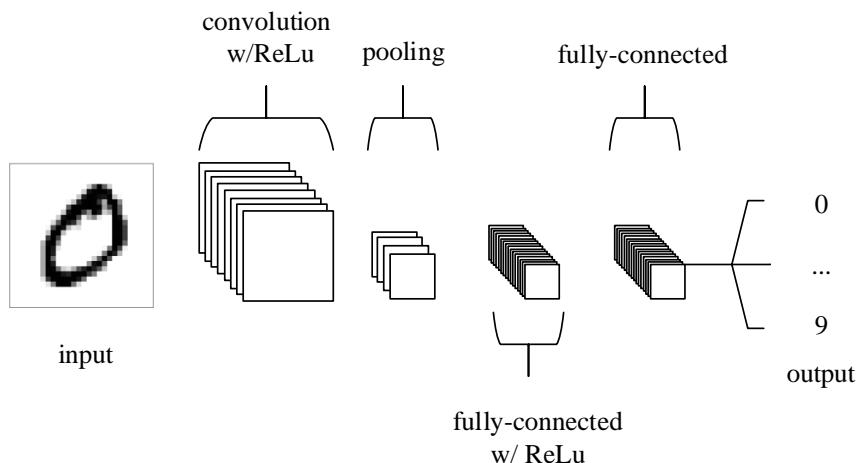
Fitur utama dari penelitian ini adalah kombinasi antara *ESP32-CAM* dan *algoritma YOLO*. *ESP32-CAM* memberikan platform *hardware* yang hemat daya dan biaya rendah, sementara *YOLO* menyediakan kemampuan deteksi objek yang cepat dan akurat. Penelitian ini menunjukkan bahwa penggunaan *Python* dan *OpenCV* sebagai *tools* dalam memrogram dan mengolah data visual memungkinkan sistem untuk menangani pemrosesan gambar dan deteksi objek dengan efisiensi yang tinggi. Solusi ini menunjukkan potensi besar dalam mengembangkan sistem deteksi objek untuk berbagai aplikasi *real-time* [3].

2.2 Object Detection

Object detection atau deteksi objek adalah teknologi inti dalam banyak aplikasi modern seperti sistem pengawasan, pengenalan wajah, dan kendaraan otonom. Teknologi ini digunakan untuk mendeteksi dan mengklasifikasikan berbagai objek dalam gambar atau video. Dalam konteks penelitian ini, *object detection* berperan penting untuk mendeteksi keberadaan manusia sehingga kursi roda otonom dapat mengikuti gerakan pengguna secara *real-time*. Perkembangan pesat dalam *deep learning*, terutama dengan penggunaan *Convolutional Neural Network* (CNN), telah mengoptimalkan kemampuan *object detection*. CNN, dengan berbagai lapisannya seperti lapisan konvolusi, aktivasi, *pooling*, dan *fully connected*, menjadi fondasi dari banyak metode deteksi objek saat ini, termasuk *YOLO* (You Only Look Once).

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah jenis jaringan saraf tiruan yang dirancang khusus untuk pengolahan data yang memiliki struktur grid, seperti gambar. CNN terdiri dari serangkaian lapisan yang bekerja sama untuk mengekstraksi dan menganalisis fitur dari data input, membuatnya sangat efektif dalam tugas-tugas seperti klasifikasi gambar, segmentasi, dan deteksi objek[4].



Gambar 2.1: An simple CNN architecture.

2.3.1 Lapisan Konvolusi (*Convolutional Layer*)

Lapisan konvolusi adalah fondasi dari CNN yang bertugas mengekstraksi fitur-fitur penting dari input gambar. Dengan menerapkan filter yang dipelajari selama pelatihan, lapisan ini mampu mengenali elemen-elemen dasar seperti tepi, tekstur, dan pola dalam gambar. Setiap filter dalam lapisan konvolusi bertanggung jawab untuk mendeteksi fitur tertentu, dan hasilnya disusun dalam peta fitur (*feature map*) yang memberikan representasi visual dari elemen-elemen yang terdeteksi. Formula untuk operasi konvolusi dapat dituliskan sebagai berikut:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau \quad (2.1)$$

2.3.2 Lapisan Aktivasi (*Activation Layer*)

Lapisan aktivasi adalah komponen yang menambahkan non-linearitas ke dalam jaringan, yang memungkinkan model untuk mempelajari hubungan yang lebih kompleks. Fungsi aktivasi seperti *ReLU* (*Rectified Linear Unit*) digunakan untuk mengaktifkan neuron hanya jika outputnya positif, sehingga mengurangi kompleksitas komputasi dan mempercepat proses pelatihan. Non-linearitas ini sangat penting untuk mengatasi masalah yang melibatkan data dengan struktur yang rumit, seperti deteksi objek dalam gambar.

2.3.3 Lapisan Pooling (*Pooling Layer*)

Lapisan *pooling* digunakan untuk mengurangi dimensi peta fitur sambil mempertahankan informasi yang paling relevan. Metode *pooling* seperti *max pooling* mengambil nilai maksimum dalam setiap area *pooling*, yang membantu jaringan menjadi lebih tahan terhadap variasi kecil dalam input, seperti perubahan skala atau rotasi objek. Dengan mereduksi jumlah data yang harus diproses, *pooling* juga membantu mengurangi risiko *overfitting* dan mempercepat pelatihan. Operasi *max pooling* dapat direpresentasikan sebagai:

$$P(x, y) = \max_{i,j \in \text{PoolRegion}} I(x + i, y + j) \quad (2.2)$$

2.3.4 Lapisan Fully Connected (*Fully Connected Layer*)

Lapisan *fully connected* adalah lapisan terakhir dalam CNN yang menghubungkan setiap neuron di lapisan sebelumnya ke setiap neuron di lapisan ini. Lapisan ini berfungsi untuk menggabungkan semua fitur yang telah diekstraksi oleh lapisan konvolusi dan *pooling*, dan menghasilkan output akhir, seperti prediksi kelas objek. Lapisan *fully connected* memegang peran penting dalam memproses informasi yang dihasilkan dari lapisan-lapisan sebelumnya untuk membuat keputusan akhir tentang klasifikasi atau deteksi. Rumus dasar untuk operasi *fully connected* dapat dituliskan sebagai:

$$y = f \left(\sum_{i=1}^n w_i x_i + b \right) \quad (2.3)$$

Dimana w_i adalah bobot, x_i adalah input, b adalah bias, dan f adalah fungsi aktivasi. Kombinasi antara fungsi aktivasi yang tepat dan pengaturan hyperparameter yang optimal akan semakin meningkatkan performa lapisan fully connected dalam berbagai tugas seperti klasifikasi, deteksi objek, dan lainnya.

2.4 Pose Estimation

Pose estimation adalah teknik untuk mengidentifikasi dan melacak posisi tubuh manusia dalam gambar atau video. Teknik ini biasanya melibatkan deteksi titik-titik kunci (*keypoints*) pada tubuh, seperti sendi atau ujung-ujung anggota tubuh, yang digunakan untuk memodelkan postur atau gerakan individu. *Pose estimation* sangat penting dalam berbagai aplikasi, seperti analisis olahraga, animasi, dan interaksi manusia-mesin.

Dalam penelitian ini, *pose estimation* digunakan untuk memastikan gerakan pengguna dapat diikuti dengan akurasi tinggi. Dengan memanfaatkan *pose estimation*, sistem dapat memahami arah dan intensitas gerakan pengguna, yang memungkinkan kursi roda untuk merespons secara tepat dan efisien.

2.5 YOLO (*You Only Look Once*)

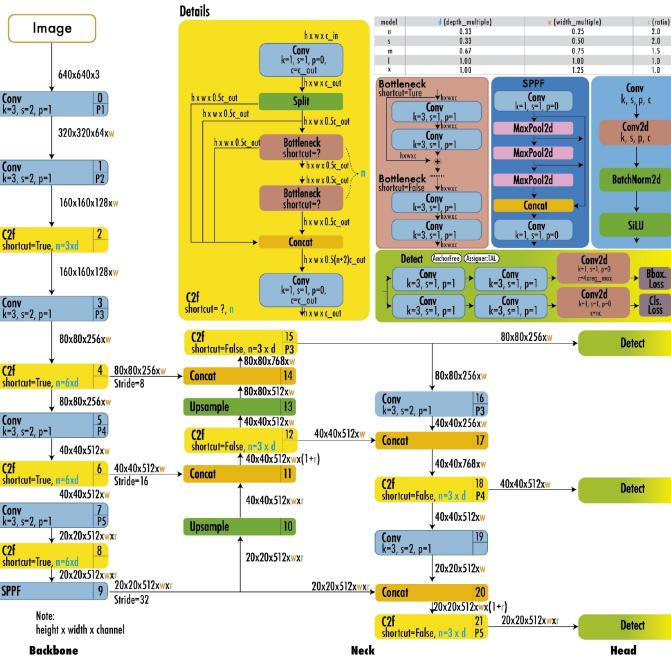
YOLO, yang dikembangkan oleh Joseph Redmon, memperkenalkan pendekatan *End-to-End* untuk deteksi objek dalam *Real-Time*. Nama *YOLO*, singkatan dari "*You Only Look Once*" (Anda Hanya Melihat Sekali), mencerminkan kemampuan model ini dalam menyelesaikan tugas deteksi hanya dengan satu kali pemrosesan jaringan. Hal ini berbeda dengan pendekatan sebelumnya, yang menggunakan teknik *sliding window* dan memerlukan pengklasifikasi yang harus dijalankan berkali-kali pada setiap gambar, atau metode lain yang memecah tugas menjadi dua langkah terpisah: pertama, mengidentifikasi daerah yang mungkin mengandung objek (*region proposals*), dan kedua, menjalankan pengklasifikasi pada daerah yang telah diidentifikasi tersebut. Selain itu, *YOLO* memanfaatkan *output* yang lebih sederhana dengan hanya menggunakan regresi untuk memprediksi hasil deteksi, berlawanan dengan metode seperti *Fast R-CNN* yang memisahkan tugas menjadi dua *output* terpisah: probabilitas klasifikasi dan regresi untuk koordinat *bounding box*.

2.5.1 YOLOv8

YOLOv8 adalah salah satu model deteksi objek yang sangat efisien, menggabungkan kecepatan tinggi dengan akurasi yang relatif tinggi. Arsitektur ini terdiri dari beberapa lapisan utama: *Backbone*, *Neck*, dan *Head*. *Backbone* bertugas untuk mengekstraksi fitur-fitur dasar dari gambar input. Kemudian, *Neck* menggabungkan informasi dari berbagai lapisan untuk menghasilkan representasi fitur yang lebih kaya, yang kemudian diproses oleh *Head* untuk menghasilkan prediksi *bounding box*, label kelas, dan skor *confidence*.

Gambar 2.2 menunjukkan arsitektur lengkap *YOLOv8*, dimulai dari input gambar dengan resolusi 640x640x3. Input ini diproses melalui serangkaian lapisan konvolusi dalam *Backbone*, yang bertugas mengekstraksi fitur penting dari gambar. *Backbone* dirancang untuk secara efisien menangkap informasi visual dari berbagai tingkat abstraksi, yang nantinya menjadi dasar untuk langkah-langkah pemrosesan berikutnya.

Setelah melalui *Backbone*, fitur-fitur tersebut diteruskan ke *Neck*, yang bertugas menggabungkan dan mengolah informasi dari berbagai level resolusi. *Neck* menghasilkan representasi multi-skala yang kaya, memungkinkan jaringan untuk memahami konteks dari objek dalam gambar dengan lebih baik. Akhirnya, *Head* memproses representasi tersebut untuk melakukan prediksi akhir, seperti menentukan *bounding boxes*, label kelas, dan *confidence score* dengan akurasi tinggi.



Gambar 2.2: Arsitektur YOLOv8.

YOLOv8 memprediksi *bounding box* menggunakan kombinasi antara koordinat pusat (bx, by), dimensi *bounding box* (bw, bh), dan skor *confidence* p_c . Formula untuk menghitung koordinat *bounding box* berdasarkan *output* jaringan adalah:

$$\begin{aligned} bx &= \sigma(t_x) + c_x \\ bw &= p_w e^{t_w} \\ by &= \sigma(t_y) + c_y \\ bh &= p_h e^{t_h} \end{aligned} \quad (2.4)$$

Di mana t_x, t_y, t_w, t_h adalah *output* dari model *neural network*, σ adalah fungsi sigmoid, c_x, c_y adalah koordinat *grid cell*, dan p_w, p_h adalah skala *anchor box*.

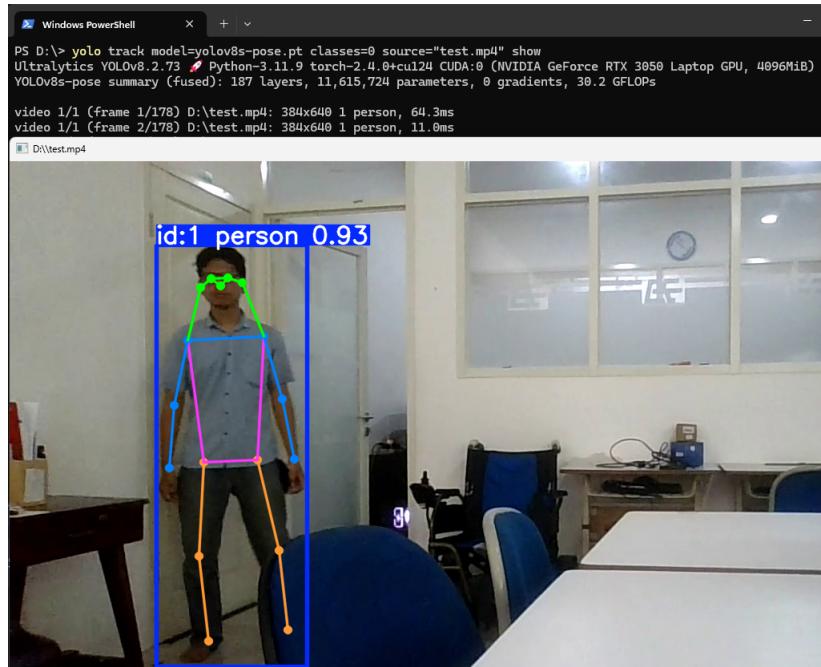
Fungsi *loss* dalam *YOLOv8* terdiri dari beberapa komponen utama yang mengukur perbedaan antara prediksi model dan *ground truth*, serta menyeimbangkan pentingnya prediksi koordinat, *confidence score*, dan klasifikasi objek. Rumus fungsi *loss*-nya adalah:

$$\begin{aligned} \text{Loss} &= \lambda_{\text{coord}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(bx_i - \hat{bx}_i)^2 + (by_i - \hat{by}_i)^2 \right] \\ &\quad + \lambda_{\text{coord}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{bw_i} - \sqrt{\hat{bw}_i})^2 + (\sqrt{bh_i} - \sqrt{\hat{bh}_i})^2 \right] \\ &\quad + \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\ &\quad + \sum_{i=0}^S \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \quad (2.5)$$

Pada persamaan ini, S adalah ukuran grid, B adalah jumlah *bounding boxes* per *grid cell*, $\mathbb{1}_{ij}^{\text{obj}}$ adalah indikator bahwa *bounding box* j di *cell* i memprediksi objek, dan λ_{coord} dan λ_{noobj} adalah hyperparameter yang mengontrol pentingnya masing-masing *loss*.

YOLOv8 Pose

YOLOv8 Pose adalah varian dari *YOLOv8* yang dirancang khusus untuk tugas *pose estimation*. Dengan menggunakan pendekatan yang menggabungkan kecepatan *YOLO* dengan kemampuan deteksi pose yang presisi, *YOLOv8 Pose* mampu mendeteksi titik-titik kunci pada tubuh manusia secara *real-time*. Setiap *bounding box* tidak hanya mengandung informasi tentang lokasi dan ukuran objek, tetapi juga koordinat titik-titik kunci yang terkait dengan pose manusia (misalnya, bahu, siku, lutut, dan sebagainya).

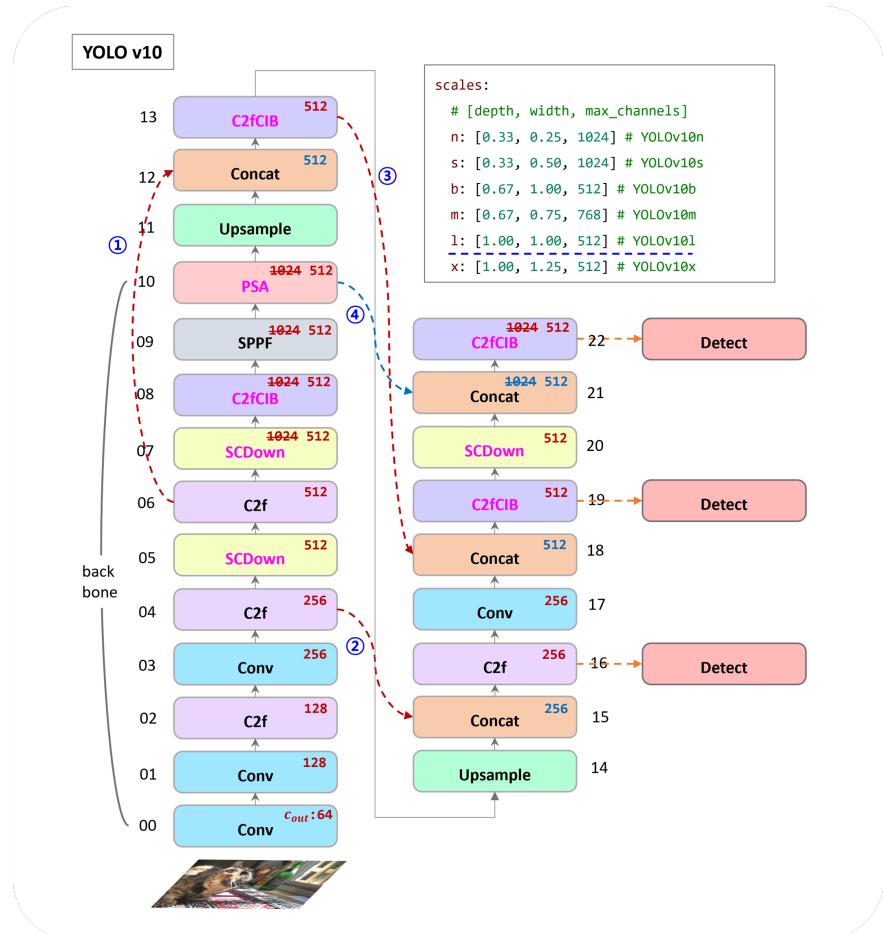


Gambar 2.3: YOLOv8 Pose.

2.5.2 YOLOv10

YOLOv10 merupakan pengembangan lebih lanjut dari *YOLOv8*, dengan beberapa perbaikan dalam efisiensi dan akurasi deteksi objek. Seperti *YOLOv8*, arsitektur *YOLOv10* terdiri dari *Backbone*, *Neck*, dan *Head*, namun dengan penambahan beberapa modul baru seperti *Path Aggregation Network (PSA)* dan *Improved Convolutional Block (C2fCIB)*.

Arsitektur *YOLOv10* dikembangkan dengan memperkenalkan beberapa peningkatan kunci dari dasar-dasar *YOLOv8*. *Backbone* *YOLOv10* tetap berfungsi sebagai ekstraktor fitur utama, namun ditingkatkan dengan modul *SCD (Squeeze-and-Excitation Convolutional Downsample)* dan *C2fCIB*, yang memungkinkan propagasi informasi yang lebih efisien dan reduksi redundansi. Modul *PSA (Path Aggregation Network)* yang baru ditambahkan dalam *Neck* membantu menggabungkan informasi dari berbagai jalur dalam jaringan, memperkaya representasi fitur untuk deteksi multi-skala.

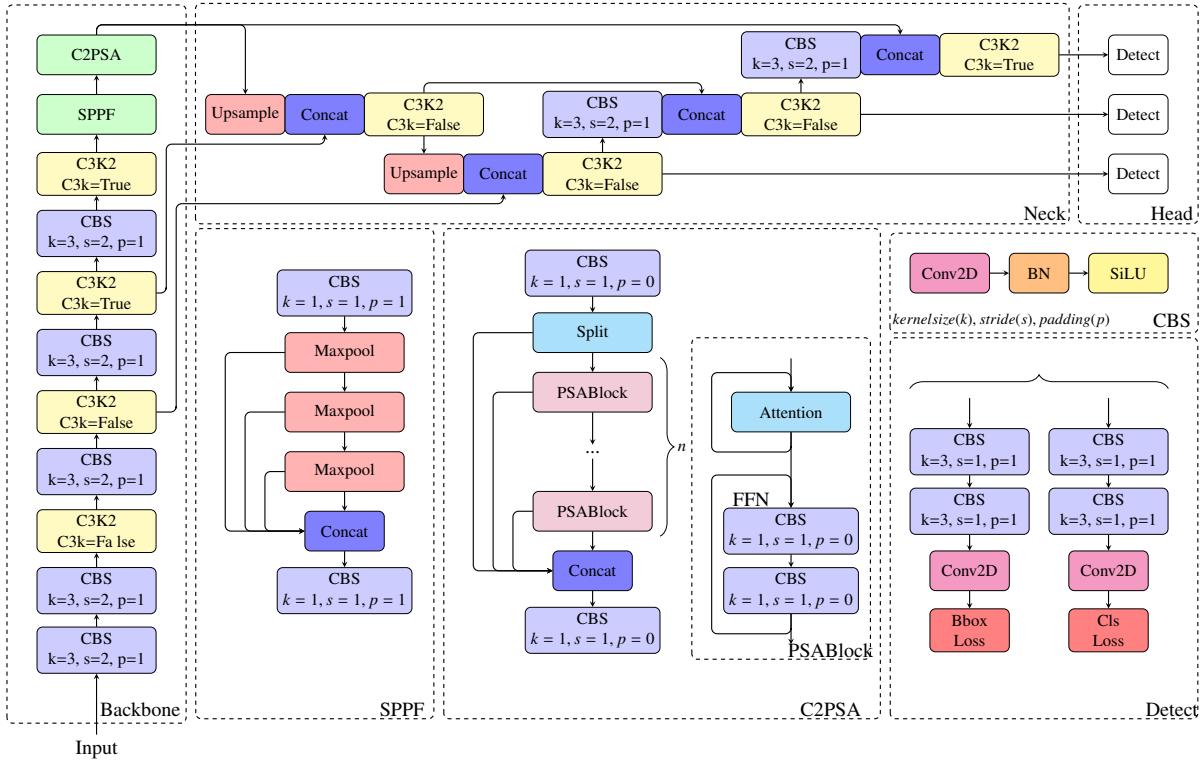


Gambar 2.4: Arsitektur YOLOv10.

2.5.3 YOLOv11

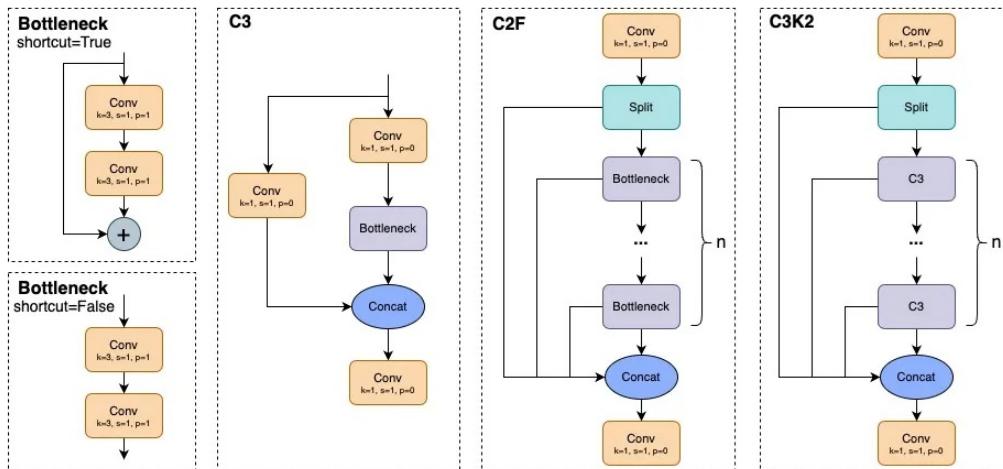
YOLOv11 adalah terobosan terbaru dalam seri detektor objek real-time dari Ultralytics. Meneruskan kemajuan dari pendahulunya, YOLOv11 menghadirkan peningkatan signifikan pada arsitekturnya, menjadikannya solusi yang kuat dan adaptif untuk berbagai aplikasi *computer vision*. Ultralytics memperkenalkan berbagai peningkatan dalam deteksi objek dan arsitektur pembelajaran mendalam. Arsitektur *backbone* dan *neck* yang ditingkatkan memperbaiki ekstraksi fitur, memungkinkan deteksi objek yang lebih akurat dan penanganan tugas yang lebih kompleks.

Efisiensi dan kecepatan juga dipertajam melalui arsitektur yang disempurnakan dan *pipeline* pelatihan yang lebih optimal, sehingga pemrosesan menjadi lebih cepat tanpa mengorbankan akurasi dan performa. YOLOv11 mendukung implementasi di berbagai platform, mulai dari perangkat *edge*, platform *cloud*, hingga sistem dengan *GPU NVIDIA*, membuatnya fleksibel untuk digunakan di berbagai lingkungan. Selain itu, YOLOv11 mendukung beragam tugas, termasuk deteksi objek, segmentasi *instance*, klasifikasi gambar, estimasi *pose*, dan deteksi objek terorientasi (*OBB*). Salah satu pembaruan utama dalam arsitektur YOLOv11 adalah pengenalan modul *C3K2* yang menggantikan modul *C2F* pada YOLOv8, serta penambahan modul *C2PSA* setelah modul *SPPF* untuk meningkatkan kapabilitas deteksi lebih lanjut.



Gambar 2.5: Arsitektur Yolov11

Arsitektur $C3K2$ merupakan versi yang dimodifikasi dari modul $C2F$. Perbedaan utama terletak pada konfigurasi parameter $c3k$. Ketika $c3k$ disetel ke False, modul $C3K2$ berperilaku seperti modul $C2F$, menggunakan struktur *bottleneck* standar. Sebaliknya, ketika $c3k$ disetel ke True, modul *bottleneck* digantikan oleh modul $C3$. Perubahan ini dapat dilihat pada gambar berikut.



Gambar 2.6: Modul Bottleneck, C3, C2F, dan C3K2

Fitur input diubah pada lapisan *feedforward* ke dalam ruang berdimensi lebih tinggi, sehingga hubungan non-linear yang kompleks dapat tertangkap dengan lebih stabil.

2.6 MediaPipe

MediaPipe adalah framework *open-source* yang dikembangkan oleh Google untuk membangun pipeline pemrosesan media yang efisien, termasuk dalam pengolahan gambar dan video. Framework ini menyediakan berbagai modul yang dapat dimanfaatkan untuk aplikasi seperti deteksi wajah, pelacakan tangan, dan estimasi pose.

Kerangka kerja *MediaPipe* menggunakan konsep "graph," di mana setiap node dalam graph berfungsi sebagai "calculator" yang menjalankan tugas spesifik, seperti deteksi objek, pelacakan pose, atau segmentasi gambar. Konfigurasi node-node ini dapat disesuaikan melalui *GraphConfig*, yang mendefinisikan topologi dan fungsionalitas dari keseluruhan sistem.

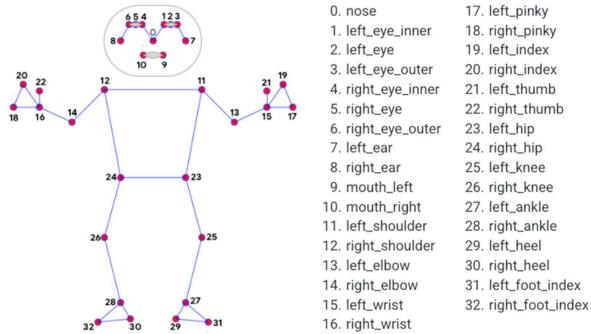


Gambar 2.7: MediaPipe 3D.

Salah satu contoh penggunaan *MediaPipe* yang paling dikenal adalah estimasi pose manusia menggunakan modul *MediaPipe Pose*. Teknik ini mengombinasikan estimasi pose 2D dengan model humanoid yang lebih kompleks, serta menggunakan metode optimasi untuk menghitung sudut sendi dalam pose 3D. Pendekatan ini efektif dalam mengatasi masalah ambiguitas kedalaman pada estimasi pose 3D, dan mampu bekerja secara *real-time*. Visualisasi pose 3D yang dihasilkan memperlihatkan setiap sendi tubuh sebagai titik dan garis penghubung antar sendi, memberikan gambaran yang jelas mengenai posisi dan orientasi tubuh di dalam ruang 3D.

2.6.1 MediaPipe Pose

MediaPipe Pose adalah modul dalam *MediaPipe* yang dirancang khusus untuk mendekati dan melacak pose manusia secara *real-time*. Dengan menggunakan model *machine learning* yang canggih, *MediaPipe Pose* dapat mengidentifikasi hingga 33 titik referensi pada tubuh manusia, memungkinkan sistem untuk memahami dan merespons gerakan pengguna secara cepat dan akurat.



Gambar 2.8: MediaPipe Pose.

2.7 Classification Performance

Classification performance mengacu pada kemampuan model untuk mengklasifikasikan data input ke dalam kategori yang benar. Proses pengklasifikasian memerlukan evaluasi untuk menilai efektivitas model yang telah dikembangkan, biasanya dengan menggunakan set data pengetesan. Salah satu pendekatan evaluasi yang sering digunakan dalam konteks pengklasifikasian adalah *confusion matrix*, yang memberikan visualisasi mengenai kinerja model dalam mengategorikan data secara akurat.

		Predicted Class	
		1 (Positive)	0 (Negative)
Actual Class	1 (Positive)	TP (True Positive)	FN (False Negative) <i>Type II Error</i>
	0 (Negative)	FP (False Positive) <i>Type I Error</i>	TN (True Negative)

Gambar 2.9: Matrix Konfusi.

Confusion matrix ini digunakan untuk mengevaluasi kinerja model klasifikasi biner dan menampilkan empat jenis hasil dari prediksi model: *True Positive* (TP), *False Negative* (FN), *False Positive* (FP), dan *True Negative* (TN). TP terjadi ketika model memprediksi kelas positif dengan benar, sedangkan FN terjadi ketika model salah memprediksi kelas negatif padahal seharusnya positif (disebut juga sebagai *Type II Error*). FP terjadi ketika model salah memprediksi kelas positif padahal seharusnya negatif (disebut juga sebagai *Type I Error*), dan TN terjadi ketika model memprediksi kelas negatif dengan benar.

2.8 Evaluation Metrics

Metrik evaluasi berfungsi sebagai dasar pemahaman dalam membandingkan efektivitas berbagai algoritma dan skenario yang berbeda. Melalui evaluasi yang teliti, perbandingan akurasi antara berbagai teknik deteksi objek dapat dilakukan, serta tingkat keakuratan yang dicapai dapat dinilai dengan tepat. Hal ini menjadi sangat penting dalam pemilihan algoritma yang paling sesuai untuk tugas deteksi tertentu. Metrik seperti *akurasi*, *presisi*, dan *recall* digunakan untuk mengevaluasi efektivitas model dalam mendeteksi dan mengidentifikasi objek dengan be-

nar. Implementasi metrik-metrik ini diperlukan untuk menentukan model yang paling efisien.

Selain itu, analisis akurasi memberikan wawasan kuantitatif yang signifikan mengenai kinerja algoritma deteksi objek, serta detail lebih lanjut mengenai kemampuan algoritma dalam menghasilkan deteksi yang akurat. Kesalahan yang teridentifikasi melalui metrik evaluasi menjadi langkah penting dalam penelitian deteksi, seperti dalam kasus deteksi asap yang diteliti. Identifikasi ini memudahkan pemahaman mengenai potensi kesalahan dalam algoritma, yang kemudian dapat mendarah pada perbaikan dan peningkatan metode deteksi. Metrik evaluasi juga dimanfaatkan untuk mendukung pengoptimalan hiperparameter algoritma.

Dalam penelitian ini, berbagai metrik evaluasi seperti *presisi*, *recall*, dan *Mean Average Precision* (mAP) telah diterapkan. Dengan menggabungkan metode evaluasi ini, penelitian ini dirancang untuk menyajikan analisis komprehensif mengenai kinerja algoritma deteksi objek yang ditinjau. Penjelasan mengenai konsep dasar metrik evaluasi akan dijabarkan sebagai berikut:

2.8.1 Precision

Precision merupakan salah satu metrik utama yang digunakan untuk mengukur seberapa akurat prediksi dari model terhadap objek yang terdeteksi. Metrik ini menunjukkan proporsi prediksi positif yang benar (*True Positive*) dibandingkan dengan keseluruhan prediksi positif, baik yang benar (*True Positive*) maupun salah (*False Positive*), yang dinyatakan dengan persamaan berikut:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.6)$$

Di mana *TP* (*True Positive*): Prediksi benar, yaitu deteksi objek yang sesuai dengan *ground truth* dan *FP* (*False Positive*): Prediksi salah, yaitu deteksi objek yang tidak sesuai dengan *ground truth*.

Precision sangat berguna dalam situasi di mana kesalahan prediksi positif (*false positive*) harus diminimalkan. Contohnya, dalam aplikasi kursi roda otonom, deteksi yang salah terhadap manusia dapat menyebabkan tindakan yang berbahaya, sehingga *precision* harus dipertahankan pada nilai yang tinggi.

Precision biasanya dikombinasikan dengan metrik lain, seperti *recall*, untuk memberikan gambaran yang lebih lengkap tentang kinerja model.

2.8.2 Recall

Recall atau sensitivitas mengukur kemampuan model dalam mendeteksi semua objek yang ada pada gambar atau video. *Recall* menekankan pada seberapa banyak objek yang benar-benar ada dalam data (*ground truth*) yang berhasil terdeteksi oleh model dengan persamaan berikut:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.7)$$

Di mana *TP* (*True Positive*): Prediksi benar, yaitu objek yang terdeteksi dengan benar dan *FP* (*False Negative*): Prediksi salah, yaitu objek yang ada tetapi tidak terdeteksi.

Metrik ini penting ketika kesalahan berupa kegagalan dalam mendeteksi objek (*false negative*) harus diminimalkan. Dalam konteks pengembangan kursi roda otonom, *recall* sangat penting karena objek seperti manusia harus selalu terdeteksi agar kursi roda dapat mengikuti

secara akurat.

Trade-off Precision dan Recall: Dalam banyak kasus, *precision* dan *recall* memiliki hubungan yang berlawanan. Jika model terlalu konservatif dalam membuat prediksi, *precision* akan tinggi, tetapi *recall* rendah. Sebaliknya, jika model terlalu longgar dalam mendekripsi objek, *recall* tinggi tetapi *precision* menurun. Oleh karena itu, diperlukan keseimbangan antara *precision* dan *recall*, yang biasanya diekspresikan melalui metrik lain seperti *F1-score*.

2.8.3 Mean Average Precision (mAP)

Mean Average Precision (mAP) merupakan metrik komprehensif yang digunakan untuk mengukur performa model deteksi objek secara keseluruhan. *mAP* adalah rata-rata dari *average precision* (AP) di berbagai kelas yang ada dalam dataset.

$$AP = \int_0^1 P(r), dr \quad (2.8)$$

Di mana $P(r)$: *Precision* sebagai fungsi dari *recall* dan dr : Diferensial dari *recall*.

AP dihitung dengan mencari area di bawah kurva *precision-recall (PR curve)* untuk setiap kelas. Setelah AP untuk semua kelas dihitung, rata-rata dari nilai-nilai tersebut akan memberikan nilai *mAP*.

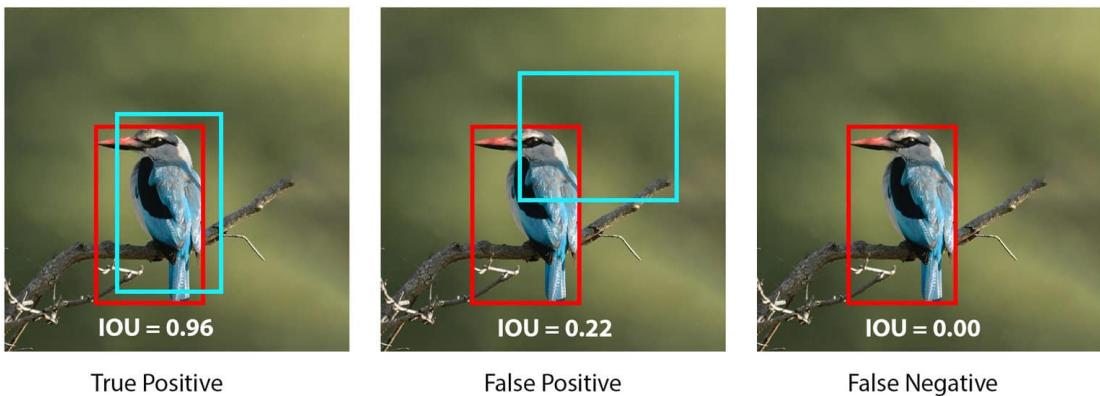
$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.9)$$

Di mana N : Jumlah kelas dalam dataset dan AP_i : *Average precision* untuk kelas ke-i.

mAP memberikan gambaran yang lebih luas mengenai performa detektor objek, karena metrik ini mempertimbangkan baik *precision* maupun *recall* secara bersamaan. Nilai *mAP* ini akan menjadi acuan untuk mengevaluasi performa sistem, seperti mendekripsi berbagai objek seperti manusia, rintangan, dan lingkungan sekitar.

2.8.4 Intersection over Union (IoU)

Intersection over Union (IoU) adalah metrik yang digunakan untuk mengevaluasi keakuratan deteksi posisi objek yang dilakukan oleh model dalam pemrosesan gambar. Area persinggungan antara kotak deteksi yang dihasilkan oleh model dan kotak referensi, yang dikenal sebagai *Ground Truth*, dihitung untuk menilai kinerja model. Rasio ini diperoleh dengan membandingkan luas area irisan dari kedua kotak tersebut terhadap total luas area gabungan yang mereka cakup. Jika kedua kotak tersebut diperlakukan sebagai satu kesatuan, maka *IoU* memberikan skor yang menunjukkan seberapa akurat model dalam memprediksi lokasi objek yang sebenarnya. Nilai *IoU* akan semakin tinggi seiring dengan bertambahnya proporsi area persinggungan relatif terhadap keseluruhan area gabungan.



Gambar 2.10: Intersection over Union.

Kinerja model deteksi objek dilakukan evaluasi kinerja model deteksi objek dengan cara membandingkan area tumpang tindih antara *bounding box* prediksi dan *bounding box ground truth*. Nilai *IoU* berkisar antara 0 hingga 1, di mana nilai yang lebih mendekati 1 menunjukkan tingkat akurasi yang lebih tinggi dalam deteksi dan penentuan lokasi objek.

Dalam proses evaluasi, *bounding box* yang dihasilkan oleh model dibandingkan dengan *bounding box ground truth*, yang ditentukan secara manual sebagai lokasi sebenarnya dari objek dalam citra. Perhitungan *IoU* dilakukan dengan membagi luas area tumpang tindih antara kedua *bounding box* tersebut dengan luas total area gabungan dari keduanya. Persamaan berikut digunakan untuk menghitung nilai *IoU*:

$$IoU = \frac{|A \cap B|}{|A \cup B|}. \quad (2.10)$$

IoU dipilih sebagai alat ukur karena kemampuannya untuk memberikan penilaian yang jelas tentang seberapa akurat model dalam mengidentifikasi dan membatasi objek di berbagai kondisi, termasuk variasi ukuran, orientasi, dan konteks objek dalam citra. Nilai *IoU* yang lebih tinggi diindikasikan sebagai tanda bahwa model dapat diandalkan dalam mendeteksi dan mengidentifikasi objek dengan tingkat presisi yang tinggi.

2.9 Tinjauan Pustaka BoT-SORT

BoT-SORT merupakan salah satu metode multi-object tracking (MOT) yang dikembangkan dengan pendekatan tracking-by-detection. Pada prinsipnya, metode ini memanfaatkan beberapa teknik dari berbagai algoritma sebelumnya, terutama ByteTrack, untuk menyajikan pelacakan yang lebih canggih. Perbaikan pada metode ini bertujuan untuk meningkatkan kinerja pelacakan dalam lingkungan dinamis maupun statis dengan memperbaiki beberapa komponen utama, yang akan dijabarkan sebagai berikut [5]:

2.9.1 Kalman Filter

Kalman Filter (KF) berfungsi untuk memprediksi lokasi objek di frame berikutnya berdasarkan pergerakan sebelumnya. BoT-SORT menggunakan KF dengan vektor status yang mencakup koordinat pusat objek (x_c, y_c), lebar (w), tinggi (h), dan laju perubahan (kecepatan) dari variabel-variabel tersebut ($\dot{x}_c, \dot{y}_c, \dot{w}, \dot{h}$) [6], [7]. Vektor status ini didefinisikan sebagai:

$$\mathbf{x}_k = [x_c(k), y_c(k), w(k), h(k), \dot{x}_c(k), \dot{y}_c(k), \dot{w}(k), \dot{h}(k)]^\top \quad (2.11)$$

$$\mathbf{z}_k = [z_{x_c}(k), z_{y_c}(k), z_w(k), z_h(k)]^\top \quad (2.12)$$

Matriks noise proses (\mathbf{Q}_k) dan noise pengukuran (\mathbf{R}_k) disesuaikan agar lebih sensitif terhadap perubahan dalam frame, yang membantu meningkatkan akurasi pelacakan. Matriks \mathbf{Q}_k dan \mathbf{R}_k ditentukan sebagai berikut:

$$\begin{aligned} \mathbf{Q}_k = & \text{diag}((\sigma_p \hat{w}_{k-1|k-1})^2, (\sigma_p \hat{h}_{k-1|k-1})^2, \\ & (\sigma_p \hat{w}_{k-1|k-1})^2, (\sigma_p \hat{h}_{k-1|k-1})^2, \\ & (\sigma_v \hat{w}_{k-1|k-1})^2, (\sigma_v \hat{h}_{k-1|k-1})^2, \\ & (\sigma_v \hat{w}_{k-1|k-1})^2, (\sigma_v \hat{h}_{k-1|k-1})^2) \end{aligned} \quad (2.13)$$

$$\begin{aligned} \mathbf{R}_k = & \text{diag}((\sigma_m \hat{w}_{k|k-1})^2, (\sigma_m \hat{h}_{k|k-1})^2, \\ & (\sigma_m \hat{w}_{k|k-1})^2, (\sigma_m \hat{h}_{k|k-1})^2) \end{aligned} \quad (2.14)$$

Dengan adanya perubahan ini, prediksi *bounding box* lebih akurat dibandingkan dengan metode Kalman Filter tradisional, terbukti dengan peningkatan nilai HOTA (Higher Order Tracking Accuracy).



Gambar 2.11: Kalman Filter bbox.

Visualisasi bentuk *bounding box* dibandingkan dengan *Kalman filter* yang banyak digunakan [7] (biru putus-putus) dan *Kalman filter* yang diusulkan (hijau). Tampak bahwa lebar *bounding box* yang dihasilkan oleh *Kalman filter* yang diusulkan lebih sesuai dengan objek.

Bounding box biru putus-putus memotong bagian kaki objek (dalam merah), sedangkan *bounding box* hijau mencapai lebar yang diinginkan.

2.9.2 Camera Motion Compensation (CMC)

Pelacak berbasis tracking-by-detection seringkali mengalami masalah ID switch atau false negatives akibat gerakan kamera, terutama dalam situasi dinamis. BoT-SORT mengimplementasikan teknik kompensasi gerakan kamera (Camera Motion Compensation) dengan memanfaatkan affine transformation untuk menghitung transformasi antara dua frame. Dengan melakukan ekstraksi keypoints gambar [8], kemudian menggunakan sparse optical flow [9] untuk pelacakan fitur dengan penolakan outlier lokal berbasis translasi. Matriks affine $A_{k-1}^k \in \mathbb{R}^{2 \times 3}$ diselesaikan menggunakan metode RANSAC [10]. Penggunaan teknik pendaftaran spars memungkinkan pengabaian objek dinamis dalam adegan berdasarkan deteksi, sehingga memiliki potensi untuk memperkirakan gerakan latar belakang dengan lebih akurat.

$$A_{k-1}^k = [\mathbf{M}_{2x2} | \mathbf{T}_{2x1}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \quad (2.15)$$

$$\tilde{\mathbf{M}}_{k-1}^k = \begin{bmatrix} \mathbf{M} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{M} \end{bmatrix}, \quad \tilde{\mathbf{T}}_{k-1}^k = \begin{bmatrix} a_{13} \\ a_{23} \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2.16)$$

$$\hat{\mathbf{x}}'_{k|k-1} = \tilde{\mathbf{M}}_{k-1}^k \hat{\mathbf{x}}_{k|k-1} + \tilde{\mathbf{T}}_{k-1}^k \quad (2.17)$$

$$\mathbf{P}'_{k|k-1} = \tilde{\mathbf{M}}_{k-1}^k \mathbf{P}_{k|k-1} \tilde{\mathbf{M}}_{k-1}^{k^\top} \quad (2.18)$$

Dimana $\mathbf{M} \in \mathbb{R}^{2 \times 2}$ merupakan matriks yang berisi skala dan rotasi dari matriks afine A , dan T mengandung komponen translasi. Trik matematis digunakan dengan mendefinisikan $\tilde{\mathbf{M}}_{k-1}^k \in \mathbb{R}^{8 \times 8}$ dan $\tilde{\mathbf{T}}_{k-1}^k \in \mathbb{R}^8$. Selanjutnya, $\hat{\mathbf{x}}_{k|k-1}$ dan $\hat{\mathbf{x}}'_{k|k-1}$ didefinisikan sebagai vektor status prediksi dari Kalman Filter (KF) pada waktu k , sebelum dan setelah kompensasi gerakan kamera, sedangkan $\mathbf{P}_{k|k-1}$ dan $\mathbf{P}'_{k|k-1}$ didefinisikan sebagai matriks kovarians KF sebelum dan setelah koreksi. Setelah itu, $\hat{\mathbf{x}}'_{k|k-1}$ dan $\hat{\mathbf{P}}'_{k|k-1}$ digunakan dalam langkah update Kalman Filter sebagai berikut.

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}'_{k|k-1} \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}'_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k)^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}'_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}'_{k|k-1}) \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}'_{k|k-1} \end{aligned} \quad (2.19)$$

“ Dalam skenario kecepatan tinggi, koreksi penuh terhadap vektor status, termasuk komponen kecepatan, sangat penting. Jika kamera berubah dengan lambat dibandingkan dengan kecepatan frame, koreksi pada persamaan 2.18 dapat diabaikan. Setelah mengompensasi gerakan kamera yang rigid dan dengan asumsi posisi objek hanya sedikit berubah dari satu frame ke frame berikutnya, pada aplikasi dengan frame rate tinggi, ketika deteksi hilang, prediksi lintasan dapat dilakukan menggunakan langkah prediksi KF, yang memungkinkan tampilan lintasan yang lebih kontinu dan MOTA yang lebih tinggi.

Visualisasi prediksi *bounding box* (BB) dari tracklet digunakan untuk asosiasi dengan BB deteksi baru berdasarkan kriteria IoU maksimum. (a.1) dan (b.1) memperlihatkan prediksi dari Kalman Filter (KF). (a.2) dan (b.2) memperlihatkan prediksi dari KF setelah kompensasi gerakan kamera. Pada gambar (b.1), diperlihatkan skenario di mana pengabaian gerakan kamera dapat menyebabkan IDSWs atau FN. Sebaliknya, pada gambar (b.2), prediksi telah sesuai dengan lokasi yang diinginkan dan asosiasi berhasil dilakukan. Gambar 2.9.2 dihasilkan dari sekuens MOT17 [11], yang mencakup pergerakan kamera akibat manuver kendaraan yang berbelok ke arah kanan.



Gambar 2.12: Kompensasi Gerakan Kamera

2.9.3 Fusi IoU - Re-ID

Fitur Re-ID diintegrasikan untuk memanfaatkan kemajuan dalam representasi visual objek. Fitur Re-ID diekstraksi menggunakan FastReID dengan backbone ResNeSt50 [12], [13], dan mekanisme exponential moving average (EMA) digunakan untuk memperbarui status appearance tracklet [14]. Pembaruan status appearance dilakukan dengan rumus:

$$e_i^k = \alpha e_i^{k-1} + (1 - \alpha) f_i^k \quad (2.20)$$

Di mana e_i^k adalah status appearance untuk tracklet ke- i pada frame k , f_i^k adalah embedding appearance deteksi saat ini, dan $\alpha = 0.9$ adalah momentum term.

Penggabungan antara informasi gerakan (IoU) dan appearance (cosine similarity) dilakukan dengan cara berikut. Pertama, kandidat dengan kesamaan cosinus rendah atau yang terlalu jauh berdasarkan skor IoU ditolak. Selanjutnya, nilai minimum pada setiap elemen matriks digunakan sebagai nilai akhir dari *cost matrix* C . Pipeline fusi IoU-ReID ini dapat diformulasikan sebagai berikut:

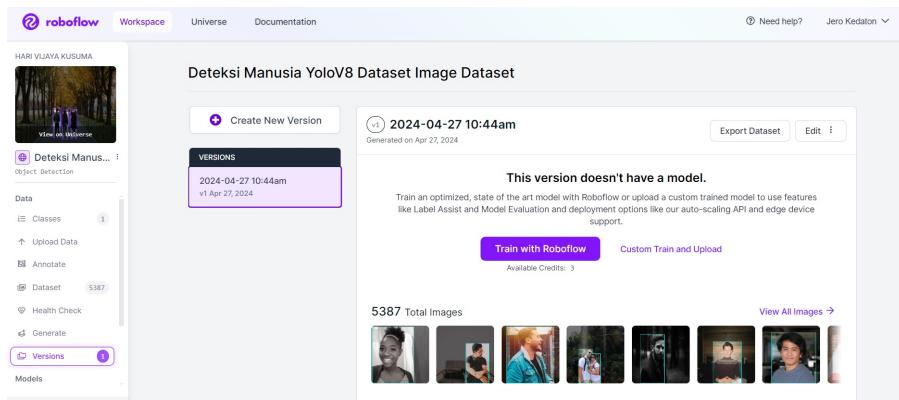
$$\hat{d}_{i,j}^{cos} = \begin{cases} 0.5 \cdot d_{i,j}^{cos}, (d_{i,j}^{cos} < \theta_{emb}) \wedge (d_{i,j}^{iou} < \theta_{iou}) \\ 1, \text{ otherwise} \end{cases} \quad C_{i,j} = \min\{d_{i,j}^{iou}, \hat{d}_{i,j}^{cos}\} \quad (2.21)$$

$$\hat{b} = \begin{cases} 0, & \Delta data \neq 0 \\ 1, & \neg(\hat{t} \in [0, t] \vee b) \\ b, & \text{otherwise} \end{cases} \quad (2.22)$$

Di mana $C_{i,j}$ merupakan elemen (i, j) dari *cost matrix* C . $d_{i,j}^{iou}$ adalah jarak IoU antara prediksi *bounding box* tracklet ke- i dan *bounding box* deteksi ke- j , yang merepresentasikan *cost* gerakan. $d_{i,j}^{cos}$ adalah jarak cosinus antara deskriptor penampilan rata-rata tracklet ke- i dan deskriptor deteksi baru ke- j . $\hat{d}_{i,j}^{cos}$ adalah *cost* penampilan baru yang digunakan. θ_{iou} adalah *threshold* kedekatan, yang ditetapkan sebesar 0.5, untuk menolak pasangan tracklet dan deteksi yang tidak mungkin. θ_{emb} adalah *threshold* penampilan, yang digunakan untuk memisahkan asosiasi positif dari keadaan penampilan tracklet dan vektor embedding deteksi dari asosiasi negatif. Permasalahan penugasan linear dari deteksi dengan *confidence* tinggi, yaitu langkah asosiasi pertama, diselesaikan menggunakan algoritma Hungarian [15] dan berdasarkan *cost matrix* C , yang dibentuk menggunakan Persamaan 2.21.

2.10 RoboFlow

RoboFlow adalah platform yang mendukung pengolahan data secara efisien dan efektif. Melalui berbagai fitur yang disediakan, mulai dari anotasi data hingga evaluasi model, *RoboFlow* memastikan bahwa setiap tahap dalam alur kerja pembelajaran mesin dapat dilakukan dengan lebih terstruktur dan mudah diintegrasikan dengan kerangka kerja yang ada. Peningkatan kinerja model dapat dikembangkan tanpa terbebani oleh kompleksitas teknis dalam penyiapan data.



Gambar 2.13: Interface RoboFlow

RoboFlow memfasilitasi proses penandaan gambar melalui alat bantu intuitif, sehingga percepatan dalam pembuatan label untuk *dataset* dapat tercapai. Gambar dalam *dataset* secara otomatis dimodifikasi melalui augmentasi data untuk menciptakan variasi guna meningkatkan *robustness* model yang dilatih. Dukungan untuk konversi antara berbagai format *dataset*

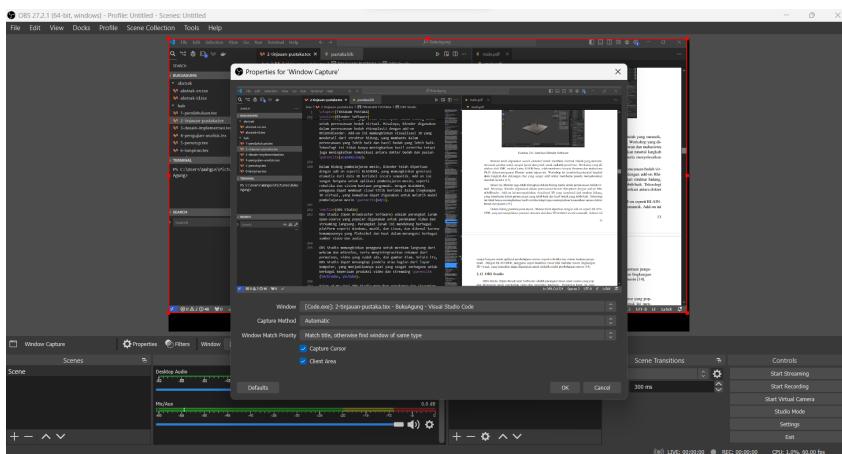
populer juga disediakan, memudahkan persiapan data untuk berbagai algoritma pembelajaran mesin. Selain itu, fungsi untuk membagi *dataset* menjadi set pelatihan, validasi, dan pengujian untuk validasi model.

RoboFlow juga menawarkan integrasi dengan banyak kerangka kerja pembelajaran mesin populer seperti *TensorFlow*, *PyTorch*, dan *YOLO*, yang mencakup:

- *Eksport Data*: *Dataset* dapat dengan mudah dieksport dalam format yang siap digunakan oleh kerangka kerja pembelajaran mesin.
- *Pelatihan Model*: Platform ini memungkinkan pelatihan model secara langsung menggunakan *dataset* yang telah disiapkan dan dioptimalkan.
- *Evaluasi Model*: Kinerja model diukur menggunakan metrik yang membantu dalam memahami dan menyesuaikan efektivitas model sesuai kebutuhan.

2.11 OBS Studio

OBS Studio adalah perangkat lunak *open-source* yang dirancang untuk streaming dan perekaman video. Dalam proyek ini, *OBS Studio* digunakan untuk mendokumentasikan dan menganalisis performa kursi roda otonom selama fase pengujian. Perangkat lunak ini memungkinkan pemilihan berbagai sumber video input, termasuk *webcam*, yang digunakan untuk merekam seluruh proses pengujian secara visual. Dengan *OBS Studio*, proses pengujian dari awal hingga akhir dapat direkam secara lengkap, memungkinkan pengumpulan data visual yang kaya dan mendetail.



Gambar 2.14: Interface OBS Studio

2.12 ESP32 Devkit V1

ESP32 Devkit V1 adalah sebuah mikrokontroler yang dilengkapi dengan kemampuan komunikasi nirkabel, termasuk *Wi-Fi* dan *Bluetooth*. Dalam proyek ini, *ESP32 Devkit V1* digunakan sebagai modul komunikasi utama untuk kursi roda otonom. Perangkat ini memungkinkan kursi roda untuk berinteraksi dengan perangkat eksternal seperti *smartphone* atau komputer melalui jaringan nirkabel.

Salah satu keunggulan utama dari *Devkit* ini adalah banyaknya pin yang dimiliki, yang memungkinkan mikrokontroler untuk diprogram dengan berbagai tugas. Pin-pin ini memberikan fleksibilitas yang tinggi dalam menghubungkan berbagai sensor dan aktuator yang diperlukan. Dengan memanfaatkan berbagai pin tersebut, sistem dapat menjalankan beberapa tugas secara simultan, seperti mengumpulkan data dari sensor, mengontrol motor, dan mengelola komunikasi nirkabel.

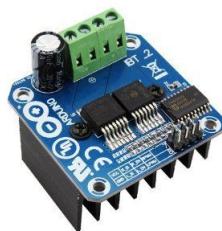


Gambar 2.15: Gambar ESP32 Devkit V1

Devkit ini kompatibel dengan *Arduino IDE* dengan bahasa pemrograman *C++*, yang merupakan platform yang umum digunakan. *Arduino IDE* menyediakan berbagai *library* dan dukungan yang memudahkan proses pemrograman *ESP32* untuk mengontrol berbagai perangkat keras yang terhubung. Dengan pemrograman dalam *C++*, kode yang efisien dan terstruktur dapat dihasilkan untuk menjalankan berbagai tugas kompleks, seperti pemrosesan data sensor dan pengendalian motor.

2.13 Motor Driver H-Bridge

Motor Driver H-Bridge adalah komponen elektronik yang krusial dalam pengendalian motor *DC* pada kursi roda. Komponen ini memungkinkan pengaturan arah dan kecepatan motor, yang esensial untuk navigasi kursi roda dalam berbagai kondisi. Dengan memanfaatkan *H-Bridge*, motor pada kursi roda dapat digerakkan maju, mundur, atau berbelok sesuai dengan kebutuhan.



Gambar 2.16: Gambar Motor Driver H-Bridge

Penggunaan *H-Bridge* dalam sistem otonom memungkinkan penyesuaian gerakan kursi roda secara *real-time* berdasarkan informasi yang diterima dari sensor. Hal ini memungkinkan kursi roda untuk mengikuti gerakan pengguna dengan presisi tinggi. Kemampuan untuk menavigasi secara mandiri menjadi sangat penting dalam memastikan bahwa kursi roda dapat beroperasi dengan aman dan efisien di berbagai lingkungan.

2.14 Kursi Roda Elektrik KY-123

Kursi roda elektrik *KY-123* digunakan sebagai platform utama dalam pengembangan kursi roda otonom dalam penelitian ini. Kursi roda ini dilengkapi dengan motor dan kontroler yang telah terintegrasi, yang memungkinkan modifikasi lebih lanjut untuk mendukung sistem otonom yang dirancang.

Dengan penambahan sensor, kamera, dan perangkat keras lainnya, kursi roda *KY-123* diharapkan dapat menjalankan fungsi otonom, termasuk kemampuan untuk mengikuti pergerakan pengguna secara mandiri. Pengintegrasian antara perangkat keras dan perangkat lunak pada *KY-123* menjadi elemen kunci dalam mencapai sistem yang handal dan efektif. Modifikasi ini bertujuan untuk meningkatkan mobilitas dan kemudahan penggunaan bagi individu dengan keterbatasan fisik, serta memastikan performa yang optimal dalam berbagai skenario penggunaan.



Gambar 2.17: Gambar Kursi Roda Elektrik KY-123

BAB III

DESAIN DAN IMPLEMENTASI

3.1 Deskripsi Sistem

Pada tahap ini, kursi roda otonom dikembangkan untuk dapat mengikuti manusia menggunakan algoritma deteksi berbasis YOLOv11. Sistem ini dirancang dengan tujuan utama meningkatkan mobilitas pengguna yang membutuhkan bantuan dalam bergerak. Sistem terdiri dari komponen perangkat keras dan perangkat lunak, yang terintegrasi untuk mencapai tujuan ini.

3.1.1 Komponen Sistem

Sistem ini terdiri dari:

1. **VS Code:** Digunakan sebagai lingkungan pengembangan untuk menjalankan kode terkait YOLOv11 dan analisis lainnya.
2. **Arduino IDE:** Mengembangkan dan mengunggah kode ke ESP32 yang mengendalikan motor dan sensor.
3. **Laptop:** Berfungsi sebagai pusat pemrosesan data yang lebih kompleks dan pengembangan perangkat lunak.
4. **Kamera (OV5640 5MP):** Menangkap gambar lingkungan secara real-time dan mendeteksi target (manusia) menggunakan YOLOv11.
5. **ESP32 Devkit V1:** Berfungsi sebagai pengontrol utama yang mengelola data dari kamera dan mengendalikan pergerakan kursi roda.
6. **2 Kontroller Motor:** Mengendalikan motor DC yang menggerakkan kursi roda.
7. **2 DC-DC Voltage Regulator:** Mengatur tegangan untuk komponen elektronik agar tetap stabil.
8. **2 Motor DC:** Menggerakkan kursi roda, dikontrol melalui driver motor yang menerima sinyal dari mikroprosesor.
9. **Baterai 24V:** Sebagai sumber daya utama untuk keseluruhan sistem, termasuk mikroprosesor, motor, dan perangkat lainnya.

3.1.2 Arsitektur Sistem

Arsitektur sistem dirancang agar kamera menangkap gambar secara terus-menerus, kemudian mengirimkan data ke mikroprosesor ESP32 untuk diproses. YOLOv11 digunakan untuk mendeteksi manusia dan memberikan informasi koordinat posisi target. Informasi ini kemudian digunakan oleh mikroprosesor untuk mengontrol motor dan mengarahkan kursi roda agar dapat mengikuti pergerakan target secara dinamis.

3.2 Perangkat Keras

Desain perangkat keras melibatkan integrasi antara beberapa komponen yang disebutkan sebelumnya. Kamera dipasang di bagian depan kursi roda untuk mendapatkan sudut pandang optimal terhadap target. Mikroprosesor ESP32 ditempatkan di bagian bawah kursi roda bersama dengan driver motor dan baterai untuk menjaga keseimbangan.

- **Unit Kontrol:** Unit Kontrol seperti komputer atau laptop digunakan sebagai pusat pemrosesan utama untuk menjalankan YOLOv11 dan mediapipe, menganalisis hasil deteksi dan menghasilkan keputusan berupa kode instruksi.
- **Kamera OV5640:** Kamera ini terhubung ke ESP32 untuk menangkap frame dengan menggunakan sensor 5MP.
- **ESP32:** Microcontroller ini menerima data dari kamera dan kemudian menjalankan algoritma untuk diproses pada komputer, lalu mengirim sinyal kontrol ke driver motor.
- **Driver Motor L298N:** Driver ini digunakan untuk mengontrol kecepatan dan arah putaran motor DC yang menggerakkan kursi roda.

3.2.1 Kamera

Kamera yang digunakan dalam sistem ini adalah OV5640, sebuah sensor kamera CMOS (Complementary Metal-Oxide-Semiconductor) dengan resolusi 5 megapiksel (MP). Sensor ini memiliki kemampuan menangkap gambar hingga resolusi maksimum 2592x1944 piksel, yang memberikan hasil citra dengan detail yang sangat baik. Sensor ini mendukung pengambilan gambar dan video secara real-time, dengan frame rate mencapai 30 frame per detik (fps) pada resolusi 1080p, sehingga ideal untuk aplikasi yang memerlukan pengolahan citra secara langsung.

Selain resolusinya yang tinggi, OV5640 juga dilengkapi dengan berbagai fitur canggih untuk pengolahan gambar otomatis. Berdasarkan datasheet, fitur seperti auto white balance, auto exposure, dan auto focus memungkinkan kamera ini beradaptasi secara otomatis terhadap perubahan kondisi pencahayaan dan jarak, sehingga tetap menghasilkan gambar berkualitas tinggi dalam berbagai situasi. OV5640 juga mendukung fitur face detection dan image scaling, yang sangat berguna dalam mendeteksi objek atau target manusia secara cepat dan akurat.

3.2.2 Unit Kontrol

Unit kontrol pada sistem ini, berperan sebagai pengolah data utama selama pengujian, dengan spesifikasi yang dirancang untuk menangani beban komputasi tinggi dan cocok untuk pengolahan data secara real-time.

Salah satu aspek penting dari unit kontrol dalam proyek ini adalah dukungan WiFi yang cepat dan stabil dalam komunikasi antar perangkat. Sistem bergantung pada data video yang dikirim oleh kamera OV5640 melalui ESP32 ke laptop untuk diolah, dan proses ini harus berlangsung tanpa hambatan. Konektivitas WiFi yang disediakan juga memungkinkan pengiriman data dalam waktu nyata dengan minim latensi, yang sangat penting untuk deteksi target secara cepat.

Kemampuan untuk menjaga kestabilan koneksi WiFi pada jarak yang lebih jauh dari router juga penting dalam pengujian di area yang luas. Seperti dengan menangani beberapa perangkat yang terhubung secara simultan, memungkinkan komunikasi yang efisien antara ESP32 tanpa penurunan performa. Teknologi ini sangat penting untuk memastikan bahwa sistem dapat terus memproses data video yang dikirimkan oleh kamera dan memberikan instruksi dengan respons yang cepat.

Data citra yang didapatkan menggunakan kamera OV5640, selanjutnya akan diproses melalui serangkaian pengolahan data citra menggunakan model deteksi yang telah dilatih sebelumnya. Model ini dilatih menggunakan arsitektur YOLOv11. Model yang digunakan diberi nama Best.pt, yang merupakan file hasil pelatihan yang telah di *training* untuk mendeteksi kelas

Manusia.

Agar sistem dapat diimplementasikan dengan baik, beberapa library atau pustaka perlu diinstal terlebih dahulu. Library tersebut mencakup OpenCV untuk pengolahan citra, Ultralytics untuk Yolo.

```
1 pip install OpenCV
2 pip install Ultralytics
3 ...dll
```

3.2.3 ESP32

ESP32 akan menerima data arah dari hasil klasifikasi deteksi manusia dalam bentuk karakter huruf seperti A, B, C, D, atau E. Berdasarkan penelitian sebelumnya, proses penerimaan data string oleh ESP32 dari perangkat yang terhubung sangat penting untuk kelancaran sistem.

Setelah data diterima dan disimpan, string tersebut diekstrak menjadi informasi tentang arah dan kecepatan. Tahap ekstraksi ini krusial untuk mengubah string menjadi format yang dapat digunakan oleh program dalam mengendalikan motor kursi roda. Data yang telah diekstrak kemudian ditampilkan untuk memastikan bahwa arah dan kecepatan yang diterima sesuai dengan yang diharapkan. Dengan informasi arah dan kecepatan yang sudah siap, ESP32 dapat mengirimkan instruksi ke motor kursi roda agar bergerak sesuai dengan data yang diterima. Proses ini akan terus berulang selama perangkat tetap terhubung dan data terus mengalir.

Untuk memberikan pemahaman yang lebih jelas mengenai instruksi yang diterima dari hasil klasifikasi deteksi manusia, berikut disajikan kode instruksi yang digunakan dalam program ini:

Tabel 3.1: Kode Instruksi dari Hasil Klasifikasi

Klasifikasi Pose	Kode Instruksi
Kiri	A
Maju	B
Stop	C
Mundur	D
Kanan	E

Kode instruksi tersebut digunakan untuk mengarahkan motor kursi roda sesuai dengan deteksi yang dilakukan oleh model klasifikasi YOLOv11. Misalnya, jika arah yang terdeteksi adalah "Kiri", maka kode instruksi 'A' akan dikirim untuk menggerakkan kursi roda ke kiri. Demikian pula, jika terdeteksi arah "Maju", instruksi 'B' akan dikirim untuk menggerakkan kursi roda maju. Kode 'C' digunakan untuk menghentikan kursi roda saat terdeteksi "Stop", 'D' untuk bergerak mundur ketika terdeteksi "Mundur", dan 'E' untuk bergerak ke kanan saat terdeteksi "Kanan".

Program ini memastikan bahwa ESP32 berfungsi secara efektif sebagai server yang menerima data dari NUC dan menggunakan untuk mengendalikan motor kursi roda. Dengan langkah-langkah yang telah dijelaskan, program ini dirancang untuk berjalan secara kontinu tanpa henti, menunggu dan memproses data yang dikirim oleh perangkat yang terhubung.

3.2.4 Driver Motor L298N

Driver motor yang digunakan dalam sistem ini adalah L298N, yang berperan penting dalam pengendalian motor DC pada kursi roda. Driver ini berfungsi sebagai perantara antara mikrokontroler dan motor, dengan menerima sinyal kontrol dari ESP32 dan meneruskannya ke motor. Penggunaan driver L298N memungkinkan sinyal digital diubah menjadi sinyal yang dapat diterima oleh motor, sehingga motor dapat beroperasi sesuai dengan instruksi yang diberikan.

Driver L298N mampu mengendalikan dua motor DC secara simultan dengan fitur pengaturan kecepatan dan arah putaran motor. Driver ini bekerja dengan prinsip rangkaian H-Bridge, yang memungkinkan perubahan arah arus untuk mengatur rotasi motor maju atau mundur. Selain itu, L298N dilengkapi dengan fitur pengaturan kecepatan melalui teknik Pulse Width Modulation (PWM), yang memberikan kendali yang lebih akurat terhadap kecepatan motor. Dengan fitur ini, kursi roda dapat bergerak maju, mundur, berbelok, atau berhenti sesuai instruksi yang diterima.

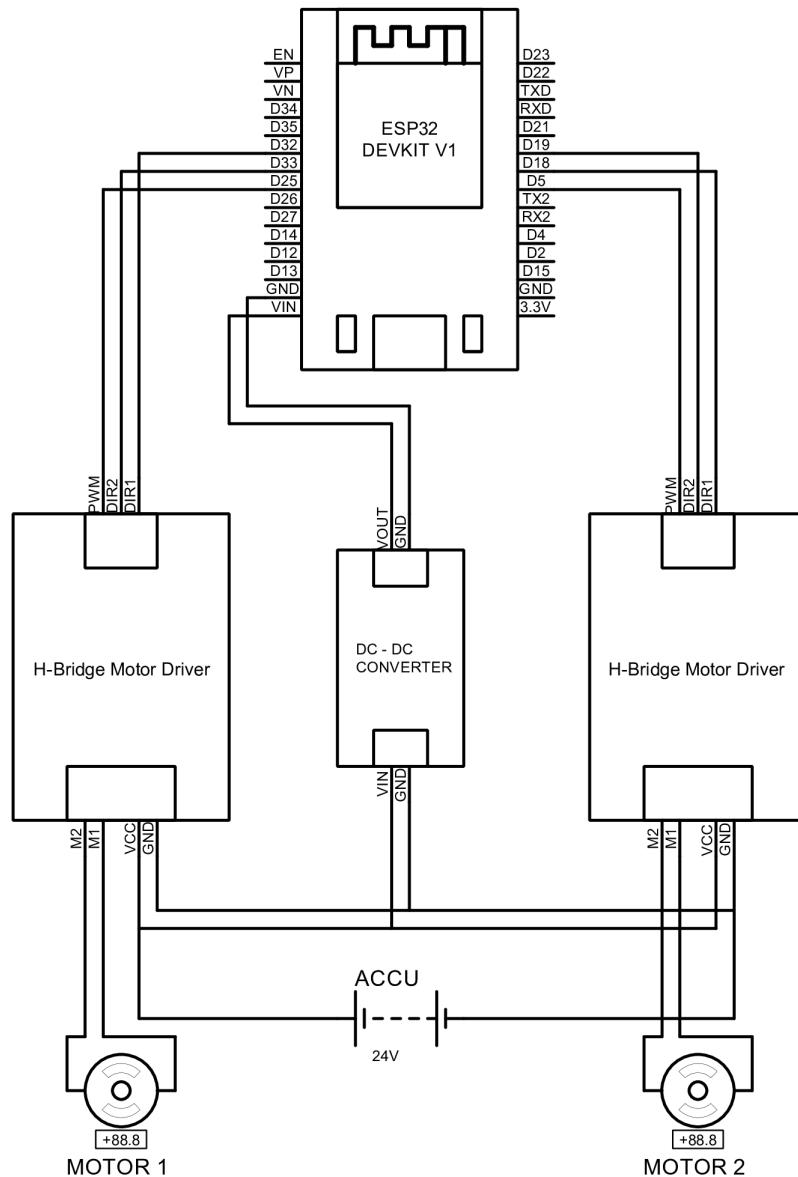
3.2.5 Skematik Alat

Dari penelitian yang telah dilakukan, telah dirancang sebuah sistem kontrol untuk motor kursi roda dengan skematik alat yang ditampilkan pada Gambar 3.1. ESP32 akan terhubung dengan dua buah H-Bridge Motor Driver dan sebuah DC-DC Converter. Setiap H-Bridge Motor Driver terhubung langsung ke motor roda kiri dan motor roda kanan untuk menggerakkan roda kursi roda secara efektif. Dalam skematik ini, ESP32 berfungsi sebagai otak dari sistem, mengirimkan sinyal kontrol ke motor driver berdasarkan data yang diterima melalui koneksi WiFi. [1]

Dalam penelitian ini, digunakan dua metode kontrol untuk menggerakkan kursi roda. Metode pertama adalah differential drive, ketika kursi roda berbelok ke kanan, roda kanan akan bergerak mundur dan berbelok ke kiri, sedangkan roda kiri bergerak maju dan berbelok ke kanan. Sebaliknya, ketika berbelok ke kiri, roda kiri akan bergerak mundur dan berbelok ke kanan, sedangkan roda kanan bergerak maju dan berbelok ke kiri. Metode kedua adalah pergerakan biasa, ketika kursi roda berbelok ke kanan, roda kanan akan diam sedangkan roda kiri bergerak maju dan berbelok ke kanan, dan sebaliknya untuk berbelok ke kiri. Saat bergerak maju atau mundur, kedua roda bergerak secara bersamaan ke arah yang diinginkan. Untuk penelitian kali ini, metode kedua digunakan untuk menggerakkan roda pada kursi roda.

Dengan demikian, program ini memastikan bahwa ESP32 dapat berfungsi secara efektif sebagai pengontrol motor kursi roda. Setiap langkah dalam flowchart dirancang untuk memastikan bahwa sistem dapat berjalan secara efisien dan responsif terhadap perintah yang diterima melalui jaringan WiFi. Sistem ini dirancang untuk beroperasi secara real-time, memberikan respon cepat terhadap perubahan perintah, dan mengendalikan kursi roda secara akurat berdasarkan data yang diterima.

Skematik yang ditampilkan memberikan gambaran yang jelas mengenai alur kerja dan koneksi perangkat keras yang digunakan dalam sistem kontrol ini. Dengan penjelasan yang mendetail, setiap aspek dari sistem ini dapat dipahami dengan baik, memastikan implementasi yang tepat dan operasi yang efisien dari kursi roda yang dikendalikan secara otonom. [1]



Gambar 3.1: Skematik kontrol motor kursi roda

3.3 Perangkat Lunak

Perangkat lunak yang digunakan dalam sistem ini terdiri dari beberapa modul, yang meliputi:

- **Algoritma Deteksi YOLOv11:** YOLOv11 digunakan untuk mendeteksi manusia dalam gambar yang ditangkap oleh kamera. Model ini dilatih untuk mengenali bentuk dan posisi manusia sehingga kursi roda dapat mengikuti target dengan tepat.
- **Estimasi Pose MediaPipe:** MediaPipe digunakan untuk mendeteksi landmark tubuh manusia setelah objek terkunci. Framework ini dipilih karena kemampuannya dalam mendeteksi keypoints pada tubuh manusia dengan akurasi tinggi, bahkan dalam kondisi pencahayaan yang beragam.
- **Komunikasi Data:** Sistem menggunakan protokol MQTT untuk mengirimkan data antara ESP32 yang terhubung ke kamera dan unit kontrol utama.
- **Kontrol Motor:** Modul kontrol motor diimplementasikan pada ESP32 untuk mengatur

arah dan kecepatan motor berdasarkan posisi target yang terdeteksi.

3.3.1 Dataset Citra

Dataset citra yang digunakan dalam penelitian ini terdiri dari kumpulan gambar yang diam-bil menggunakan kamera OV5640. Objek yang dideteksi pada gambar ini adalah manusia, yang difungsikan sebagai target untuk diikuti oleh kursi roda. Dataset mencakup berbagai pose dan posisi manusia, serta kondisi pencahayaan yang berbeda, agar model YOLOv11 dapat dilatih secara optimal dalam mengenali target dalam berbagai situasi. Gambar-gambar ini diperoleh dari setiap frame video yang diambil menggunakan kamera yang terhubung dengan komputer. Proses pendekripsi dilakukan secara real-time untuk memastikan sistem mampu mengidentifikasi target secara berkelanjutan.

3.3.2 Labeling

Dataset citra yang telah diperoleh kemudian melalui proses labeling dan augmentasi menggunakan alat seperti Roboflow, yang menyediakan berbagai fitur untuk mempermudah proses ini. Proses labeling ini melibatkan impor dataset, pemberian label pada setiap gambar, dan augmentasi data. Labeling dilakukan untuk memastikan bahwa setiap objek dalam gambar dikenali dengan benar, terutama manusia yang menjadi target. Penamaan class harus konsisten dengan objek yang akan dideteksi untuk memastikan model YOLOv11 dapat dilatih dengan baik.

Selain itu, Roboflow juga menyediakan fitur preprocessing dataset yang membantu men-standardkan format gambar, misalnya mengubah ukuran semua gambar menjadi seragam. Langkah ini penting untuk menjaga konsistensi dataset sebelum melatih model. Beberapa fitur yang tersedia antara lain Auto-Orient, Resize, Grayscale, Auto Adjust Contrast, Isolate Objects, Static Crop, Tile, Modify Classes, dan Filter Null. Preprocessing ini memastikan bahwa data yang digunakan memiliki kualitas yang seragam untuk meningkatkan performa model.

Fitur augmentasi data juga disediakan untuk menambah variasi pada dataset. Augmentasi bertujuan untuk meningkatkan keragaman dalam dataset, yang pada akhirnya membantu meningkatkan performa model dalam mendekripsi manusia. Proses augmentasi ini sangat penting dalam tugas akhir ini karena membutuhkan variasi yang luas dalam data citra manusia, yang memungkinkan sistem lebih robust dalam berbagai kondisi.

3.3.3 Klasifikasi YOLOv11

Dalam proses klasifikasi, setiap citra yang telah melalui proses labeling dikenali dengan menggunakan YOLOv11 yang telah dilatih untuk mendekripsi manusia pada citra. Model ini mampu mendekripsi keberadaan manusia secara akurat dan real-time, memungkinkan sistem untuk mengidentifikasi target dengan efisien. Setiap citra diproses oleh model YOLOv11, yang kemudian memberikan output berupa bounding box dan confidence score yang menunjukkan keberadaan manusia serta keyakinan model terhadap deteksi tersebut. Proses ini dilakukan secara real-time, memungkinkan sistem untuk memberikan umpan balik langsung ke kontrol kursi roda berdasarkan deteksi manusia dalam gambar.

Model YOLOv11 yang digunakan memiliki output kelas utama yaitu manusia. Model ini menghasilkan bounding box, yang menunjukkan area lokasi objek, dan confidence score untuk memberikan keyakinan deteksi. Algoritma YOLOv11 menggunakan Convolutional Neural Network (CNN) sebagai basisnya, yang berfungsi untuk mengekstraksi fitur dari gambar input, lalu memprediksi bounding box dan kelas objek langsung dari gambar tersebut. Proses ini diilustrasikan pada Gambar, yang menunjukkan blok diagram arsitektur model YOLOv11.

YOLOv11 terdiri dari beberapa lapisan, antara lain Conv2D (Convolutional 2D), Blok C2f, Blok SPPF (Spatial Pyramid Pooling Fast), lapisan Upsampling, dan lapisan Concatenate. Setiap lapisan ini memiliki peran tertentu dalam memproses gambar, mengekstraksi fitur, dan membuat prediksi yang tepat. Lapisan Conv2D digunakan untuk mengekstraksi fitur dari gambar input, sementara Blok C2f dan SPPF membantu dalam pemrosesan informasi dari berbagai skala untuk menangkap detail yang lebih baik. Lapisan Upsampling digunakan untuk meningkatkan resolusi fitur, sementara lapisan Concatenate menggabungkan berbagai informasi dari jalur yang berbeda untuk membuat representasi yang lebih kuat.

Gambar menunjukkan setiap jenis lapisan dalam YOLOv11 dengan warna yang berbeda, seperti lapisan Conv2D yang ditampilkan dalam warna biru, Blok C2f dengan warna kuning, lapisan SPPF dengan warna hijau muda, lapisan Upsampling dengan warna merah muda, dan lapisan Concatenate dengan warna ungu. Visualisasi ini membantu memahami bagaimana model YOLOv11 memproses gambar dan bagaimana setiap lapisan berkontribusi dalam menghasilkan bounding box dan prediksi kelas yang akurat. Selain itu, input dan output dari setiap lapisan dapat dilihat pada Gambar, yang menggambarkan detail dari tiap proses pemrosesan dalam model YOLOv11.

Lapisan deteksi pada YOLOv11 menghasilkan output yang mencakup bounding box dan confidence score untuk manusia yang terdeteksi, yang kemudian digunakan sebagai acuan dalam sistem untuk mengikuti target. Model ini dioptimalkan untuk dapat mengenali manusia dengan berbagai posisi dan kondisi pencahayaan, sehingga kursi roda dapat beradaptasi dengan pergerakan target dalam berbagai situasi secara efektif.

3.3.4 Estimasi Pose MediaPipe

Pada penelitian ini, pose manusia dideteksi menggunakan Python dengan library OpenCV dan MediaPipe. Proses dimulai setelah objek manusia terdeteksi dalam frame, di mana MediaPipe kemudian digunakan untuk mengidentifikasi landmark pada tubuh. MediaPipe dipilih karena kemampuannya mendeteksi titik kunci (keypoints) pada tubuh manusia dengan akurasi tinggi dalam berbagai kondisi pencahayaan dan posisi. Setelah landmark berhasil didektksi, titik-titik yang relevan akan digambarkan menggunakan garis untuk membentuk kerangka sesuai dengan pose tubuh.

Proses deteksi diawali dengan inisialisasi kamera yang menangkap frame secara real-time. Setelah frame diterima, dilakukan pra-pengolahan seperti konversi gambar ke skala abu-abu untuk mengurangi kompleksitas dan meningkatkan kecepatan deteksi. Gambar yang telah diprapengolah tersebut kemudian diproses oleh model MediaPipe untuk mendeteksi pose.

Dalam penelitian ini, beberapa landmark yang dipilih untuk analisis adalah titik-titik pada siku, lengan bawah, dan bahu kanan serta kiri. Pemilihan titik-titik ini didasarkan pada visibilitas dan konsistensi dalam proses deteksi. Tabel menampilkan nomor dan nama keypoint yang digunakan dalam estimasi pose:

Tabel 3.2: Tabel Keypoint yang digunakan

Nomor Keypoint	Nama Keypoint
11	RIGHT SHOULDER
12	LEFT SHOULDER
14	RIGHT ELBOW
16	RIGHT WRIST

Setelah landmark diperoleh, jarak antar titik pada piksel dihitung. Pengukuran ini dilakukan dengan menggunakan jarak Euclidean antara dua titik kunci, yang merupakan metode efisien untuk menghitung jarak dalam ruang dua dimensi. Nilai jarak ini kemudian digunakan sebagai acuan untuk pergerakan kursi roda mengikuti target di depan.

3.3.5 Pemrosesan Citra

Pemrosesan citra dilakukan untuk meningkatkan kualitas gambar yang diambil dari kamera dan mempermudah deteksi objek. Langkah-langkah pemrosesan citra meliputi konversi warna, penghilangan noise, dan pemfilteran tepi untuk menyorot bagian-bagian penting dari gambar.

3.3.5.1 Pembuatan Tracking ID untuk Mengunci Target

Sistem ini memanfaatkan Ultralytics YOLO untuk mendeteksi target manusia dalam setiap frame, lalu memberikan ID unik pada setiap objek yang terdeteksi. ID tersebut berfungsi untuk mengidentifikasi dan melacak individu yang sama pada frame berikutnya, memungkinkan untuk terus mengikuti target meskipun terjadi pergerakan dalam frame. Deteksi yang akurat dan pemberian ID ini sangat penting agar kursi roda dapat merespons secara tepat terhadap perubahan posisi target.

3.3.5.2 Keputusan untuk Bergerak Maju

Setelah target terdeteksi dan tracking ID dibuat, sistem akan menghitung jarak antara kursi roda dengan target. Jika jarak target berada dalam kisaran yang aman dan berada di tengah frame, sistem akan mengirimkan perintah untuk bergerak maju. Hal ini dilakukan dengan menggunakan algoritma yang menghitung jarak dari bounding box target dan memastikan bahwa target berada pada posisi yang tepat untuk diikuti.

3.3.5.3 Keputusan untuk Berbelok ke Kiri

Jika target bergerak ke arah kiri dan keluar dari batas tengah frame, kursi roda akan mengirimkan perintah untuk berbelok ke kiri. Sistem mendeteksi posisi target di bagian kiri frame dan memberikan instruksi ke motor untuk mengarahkan kursi roda ke kiri agar tetap dapat mengikuti pergerakan target secara optimal.

3.3.5.4 Keputusan untuk Berbelok ke Kanan

Sebaliknya, jika target bergerak ke arah kanan dan keluar dari batas tengah frame, sistem akan mengirimkan perintah untuk berbelok ke kanan. Posisi target yang berada di sisi kanan frame akan memicu motor untuk berbelok ke kanan, memastikan kursi roda dapat menyesuaikan posisinya dengan target yang bergerak.

3.3.5.5 Keputusan Jika Target Menghilang dari Frame

Apabila target menghilang dari frame, sistem akan mencari target berdasarkan posisi terakhir yang terdeteksi. Jika ada target baru yang terdeteksi di frame, sistem akan membuat tracking ID baru dan melanjutkan pelacakan terhadap target tersebut. Sistem akan menunggu hingga target muncul kembali dalam frame atau mendeteksi target baru untuk melanjutkan pelacakan. Pendekatan ini memastikan bahwa kursi roda bergerak sesuai arah terakhir ketika target tidak terlihat.

3.3.6 Komunikasi Data

Pada sistem ini, komunikasi data dilakukan menggunakan kombinasi WiFi dan ESP-NOW. WiFi digunakan untuk membangun Access Point pada ESP32 yang terhubung dengan kamera, memungkinkan streaming video real-time ke unit kontrol melalui jaringan lokal. Stream ini dapat diakses melalui server HTTP di port 81, di mana data video dari kamera diproses oleh unit kontrol untuk deteksi target.

Selain itu, ESP-NOW digunakan untuk pengiriman data secara langsung antara dua perangkat ESP32. Data yang diterima dari port 80 oleh ESP32 yang terhubung dengan kamera dikirimkan ke ESP32 lainnya (yang mengendalikan motor) melalui ESP-NOW. Protokol ini memungkinkan pertukaran data dengan latensi rendah tanpa memerlukan koneksi internet atau router eksternal. Dengan kombinasi ini, sistem dapat mempertahankan komunikasi yang cepat dan stabil, memastikan kursi roda dapat merespons pergerakan target secara real-time.

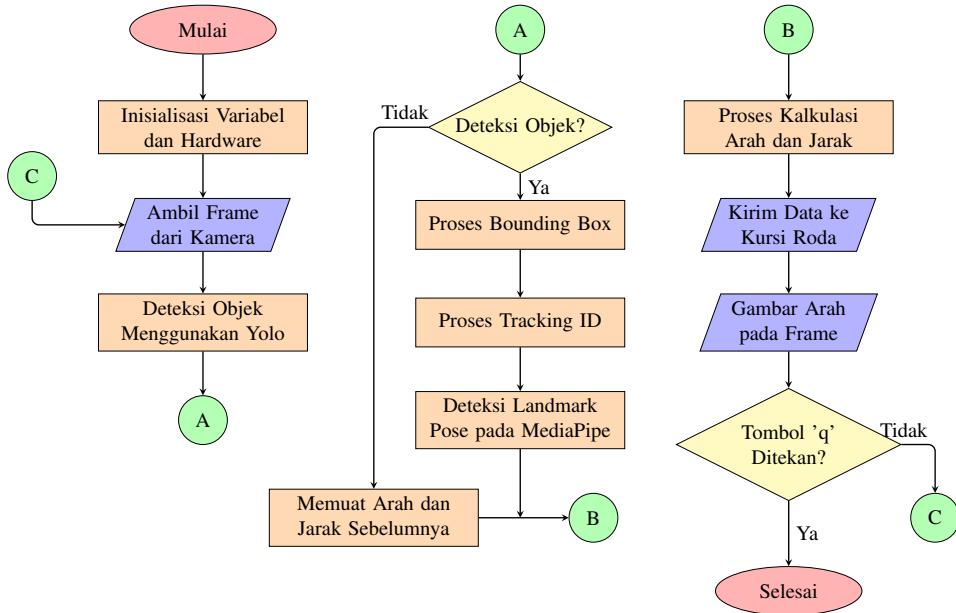
3.3.7 Kontrol Motor

Kontrol motor pada sistem kursi roda ini diimplementasikan menggunakan ESP32 yang bertindak sebagai pengendali utama untuk arah dan kecepatan motor. Setelah target manusia terdeteksi oleh kamera dan diproses oleh YOLOv11, informasi mengenai posisi target dikirim ke ESP32 untuk menentukan perintah pergerakan yang sesuai. Berdasarkan posisi target di dalam frame, ESP32 akan memberikan sinyal kepada driver motor untuk mengendalikan arah gerakan kursi roda, baik bergerak maju, berbelok ke kiri, berbelok ke kanan, atau berhenti.

Motor dikendalikan melalui sinyal PWM (Pulse Width Modulation) yang dihasilkan oleh ESP32, yang digunakan untuk mengatur kecepatan motor secara halus. Kombinasi sinyal arah dan PWM memungkinkan sistem untuk menggerakkan motor dengan responsif, baik untuk mengejar target, berbelok, atau berhenti secara tiba-tiba jika diperlukan. Selain itu, sistem juga menggunakan kontrol loop untuk terus memantau posisi target dan menyesuaikan pergerakan kursi roda secara real-time agar tetap dapat mengikuti target dengan akurat. Proses kontrol ini berlangsung terus-menerus untuk memastikan bahwa kursi roda selalu dapat beradaptasi dengan pergerakan target.

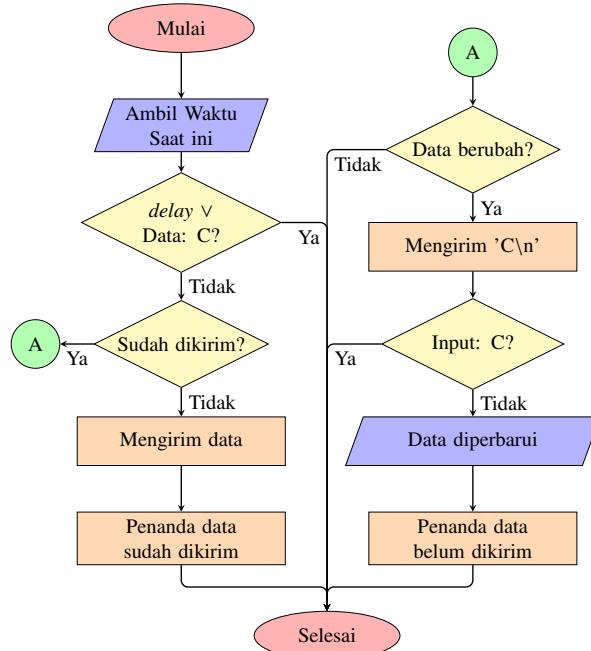
3.3.8 Kode Program

Kode program yang digunakan dalam penelitian ini dimulai dengan inisialisasi variabel dan perangkat keras yang diperlukan, termasuk mengaktifkan kamera untuk menangkap gambar secara real-time. Setelah gambar diperoleh, algoritma YOLOv11 digunakan untuk mendeteksi keberadaan manusia dalam frame tersebut. Jika manusia terdeteksi, program melanjutkan dengan mengidentifikasi bounding box dan memberi label pada objek, yang kemudian digunakan untuk melacak objek tersebut di frame berikutnya. Setelah itu, framework MediaPipe diimplementasikan untuk mendeteksi landmark tubuh, seperti bahu, siku, dan pergelangan tangan, yang memungkinkan sistem memperoleh pose manusia secara lebih rinci.



Gambar 3.2: Flowchart program python

Program kemudian menghitung arah dan jarak target, yang digunakan untuk menentukan instruksi pergerakan kursi roda. Instruksi tersebut dikirimkan ke ESP32, yang mengendalikan motor kursi roda untuk mengikuti manusia secara otomatis. Selain itu, arah pergerakan digambaran pada frame video yang ditampilkan sebagai bentuk visualisasi dari proses yang sedang berjalan. Program akan terus melakukan loop, menangkap gambar baru, memproses deteksi, dan mengirimkan perintah hingga pengguna menekan tombol 'q' untuk mengakhiri program.

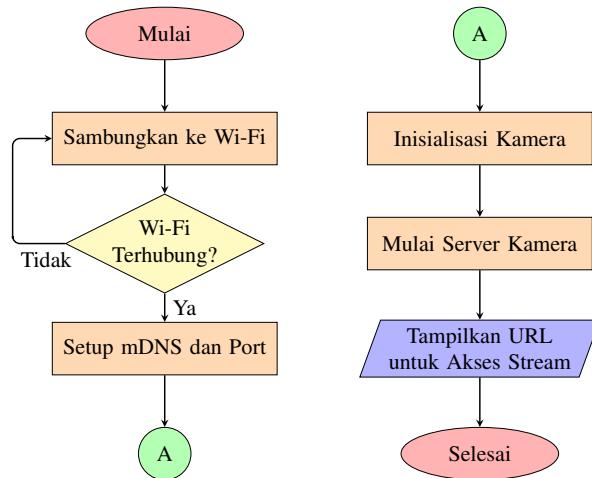


Gambar 3.3: Flowchart regulasi arah

Regulasi arah kursi roda dilakukan melalui pengiriman data arah ke socket yang berhubungan dengan ESP32.

gan dengan ESP32. Program mengecek apakah socket tersedia dan apakah data arah berubah, serta menetapkan penanda untuk menghindari pengiriman data yang berulang. Sebelum arah diubah, data 'C\n' dikirim terlebih dahulu untuk memastikan kursi roda berhenti dan stabil sebelum menerima instruksi arah baru. Hal ini penting untuk mencegah gerakan yang tidak diinginkan atau perubahan arah yang terlalu mendadak. Setiap kali arah berubah setelah berhenti, data baru akan dikirim ke socket untuk mengontrol motor kursi roda.

Untuk ESP32 CAM, sistem dimulai dengan inisialisasi kamera yang terhubung ke ESP32. Langkah pertama adalah pengaturan kamera agar siap menangkap gambar lingkungan secara real-time. Kamera yang digunakan adalah OV5640 dengan resolusi 5MP, yang memungkinkan pengambilan gambar berkualitas tinggi untuk deteksi yang lebih akurat.



Gambar 3.4: Flowchart ESP-CAM

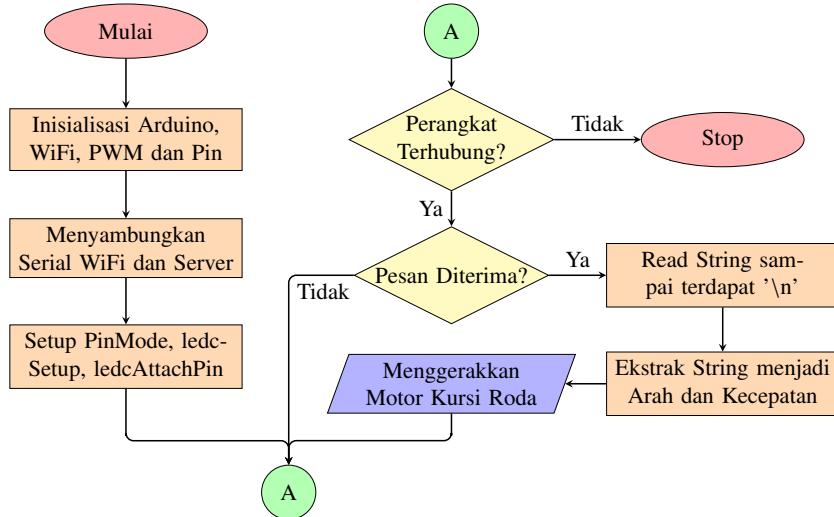
Sistem ini menerapkan protokol *mDNS (Multicast DNS)* untuk secara otomatis memberikan nama unik pada setiap kamera. Setiap kamera terhubung dengan nama berbeda, seperti *camera.local*, *camera2.local*, dan seterusnya. Alokasi port untuk streaming diatur secara otomatis, misalnya melalui port 81, 82, dan seterusnya, dengan akses langsung ke *endpoint /stream*. Pendekatan ini memudahkan penambahan kamera tanpa perlu konfigurasi manual. Dengan proses pendaftaran otomatis, sistem dapat menyesuaikan diri untuk berbagai kebutuhan, seperti pemantauan dari berbagai sudut atau penggunaan kamera cadangan sebagai redundansi. Setiap perangkat diakses melalui jaringan lokal menggunakan URL unik, yang menyederhanakan manajemen dan akses kamera dalam jaringan.

Selain itu, penggunaan server WiFi memungkinkan unit kontrol (seperti komputer atau laptop) mengakses data gambar secara langsung melalui jaringan lokal. Hal ini sangat berguna untuk menganalisis gambar secara lebih rinci dan mengambil keputusan kontrol yang lebih kompleks, seperti perubahan arah atau kecepatan kursi roda berdasarkan posisi target dalam frame. Dengan demikian, proses ini memberikan fleksibilitas yang tinggi dalam mengatur pergerakan kursi roda berdasarkan data yang diperoleh dari kamera.

Dengan memastikan koneksi Wi-Fi sebelum mengaktifkan kamera pada ESP32-CAM, konsumsi daya dapat dikelola secara optimal, sehingga panas yang dihasilkan tidak berlebih. Hal ini menjaga suhu modul kamera dan chip ESP32 dalam batas yang aman, yang berperan penting dalam mencegah overheating. Pengendalian suhu yang baik juga mendukung fungsi sensor kamera dan komponen elektronik lainnya, sehingga perangkat dapat beroperasi dengan

kinerja maksimal dalam jangka panjang.

Frame yang ditangkap oleh kamera dikirim melalui alamat lokal tertentu yang juga digunakan dalam kode program Python untuk melakukan streaming data. Data arah yang diterima dari client kemudian diteruskan ke ESP32 B melalui TCP/IP dengan unit kontrol untuk mengontrol kursi roda. Sistem ini dirancang untuk memastikan komunikasi yang cepat dan handal, sehingga kursi roda dapat merespons pergerakan target dengan tepat.



Gambar 3.5: Flowchart ESP Motor

Pada sisi ESP32 Motor, perangkat dimulai dengan inisialisasi berbagai komponen, seperti PWM, WiFi, dan pin untuk mengendalikan motor. Setelah terhubung ke WiFi, ESP32 B menunggu pesan dari server yang berfungsi sebagai pusat kendali. Jika pesan diterima, string yang mengandung informasi arah dan kecepatan akan diekstrak, kemudian diolah untuk menentukan pergerakan motor kursi roda. Sistem menggunakan data ini untuk mengatur arah dan kecepatan motor, yang memastikan kursi roda dapat mengikuti target secara akurat dan responsif. Proses ini dilakukan secara berulang untuk setiap update data yang diterima, memungkinkan kursi roda menyesuaikan gerakan dengan perubahan posisi target secara real-time.

BAB IV

PENGUJIAN DAN ANALISIS

Pada bab ini, akan dijelaskan mengenai hasil pengujian dan pembahasan dari penelitian yang telah diuraikan pada metodologi. Selain itu, akan dipaparkan juga mengenai skenario pengujian yang dilakukan untuk mengevaluasi performa sistem secara keseluruhan. Pengujian ini dilakukan dengan tujuan untuk memastikan bahwa sistem yang dirancang mampu berfungsi dengan baik dalam berbagai kondisi dan situasi yang mungkin dihadapi dalam penggunaannya.

4.1 Skenario Pengujian

Pengujian dilakukan untuk mengetahui performa model dalam melakukan deteksi dan mengikuti objek oleh kursi roda otonom. Skenario pengujian ini dirancang untuk mengukur berbagai aspek dari sistem, termasuk akurasi deteksi, kecepatan pemrosesan, respons sistem terhadap objek, dan tingkat keberhasilan mengidentifikasi tracking. Skenario pengujian yang akan dilakukan adalah sebagai berikut:

1. Hasil Pengujian Performa Model
2. Pengujian Berdasarkan FPS
3. Pengujian Berdasarkan Hasil Regulator Time
4. Pengujian Keberhasilan Tracking
5. Pengujian Tingkat Pencahayaan
6. Pengujian Kesesuaian Jarak Deteksi
7. Performa Pergerakan Mengikuti Objek
8. Performa Keberhasilan Mengikuti Objek

4.2 Hasil Pengujian Performa Menggunakan Confusion Matrix

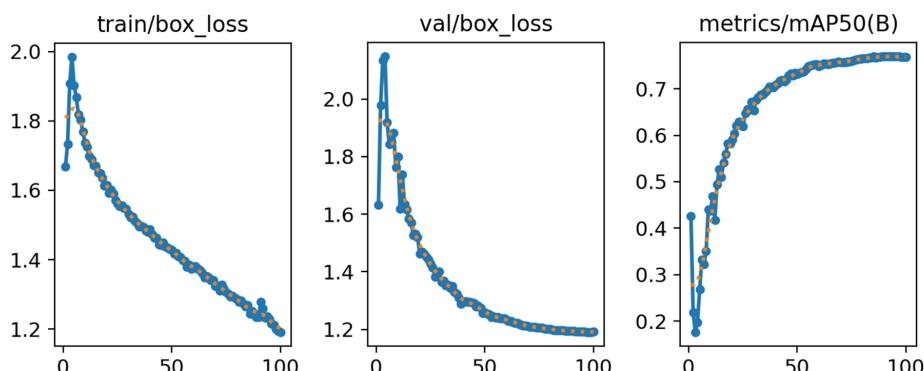
Sebanyak 15163 citra manusia dan 590 data validasi digunakan sebagai data latih awal. Melalui proses augmentasi, jumlah data latih meningkat menjadi 4.359, sementara data validasi bertambah menjadi 1.023. Setelah jumlah dataset ditentukan, langkah selanjutnya adalah membuat API key pada Roboflow yang kemudian diintegrasikan untuk proses pelatihan model.

Proses pelatihan model dimulai setelah seluruh dataset dimuat. Selama pelatihan, beberapa parameter digunakan dan hasilnya dibandingkan menggunakan nilai confusion matrix serta metrik akurasi deteksi seperti mAP score, precision, dan box loss. Layer input pertama dijelaskan sebagai tahap awal pelatihan model.

Input Layer : Citra Manusia	Input	[(16, 800, 800, 3)]
	Output	[(16, 800, 800, 3)]

Gambar 4.1: Input Layer Pelatihan Pertama

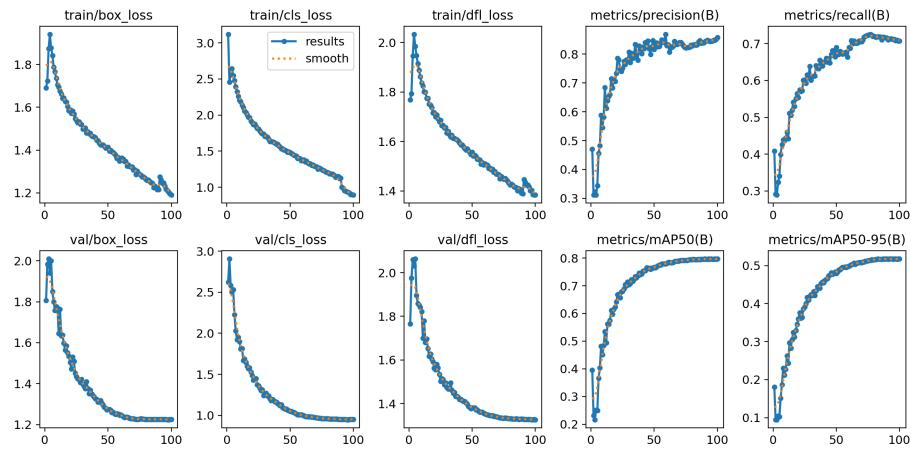
Pelatihan dilakukan selama 100 epoch pertama dengan ukuran batch sebesar 16 piksel, dan data diubah ukurannya menjadi 800 x 800 piksel selama preprocessing. Tujuan dari pelatihan ini adalah untuk mengevaluasi seberapa besar peningkatan performa model terlatih sebelumnya dalam mendekripsi manusia berdasarkan jumlah epoch yang dilakukan. Pada akhir 100 epoch, nilai box loss yang dihasilkan adalah 0,82978, yang menunjukkan kemampuan model untuk memprediksi bounding box dengan baik di sekitar objek. Penurunan nilai box loss selama pelatihan menunjukkan bahwa model berhasil belajar mengidentifikasi koordinat bounding box secara akurat.



Gambar 4.2: Loss Plot

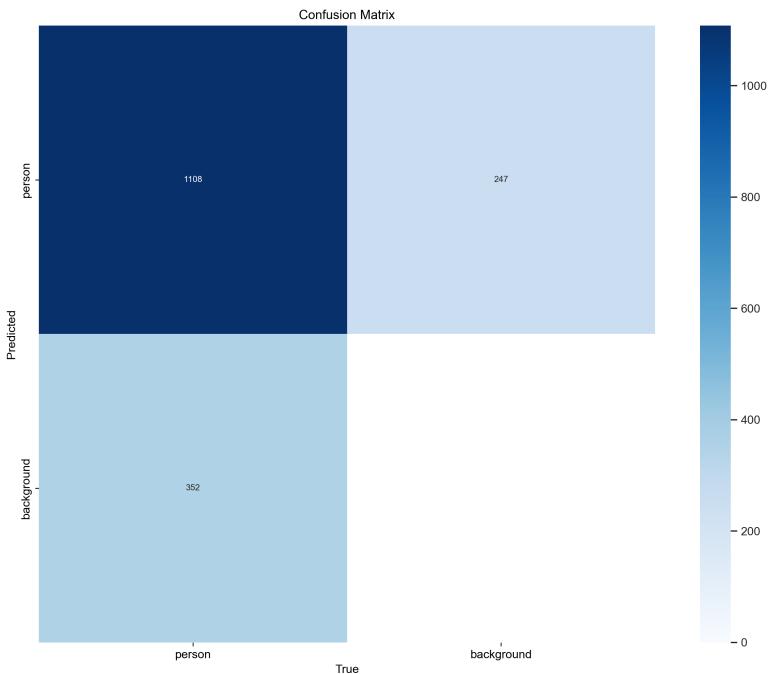
Selama validasi, nilai box loss tercatat sebesar 1,199, yang mengindikasikan kemampuan model dalam mengenali objek pada data uji. Penurunan box loss pada tahap validasi mencerminkan kemampuan model untuk mendekripsi objek secara general, tidak hanya pada data latih. Nilai mAP score divisualisasikan pada Gambar, dengan hasil skor mAP sebesar 81,85

Secara keseluruhan, hasil pelatihan model dirangkum pada Gambar. Nilai box loss pada pelatihan adalah 0,82978, nilai cls loss adalah 0,57615, dan dfl loss sebesar 1,1197. Metrik lainnya meliputi precision (0,84974), recall (0,72608), mAP50 (0,81855), dan mAP50-95 (0,53527). Selama validasi, nilai box loss tercatat sebesar 1,199, cls loss sebesar 0,86314, dan dfl loss sebesar 1,4956.



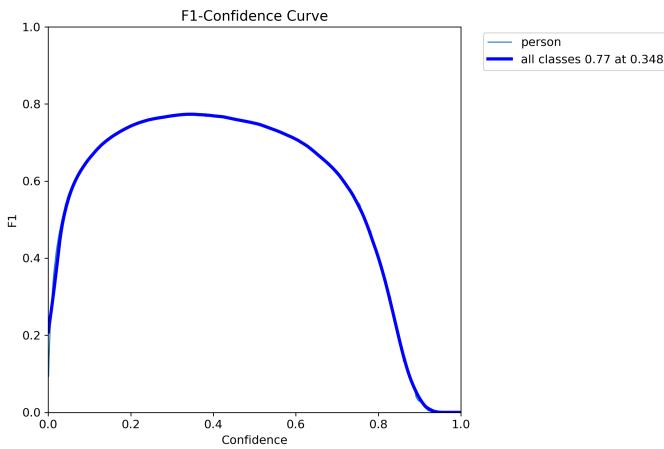
Gambar 4.3: Summary of Training Results

Visualisasi hasil model disajikan melalui confusion matrix yang menggambarkan kinerja deteksi secara rinci. Matrix ini menunjukkan 1.806 data sebagai True Positive (manusia terdeteksi dengan benar), 483 data sebagai False Positive (objek salah terdeteksi sebagai manusia), dan 475 data sebagai False Negative (manusia yang tidak terdeteksi).



Gambar 4.4: Confusion Matrix

Kurva F1-Confidence yang dihasilkan selama pelatihan menunjukkan hubungan antara confidence score dan F1-score. Model mencapai nilai F1-score tertinggi sebesar 0,78 pada confidence score 0,450, mengindikasikan keseimbangan optimal antara precision dan recall. Namun, kurva menunjukkan penurunan signifikan setelah confidence score mencapai sekitar 0,8, yang menandakan bahwa pada tingkat confidence yang sangat tinggi, model cenderung mengabaikan banyak true positives, sehingga menurunkan F1-score keseluruhan.



Gambar 4.5: F1-Confidence Curve

Inferensi terhadap data uji juga dilakukan menggunakan model yang telah dilatih. Hasilnya menunjukkan tingkat confidence yang tinggi pada deteksi objek manusia, sebagaimana divisualisasikan dalam gambar hasil pengujian



Gambar 4.6: Inference Results

4.3 Pengujian Berdasarkan FPS

Pengujian ini dilakukan dengan menggunakan laptop MSI GF63 yang memiliki spesifikasi seperti yang tertera pada Tabel 4.1. Laptop ini digunakan sebagai unit kontrol untuk menjalankan program dan mengirimkan instruksi ke unit eksekusi.

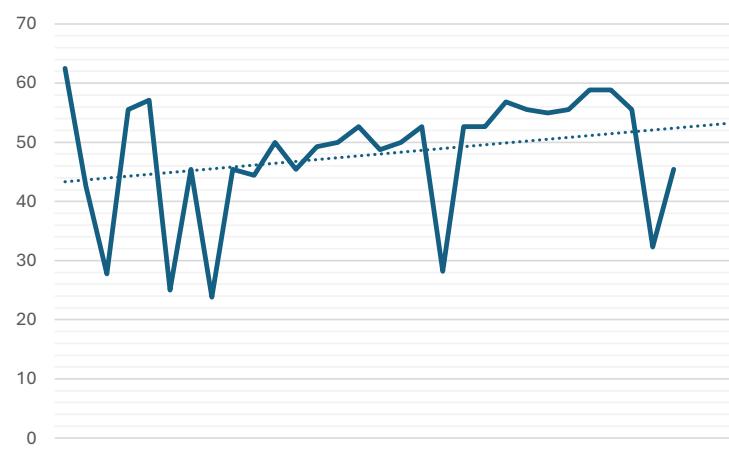
Tabel 4.1: Spesifikasi Laptop MSI GF63 untuk Unit Kontrol

Komponen	Spesifikasi
Model Laptop	MSI GF63
Prosesor	Intel Core i7-10750H, 6 core, 12 threads
GPU	NVIDIA RTX 3050
RAM	16GB DDR4, 3200 MHz

Output yang dihasilkan setiap kali sistem memproses frame diperoleh dari verbose yang mencatat detail penting, seperti resolusi frame, jumlah objek yang terdeteksi, dan waktu pemrosesan dalam milidetik untuk setiap frame. Dalam pengujian ini, hanya 30 data pertama yang diambil dari hasil verbose tersebut untuk dianalisis lebih lanjut guna mengevaluasi konsistensi dan performa sistem dalam mendeteksi objek secara real-time.

-
- ¹ 0: 480x640 1 person , 16.0ms
 - ² 0: 480x640 1 person , 23.5ms
 - ³ 0: 480x640 1 person , 36.0ms
 - ⁴ 0: 480x640 1 person , 18.0ms
 - ⁵ 0: 480x640 1 person , 17.5ms
-

Dari grafik 4.7, terlihat bahwa sistem mampu mencapai FPS tertinggi sebesar 62,5 dan FPS terendah sebesar 23,81. Fluktuasi ini disebabkan oleh variasi waktu pemrosesan (dalam milidetik) untuk setiap frame, yang dipengaruhi oleh kondisi lingkungan, kompleksitas objek yang terdeteksi, dan performa perangkat keras yang digunakan. Hasil ini menunjukkan bahwa sistem secara konsisten dapat memproses frame dengan rata-rata FPS di atas 40. Kestabilan nilai pada laptop menandakan bahwa performa sistem yang dijalankan pada laptop berjalan dengan sangat baik dan efisien.



Gambar 4.7: FPS Trend

4.4 Pengujian Berdasarkan Regulator Time

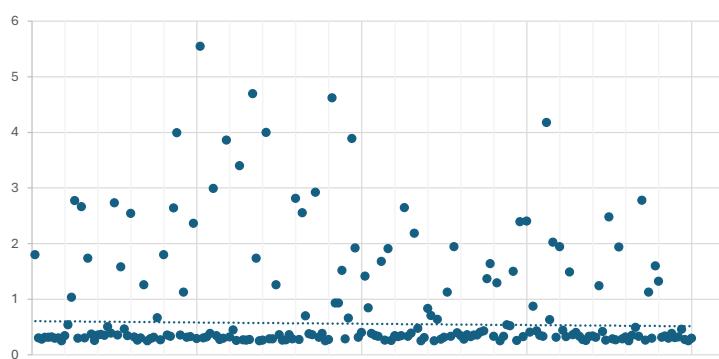
Bagian ini mengukur waktu yang dibutuhkan sistem untuk meregulasi setiap perubahan lingkungan atau perintah yang diterima. Output yang dihasilkan setiap kali pengiriman data berisi Arah dan time stamp pada Ardruino IDE dan Visual Studio code. Berikut merupakan contoh data mentah untuk kedua hasil output:

```
1 22:18:32.273 -> Stop
2 22:18:32.948 -> Arah : E
3 22:18:33.131 -> Kanan
4 22:18:33.759 -> Arah : C
5 22:18:33.936 -> Stop
6 22:18:34.748 -> Arah : B
7 22:18:35.121 -> Maju
8 22:18:35.439 -> Arah : C
9 22:18:35.815 -> Stop
10 22:18:36.410 -> Arah : E
11 22:18:36.616 -> Kanan
```

Output yang didapatkan pada ardruino berisikan Timestamp. Timestamp yang dihasilkan yaitu *Timestamp Receive ESP* dan *Timestamp Receive Motor*. Selain pada ardruino pada vscode juga didapatkan nilai time stamp *sent*, contoh outputnya dapat dilihat sebagai berikut.

```
1 BELOK KANAN ,0 .8 ,2024-05-08 22:18:14.263
2 MAJU ,0 .4 ,2024-05-08 22:18:18.762
3 BELOK KANAN ,0 .0 ,2024-05-08 22:18:28.614
4 MAJU ,0 .0 ,2024-05-08 22:18:30.940
5 BELOK KANAN ,1 .2 ,2024-05-08 22:18:32.899
6 MAJU ,0 .8 ,2024-05-08 22:18:34.728
7 BELOK KANAN ,0 .0 ,2024-05-08 22:18:36.392
```

Data yang dihasilkan selama pengujian kemudian diolah dan divisualisasikan dalam bentuk plot untuk mempermudah analisis. Plot ini menunjukkan distribusi waktu yang dibutuhkan oleh sistem untuk merespons setiap perintah yang diterima.



Gambar 4.8: Regulator Time Plot

Plot menunjukkan bahwa sekitar 66,17% dari seluruh pengujian mampu merespons dengan cepat, berada dalam waktu yang relatif singkat. Persentase ini menunjukkan performa sistem yang cukup baik dalam hal kecepatan respons terhadap perubahan lingkungan atau perintah. Namun, sisa 34% data yang berada di atas 0,5 detik menunjukkan adanya variasi atau potensi keterlambatan.

4.5 Pengujian Keberhasilan Tracking

Bagian ini mengevaluasi keberhasilan sistem dalam melakukan tracking terhadap objek target. Pengujian ini dilakukan dengan 15 kali percobaan untuk mengukur keandalan sistem dalam menjaga deteksi terhadap objek target secara konsisten. Tingkat keberhasilan diukur berdasarkan keberhasilan sistem dalam mengunci deteksi objek target (Locked Detection) atau kehilangan fokus pada objek target (Distracted Detection).

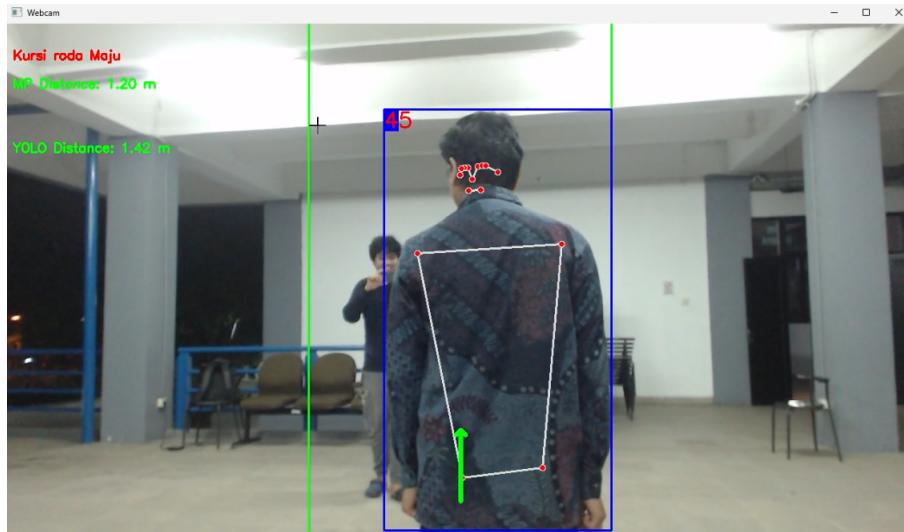
Tabel 4.2 menunjukkan data keberhasilan tracking selama pengujian. Data ini mencakup hasil tracking pada setiap percobaan, yang divisualisasikan dalam bentuk tabel dengan kode warna. Warna hijau menunjukkan bahwa sistem berhasil mengunci deteksi objek, sedangkan warna merah menunjukkan bahwa sistem gagal mengunci deteksi objek. Data ini memberikan gambaran kinerja sistem dalam mengikuti objek target, termasuk kecepatan respons, akurasi deteksi, dan ketepatan instruksi yang dikirimkan.

Tabel 4.2: Data Tracking Success

Trial	Result
1	Locked Detection
2	Locked Detection
3	Distracted Detection
4	Distracted Detection
5	Distracted Detection
6	Distracted Detection
7	Locked Detection
8	Locked Detection
9	Distracted Detection
10	Distracted Detection
11	Distracted Detection
12	Distracted Detection
13	Distracted Detection
14	Distracted Detection
15	Distracted Detection

Kemampuan BotSORT pada YOLO dalam melakukan tracking objek manusia terlihat dari 15 kali percobaan. Sistem berhasil mengunci deteksi objek sebanyak 4 kali, sedangkan 11 kali gagal mengunci deteksi objek secara akumulasi. Hasil ini menunjukkan bahwa sistem mampu melakukan tracking objek dengan tingkat keberhasilan sebesar 26,67%. Namun, terdapat 73,33% kegagalan dalam mengunci deteksi objek, yang menunjukkan adanya kekurangan dalam sistem.

Hasil ini menunjukkan bahwa sistem masih memiliki keterbatasan dalam menjaga stabilitas pelacakan terhadap objek target secara konsisten. Ketidakstabilan dalam pelacakan objek sering kali menyebabkan sistem kehilangan fokus pada target dan menghasilkan nomor ID baru untuk setiap deteksi ulang. Namun, meskipun terjadi kegagalan pelacakan, penggunaan ID terkecil memungkinkan identifikasi objek tetap dilakukan dengan tingkat akurasi yang memadai. Pendekatan ini membantu proses pengujian tetap berjalan dengan acuan identitas objek yang konsisten meskipun terdapat keterbatasan dalam menjaga kesinambungan pelacakan.



Gambar 4.9: Tracking Numeral

Gambar 4.9 menunjukkan peningkatan nomor track yang terus bertambah. Hal ini disebabkan oleh performa yang kurang optimal dari fungsi `model.track()` pada YOLO. Ketidakmampuan model untuk secara konsisten melacak objek menyebabkan nomor track baru dibuat setiap kali objek gagal terdeteksi, sehingga menghasilkan peningkatan jumlah track yang tidak diinginkan. Akan tetapi penguncian ini masih bisa digunakan walau tidak seperti satu id tiap object dan masih bisa digunakan dengan id terkecil sebagai penguncian. Hal ini menunjukkan bahwa meskipun sistem tidak selalu memberikan ID yang konsisten untuk setiap objek, ID terkecil yang diberikan dapat digunakan sebagai referensi untuk penguncian objek. Dengan demikian, sistem masih dapat melacak objek dengan tingkat keberhasilan tertentu meskipun terdapat beberapa kekurangan dalam konsistensi ID.

4.6 Pengujian Tingkat Pencahayaan

Pengujian dilakukan untuk mengukur kemampuan sistem dalam mendeteksi objek pada berbagai tingkat pencahayaan. Fokus pengujian ini adalah memastikan bahwa sistem dapat berfungsi dengan baik dalam kondisi pencahayaan yang berbeda.

Tabel 4.3: Lighting Level Evaluation

Time of Day	Lux			Conf.	
	Min	Avg	Max	YOLO	MP
Morning	100	300	500	0.73	0.71
Afternoon	2622	3753	5574	0.82	0.76
Evening	50	200	400	0.75	0.70
Night	26	215	538	0.60	0.56

Tabel 4.3 menunjukkan data tingkat pencahayaan yang diukur pada berbagai waktu sepanjang hari. Data ini mencakup tingkat pencahayaan minimum, rata-rata, dan maksimum yang diukur dalam lux, serta tingkat kepercayaan deteksi yang dihasilkan oleh YOLOv11 dan MediaPipe Pose. Data ini memberikan gambaran kinerja sistem dalam berbagai kondisi pencahayaan, termasuk akurasi deteksi, kecepatan respons, dan ketepatan instruksi yang dikirimkan.

4.7 Pengujian Kesesuaian Jarak Deteksi

Pengujian dilakukan untuk mengukur kemampuan sistem dalam mendeteksi objek pada berbagai jarak dan menganalisis performa sistem pada jarak-jarak tersebut. Fokus pengujian ini adalah memastikan bahwa ketika objek berada pada jarak yang sangat dekat (<1m), sistem mengirimkan kode instruksi untuk berhenti (diam) guna mencegah tabrakan.

Tabel 4.4: Data Jarak (<1m) untuk Diam

Waktu	Reg (s)	YOLO (m)	MP (m)	x (px)	Deteksi	Terkirim	Keterangan
12:55:12	0.8402	2.0631	0.9852	833	b'C\n'	b'C\n'	Stop
12:55:15	0.2585	1.1003	0.8942	242	b'C\n'	b'C\n'	Stop
12:55:24	2.1821	1.1773	0.9652	511	b'C\n'	b'C\n'	Stop
12:58:13	1.4998	1.4123	0.9574	531	b'C\n'	b'C\n'	Stop
12:58:14	0.2538	1.4538	0.9899	555	b'C\n'	b'C\n'	Stop
12:58:19	2.4033	1.1691	0.9213	852	b'C\n'	b'C\n'	Stop
12:58:19	0.4079	1.1839	0.9118	854	b'C\n'	b'C\n'	Stop
12:58:20	0.8708	1.0960	0.9548	528	b'C\n'	b'C\n'	Stop
12:58:21	0.4262	1.1003	0.9189	547	b'C\n'	b'C\n'	Stop
12:58:21	0.3497	1.1032	0.8985	554	b'C\n'	b'C\n'	Stop
12:58:21	0.3314	1.1032	0.8785	518	b'C\n'	b'C\n'	Stop
13:00:57	1.4848	1.1594	0.9943	208	b'C\n'	b'C\n'	Stop
13:01:15	1.5987	1.3643	0.9899	849	b'C\n'	b'C\n'	Stop
13:01:19	0.2719	1.0989	0.7700	856	b'C\n'	b'C\n'	Stop
13:01:19	0.2509	1.1452	0.5782	856	b'C\n'	b'C\n'	Stop
17:41:34	0.3036	1.4339	0.9447	647	b'C\n'	b'C\n'	Stop
17:41:35	0.3326	1.4291	0.8357	642	b'C\n'	b'C\n'	Stop
17:41:36	0.3588	1.4389	0.8222	672	b'C\n'	b'C\n'	Stop
17:41:45	0.3092	1.3913	0.7249	640	b'C\n'	b'C\n'	Stop
17:42:04	15.5192	1.4563	0.5955	213	b'C\n'	b'C\n'	Stop

Tabel 4.4 menunjukkan data jarak yang diukur ketika objek berada pada jarak kurang dari 1 meter. Data ini mencakup waktu deteksi, jarak deteksi dari YOLOv11 dan MediaPipe Pose, posisi objek dalam frame, hasil deteksi, instruksi yang dikirimkan, dan keterangan mengenai status objek. Data ini memberikan gambaran kinerja sistem dalam mengikuti objek yang berada pada jarak dekat, termasuk kecepatan respons, akurasi deteksi, dan ketepatan instruksi yang dikirimkan.

Pengujian jarak juga telah diverifikasi secara fisik dengan pengukuran dan perbandingan yang cermat sebelum menyajikan data. Hal ini memastikan akurasi dan keandalan pengukuran jarak yang digunakan dalam evaluasi. Proses verifikasi melibatkan penggunaan alat ukur yang terkalibrasi dan uji coba berulang untuk meminimalkan kesalahan.

4.8 Performa Pergerakan Mengikuti Objek

Analisis dilakukan untuk mengukur seberapa akurat sistem dapat mengikuti objek target, termasuk mempertahankan jarak yang tepat dan tidak kehilangan objek dalam berbagai kondisi.

4.8.1 Percobaan Dalam Frame

Bagian ini membahas kemampuan sistem dalam mengikuti objek yang berada di dalam frame kamera dengan fokus utama pada evaluasi kinerja regulator. Tujuannya adalah memastikan bahwa sistem dapat merespons hasil deteksi dengan akurat dan konsisten, sesuai dengan kondisi objek yang terdeteksi di dalam frame kamera. Analisis dilakukan menggunakan 100 data sampel untuk memberikan gambaran kinerja sistem secara menyeluruh.

Tabel 4.5: Data Status Frame (Dalam Frame)

Waktu	Reg (s)	YOLO (m)	MP (m)	x (px)	Deteksi	Terkirim	Keterangan
12:48:09	0.2960	2.4469	2.6214	765	b'E\n'	b'C\n'	Turn Right
12:48:09	0.2991	5.6875	4.1146	484	b'B\n'	b'E\n'	Forward
12:48:10	0.2516	2.2093	2.6063	595	b'B\n'	b'C\n'	Forward
12:48:14	2.7739	1.4513	1.7409	128	b'A\n'	b'A\n'	Turn Left
12:48:17	2.6621	1.7109	1.1479	447	b'B\n'	b'C\n'	Forward
12:50:45	0.5046	2.5900	1.4523	391	b'B\n'	b'B\n'	Forward
12:50:48	2.7333	1.1421	1.9733	164	b'A\n'	b'C\n'	Turn Left
12:50:51	0.4610	1.4219	1.0142	830	b'E\n'	b'E\n'	Turn Right
12:50:54	2.5388	1.4768	1.1208	501	b'B\n'	b'C\n'	Forward
12:50:57	0.2675	2.0939	1.4524	825	b'E\n'	b'E\n'	Turn Right
12:51:00	0.3556	1.7144	1.5984	492	b'B\n'	b'C\n'	Forward
12:51:00	0.3290	1.4978	1.4964	466	b'B\n'	b'B\n'	Forward
12:51:07	3.9922	1.7796	1.3830	896	b'E\n'	b'C\n'	Turn Right
12:51:08	0.3129	1.6801	1.7017	525	b'B\n'	b'C\n'	Forward
12:51:09	0.3226	1.6408	1.7256	463	b'B\n'	b'B\n'	Forward
12:51:11	2.3641	1.5332	1.1242	850	b'E\n'	b'C\n'	Turn Right
12:51:17	5.5467	2.2628	1.6525	563	b'B\n'	b'C\n'	Forward
12:51:17	0.3035	1.9853	1.6709	408	b'B\n'	b'B\n'	Forward
12:51:18	0.3176	1.6668	1.5184	189	b'A\n'	b'C\n'	Turn Left
12:51:18	0.3818	1.5277	1.4295	202	b'A\n'	b'A\n'	Turn Left
12:51:21	2.9916	2.0186	1.6728	440	b'B\n'	b'C\n'	Forward
12:51:21	0.3446	2.4830	1.6701	601	b'B\n'	b'B\n'	Forward
12:51:22	0.2721	2.6141	1.7926	769	b'E\n'	b'C\n'	Turn Right
12:51:22	0.2947	2.7330	1.5570	867	b'E\n'	b'E\n'	Turn Right
12:51:26	3.8579	2.5354	1.6134	553	b'B\n'	b'C\n'	Forward
12:51:26	0.4472	1.9174	1.2873	212	b'A\n'	b'C\n'	Turn Left
12:51:27	0.2536	1.6668	1.9112	142	b'A\n'	b'A\n'	Turn Left
12:51:30	3.3993	2.4258	1.4468	551	b'B\n'	b'C\n'	Forward
12:51:36	4.6957	1.7144	1.8941	478	b'B\n'	b'B\n'	Forward
12:51:37	1.7377	1.5194	1.8537	182	b'A\n'	b'C\n'	Turn Left

Lanjutan ke halaman berikutnya

Tabel 4.5: Data Status Frame (Dalam Frame)

Waktu	Reg (s)	YOLO (m)	MP (m)	x (px)	Deteksi	Terkirim	Keterangan
12:51:42	0.2820	1.4820	1.5157	424	b'B\n'	b'C\n'	Forward
12:51:42	0.2867	1.5473	1.6206	481	b'B\n'	b'B\n'	Forward
12:51:44	0.3591	2.1528	1.7878	245	b'A\n'	b'C\n'	Turn Left
12:51:44	0.2619	1.6003	1.9633	426	b'B\n'	b'A\n'	Forward
12:51:45	0.2692	1.5704	1.6022	449	b'B\n'	b'C\n'	Forward
12:51:45	0.3616	1.8259	1.5129	249	b'A\n'	b'B\n'	Turn Left
12:51:48	2.8133	1.7721	1.1400	766	b'E\n'	b'C\n'	Turn Right
12:51:48	0.2701	1.9485	1.8638	792	b'E\n'	b'E\n'	Turn Right
12:51:51	2.5508	2.2628	1.9379	569	b'B\n'	b'C\n'	Forward
12:51:52	0.6972	1.9174	1.6452	394	b'B\n'	b'B\n'	Forward
12:51:52	0.3755	1.5912	1.3909	185	b'A\n'	b'C\n'	Turn Left
12:51:52	0.3613	1.4716	1.2101	174	b'A\n'	b'A\n'	Turn Left
12:51:55	0.3125	1.9262	1.1721	564	b'B\n'	b'C\n'	Forward
12:51:56	0.3797	1.9667	1.6343	835	b'E\n'	b'B\n'	Turn Right
12:51:56	0.2506	1.9262	1.7147	880	b'E\n'	b'C\n'	Turn Right
12:51:56	0.2716	1.9759	1.6276	871	b'E\n'	b'E\n'	Turn Right
12:52:05	0.2830	1.9395	1.9075	502	b'B\n'	b'C\n'	Forward
12:55:09	1.9217	1.5004	1.8914	861	b'E\n'	b'E\n'	Turn Right
12:55:10	0.3128	1.4414	1.8698	858	b'E\n'	b'C\n'	Turn Right
12:55:10	0.3967	1.7356	1.0821	856	b'E\n'	b'E\n'	Turn Right
12:55:13	0.3498	2.0186	1.9290	520	b'B\n'	b'C\n'	Forward
12:55:14	0.3321	1.8664	1.4010	511	b'B\n'	b'B\n'	Forward
12:55:15	1.6785	1.1238	1.0672	170	b'A\n'	b'C\n'	Turn Left
12:55:18	0.2536	1.2489	1.0783	483	b'B\n'	b'C\n'	Forward
12:55:18	0.3224	1.2129	1.4666	168	b'A\n'	b'C\n'	Turn Left
12:55:19	0.3399	1.1515	1.3422	165	b'A\n'	b'A\n'	Turn Left
12:55:22	0.3309	1.5194	1.5842	504	b'B\n'	b'C\n'	Forward
12:55:22	0.3887	1.4690	1.6042	528	b'B\n'	b'B\n'	Forward
12:55:25	0.4715	1.1164	1.4924	170	b'A\n'	b'C\n'	Turn Left
12:55:29	1.1263	1.4029	1.2103	509	b'B\n'	b'C\n'	Forward
12:55:30	0.3296	1.4219	1.2387	508	b'B\n'	b'B\n'	Forward
12:58:02	1.9436	1.2696	1.0367	149	b'A\n'	b'C\n'	Turn Left
12:58:03	0.3955	1.0975	1.0832	145	b'A\n'	b'A\n'	Turn Left
12:58:03	0.3395	1.6971	1.2674	619	b'B\n'	b'C\n'	Forward
12:58:03	0.2777	1.6473	1.6845	616	b'B\n'	b'B\n'	Forward
12:58:04	0.3573	1.4291	1.9911	795	b'E\n'	b'C\n'	Turn Right
12:58:05	0.3560	1.6768	1.8145	795	b'E\n'	b'E\n'	Turn Right
12:58:05	0.3549	1.3890	1.6223	130	b'A\n'	b'C\n'	Turn Left
12:58:05	0.4054	1.3446	1.3906	174	b'A\n'	b'A\n'	Turn Left
12:58:09	1.6364	1.4665	1.1057	425	b'B\n'	b'C\n'	Forward
12:58:10	1.2931	1.4123	1.6479	320	b'A\n'	b'C\n'	Turn Left
12:58:11	0.3293	1.2094	1.7544	445	b'B\n'	b'C\n'	Forward

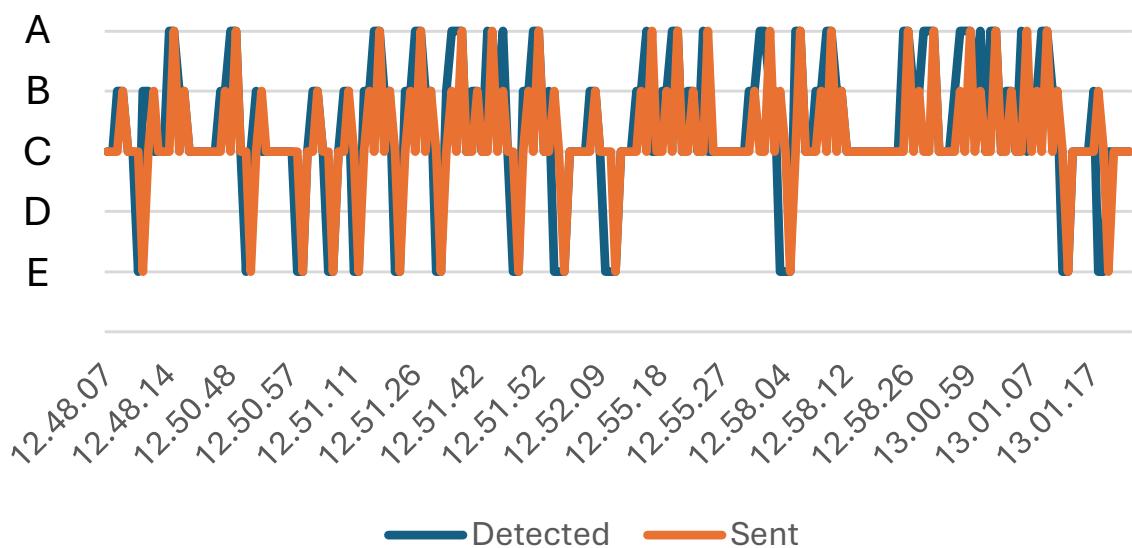
Lanjutan ke halaman berikutnya

Tabel 4.5: Data Status Frame (Dalam Frame)

Waktu	Reg (s)	YOLO (m)	MP (m)	x (px)	Deteksi	Terkirim	Keterangan
12:58:11	0.5372	1.2008	1.7284	503	b'B\n'	b'B\n'	Forward
12:58:25	4.1776	1.1047	1.2300	197	b'A\n'	b'C\n'	Turn Left
12:58:26	0.6310	1.1284	1.2368	195	b'A\n'	b'A\n'	Turn Left
12:58:28	2.0228	1.3319	1.2516	442	b'B\n'	b'C\n'	Forward
12:58:28	0.3156	1.3754	1.5678	432	b'B\n'	b'B\n'	Forward
13:00:55	1.9408	1.0975	1.0855	185	b'A\n'	b'C\n'	Turn Left
13:00:55	0.4380	1.0975	1.1142	177	b'A\n'	b'C\n'	Turn Left
13:00:56	0.3263	1.0960	1.1078	167	b'A\n'	b'A\n'	Turn Left
13:00:58	0.3332	1.3091	1.8538	444	b'B\n'	b'C\n'	Forward
13:00:58	0.2737	1.6903	1.0003	215	b'A\n'	b'B\n'	Turn Left
13:00:59	0.2535	1.6869	1.4439	228	b'A\n'	b'C\n'	Turn Left
13:00:59	0.3323	1.7536	1.9650	225	b'A\n'	b'A\n'	Turn Left
13:00:59	0.3355	1.4267	1.5125	456	b'B\n'	b'C\n'	Forward
13:01:00	0.3113	1.2734	1.1814	173	b'A\n'	b'B\n'	Turn Left
13:01:01	0.4180	1.6188	1.0805	170	b'A\n'	b'C\n'	Turn Left
13:01:02	0.2600	1.5882	1.5507	222	b'A\n'	b'A\n'	Turn Left
13:01:04	0.2826	1.3173	1.3641	483	b'B\n'	b'C\n'	Forward
13:01:05	0.2677	1.3256	1.3744	534	b'B\n'	b'B\n'	Forward
13:01:07	0.2501	1.1872	1.1742	513	b'B\n'	b'C\n'	Forward
13:01:08	0.3496	1.3298	1.2935	499	b'B\n'	b'B\n'	Forward
13:01:08	0.4923	1.6188	1.3519	211	b'A\n'	b'C\n'	Turn Left
13:01:09	0.3287	1.7039	1.7476	202	b'A\n'	b'A\n'	Turn Left
13:01:11	2.7773	1.5305	1.8361	517	b'B\n'	b'C\n'	Forward
13:01:13	1.1224	1.8459	1.5848	850	b'E\n'	b'C\n'	Turn Right
13:01:13	0.2936	1.5942	1.0293	832	b'E\n'	b'E\n'	Turn Right
13:01:17	0.2969	1.1193	1.0841	540	b'B\n'	b'C\n'	Forward
13:01:17	0.3835	1.1238	1.0723	862	b'E\n'	b'B\n'	Turn Right
13:01:18	0.3058	1.1253	1.0494	845	b'E\n'	b'C\n'	Turn Right

Tabel 4.5 menunjukkan data status dalam frame selama pengujian. Data ini mencakup waktu deteksi, jarak deteksi dari YOLOv11 dan MediaPipe Pose, posisi objek dalam frame, hasil deteksi, instruksi yang dikirimkan, dan keterangan mengenai status objek. Data ini memberikan gambaran kinerja sistem dalam mengikuti objek yang berada di dalam frame kamera, termasuk kecepatan respons, akurasi deteksi, dan ketepatan instruksi yang dikirimkan.

Data yang terdapat pada tabel telah diolah dan ditampilkan dalam bentuk plot untuk mempermudah visualisasi hubungan antara waktu dan instruksi yang dikirimkan oleh sistem. Dengan memanfaatkan plot ini, pola serta tren dalam data dapat diamati secara lebih jelas, sehingga memberikan gambaran yang lebih komprehensif mengenai kinerja sistem selama pengujian.



Gambar 4.10: Detection and Transmission Plots.

Visualisasi pada gambar 4.10 berperan penting dalam mendeteksi adanya anomali atau kesalahan yang mungkin terjadi selama proses pengujian. Data yang dipilih untuk divisualisasikan merupakan salah satu hasil terbaik dari serangkaian percobaan terpisah yang telah dilakukan. Hal ini menunjukkan bahwa data tersebut merepresentasikan kondisi sistem yang optimal dan relevan untuk dianalisis lebih lanjut.

Tabel 4.6: Summary of Data Transmission and Detection

	Percentage
Transmitted as C	55.4
Same Register	38.3
Different Register	6.3

Tabel 4.6 merangkum data transmisi dan deteksi. Persentase data yang dikirim sebagai C adalah 55.4%, yang menunjukkan bahwa sistem sering mengirim instruksi untuk berhenti. Persentase data di mana deteksi dan data yang dikirim sama adalah 38.3%, yang menunjukkan bahwa sistem berjalan sesuai dengan deteksi yang dilakukan. Persentase data di mana deteksi dan data yang dikirim berbeda adalah 6.3%, yang menunjukkan adanya galat atau kesalahan dalam sistem.

4.8.2 Percobaan Bergerak Maju

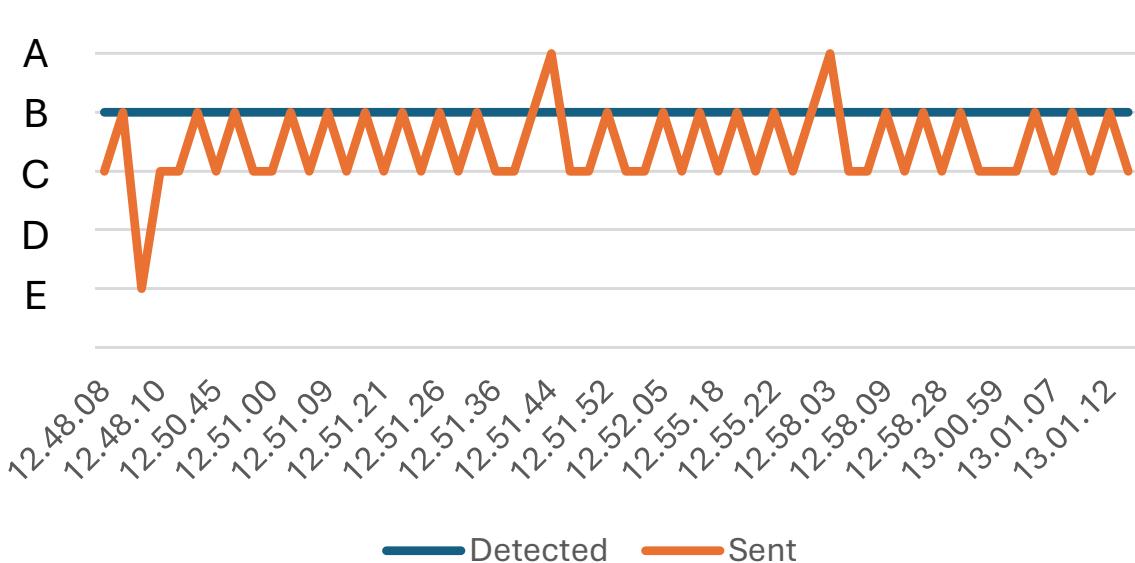
Bagian ini membahas kemampuan sistem dalam mengikuti objek yang bergerak maju dengan kecepatan konstan. Tantangan yang dihadapi adalah bagaimana sistem mempertahankan objek dalam frame sambil melakukan penyesuaian kecepatan yang diperlukan. Solusinya meliputi pemanfaatan fusi data dari YOLOv11 dan MediaPipe Pose, serta penerapan algoritma kontrol yang mampu melakukan koreksi arah tepat waktu, sehingga sistem dapat mengikuti objek yang bergerak maju dengan stabil dan akurat.

Tabel 4.7: Data Performa Bergerak Maju

Waktu	Reg (s)	YOLO (m)	MP (m)	x (px)	Deteksi	Terkirim	Keterangan
12.48.09	0.2991	5.6875	46.1146	484	b'B\n'	b'E\n'	Forward
12.48.10	0.2516	2.2093	5.6063	595	b'B\n'	b'C\n'	Forward
12.48.17	2.6621	1.7109	3.1479	447	b'B\n'	b'C\n'	Forward
12.50.45	0.5046	2.5900	5.4523	391	b'B\n'	b'C\n'	Forward
12.50.46	0.3870	2.5980	4.9960	414	b'B\n'	b'B\n'	Forward
12.50.54	2.5388	1.4768	3.1208	501	b'B\n'	b'C\n'	Forward
12.51.00	0.3556	1.7144	1.5984	492	b'B\n'	b'C\n'	Forward
12.51.00	0.3290	1.4978	1.4964	466	b'B\n'	b'B\n'	Forward
12.51.08	0.3129	1.6801	1.7017	525	b'B\n'	b'C\n'	Forward
12.51.09	0.3226	1.6408	1.7256	463	b'B\n'	b'B\n'	Forward
12.51.17	5.5467	2.2628	2.6525	563	b'B\n'	b'C\n'	Forward
12.51.17	0.3035	1.9853	2.1709	408	b'B\n'	b'B\n'	Forward
12.51.21	2.9916	2.0186	3.1728	440	b'B\n'	b'C\n'	Forward
12.51.21	0.3446	2.4830	3.1701	601	b'B\n'	b'B\n'	Forward
12.51.26	3.8579	2.5354	2.0134	553	b'B\n'	b'C\n'	Forward
12.51.30	3.3993	2.4258	2.4468	551	b'B\n'	b'C\n'	Forward
12.51.36	4.6957	1.7144	1.8941	478	b'B\n'	b'C\n'	Forward
12.51.42	0.2820	1.4820	2.5157	424	b'B\n'	b'C\n'	Forward
12.51.42	0.2867	1.5473	1.6206	481	b'B\n'	b'B\n'	Forward
12.51.44	0.2619	1.6003	1.9633	426	b'B\n'	b'A\n'	Forward
12.51.45	0.2692	1.5704	1.6022	449	b'B\n'	b'C\n'	Forward
12.51.51	2.5508	2.2628	1.8379	569	b'B\n'	b'C\n'	Forward
12.51.52	0.6972	1.9174	2.2452	394	b'B\n'	b'B\n'	Forward
12.51.55	0.3125	1.9262	2.1721	564	b'B\n'	b'C\n'	Forward
12.52.05	0.2830	1.9395	2.9075	502	b'B\n'	b'C\n'	Forward
12.55.13	0.3498	2.0186	1.9290	520	b'B\n'	b'C\n'	Forward
12.55.14	0.3321	1.8664	2.8010	511	b'B\n'	b'B\n'	Forward
12.55.18	0.2536	1.2489	1.0783	483	b'B\n'	b'C\n'	Forward
12.55.18	0.3409	1.2734	1.0403	522	b'B\n'	b'B\n'	Forward
12.55.22	0.3309	1.5194	3.5842	504	b'B\n'	b'C\n'	Forward

Tabel 4.7 menunjukkan data performa sistem saat bergerak maju. Data ini mencakup waktu deteksi, jarak deteksi dari YOLOv11 dan MediaPipe Pose, posisi objek dalam frame, hasil deteksi, instruksi yang dikirimkan, dan keterangan mengenai arah gerak. Data ini memberikan gambaran kinerja sistem dalam mengikuti objek yang bergerak maju, termasuk kecepatan re-

spons, akurasi deteksi, dan ketepatan instruksi yang dikirimkan.



Gambar 4.11: Straight Movement Plots

Visualisasi pada gambar 4.11 berperan penting dalam mendeteksi adanya anomali atau kesalahan yang mungkin terjadi selama proses pengujian. Data yang dipilih untuk divisualisasikan merupakan salah satu hasil terbaik dari serangkaian percobaan terpisah yang telah dilakukan. Hal ini menunjukkan bahwa data tersebut merepresentasikan kondisi sistem yang optimal dan relevan untuk dianalisis lebih lanjut.

Tabel 4.8: Summary of Data During Straight Movement

	Percentage
Transmitted as C	15.4
Same Register	11.0
Different Register	1.5

Tabel 4.8 merangkum data transmisi dan deteksi selama gerakan maju. Persentase data yang dikirim sebagai C adalah 15.4%, yang menunjukkan bahwa sistem sering mengirim instruksi untuk berhenti. Persentase data di mana deteksi dan data yang dikirim sama adalah 11.0%, yang menunjukkan bahwa sistem berjalan sesuai dengan deteksi yang dilakukan. Persentase data di mana deteksi dan data yang dikirim berbeda adalah 1.5%, yang menunjukkan adanya galat atau kesalahan dalam sistem.

4.8.3 Percobaan Belok Kiri

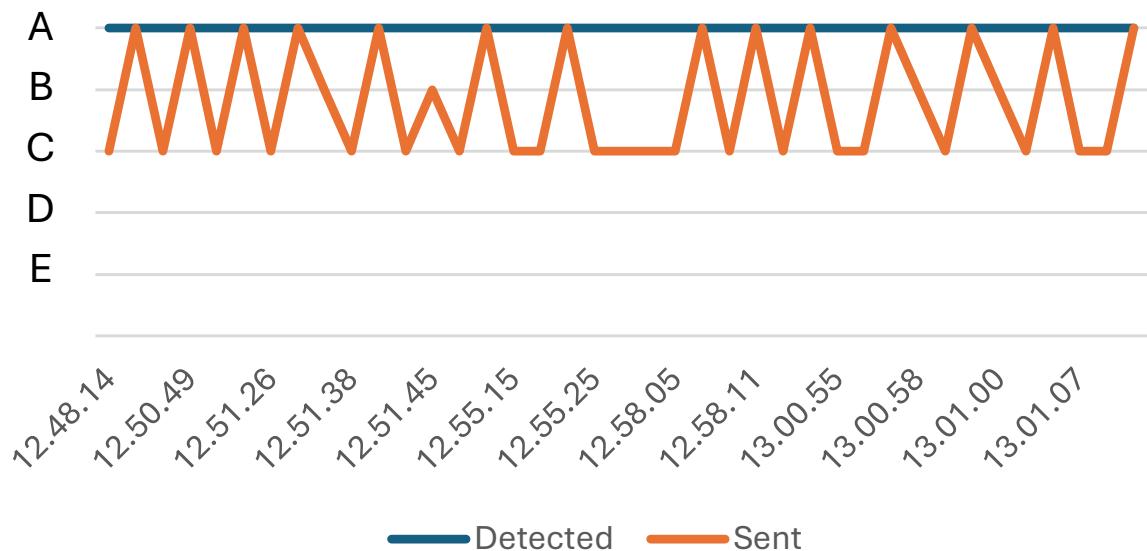
Bagian ini membahas kemampuan sistem dalam mengikuti objek saat berbelok ke kiri. Pada kondisi ini, tantangan yang muncul meliputi perubahan posisi relatif yang terjadi lebih cepat, potensi kehilangan objek dari frame, serta perlunya adaptasi kecepatan motor penggerak. Solusi yang diterapkan adalah penyesuaian parameter kontrol, algoritma pendekripsi pose yang lebih adaptif, serta strategi gerak yang mempertimbangkan arah belok objek sehingga sistem dapat mempertahankan jarak optimal dan ketepatan manuver ke kiri.

Tabel 4.9: Data Performa Belok (Kiri)

Waktu	Reg (s)	YOLO (m)	MP (m)	x (px)	Deteksi	Terkirim	Keterangan
12:48:14	2.7739	1.4513	1.1409	128	b'A\n'	b'C\n'	Turn Left
12:50:48	2.7333	1.1421	1.0733	164	b'A\n'	b'C\n'	Turn Left
12:51:18	0.3176	1.6668	1.1184	189	b'A\n'	b'C\n'	Turn Left
12:51:18	0.3818	1.5277	1.1295	202	b'A\n'	b'A\n'	Turn Left
12:51:26	0.4472	1.9174	1.4873	212	b'A\n'	b'C\n'	Turn Left
12:51:27	0.2536	1.6668	1.2112	142	b'A\n'	b'A\n'	Turn Left
12:51:37	1.7377	1.5194	1.4537	182	b'A\n'	b'B\n'	Turn Left
12:51:44	0.3591	2.1528	1.7878	245	b'A\n'	b'C\n'	Turn Left
12:51:45	0.3616	1.8259	1.5129	249	b'A\n'	b'B\n'	Turn Left
12:51:52	0.3755	1.5912	1.3909	185	b'A\n'	b'C\n'	Turn Left
12:51:52	0.3613	1.4716	1.2101	174	b'A\n'	b'A\n'	Turn Left
12:55:15	1.6785	1.1238	1.0672	170	b'A\n'	b'C\n'	Turn Left
12:55:18	0.3224	1.2129	1.4666	168	b'A\n'	b'C\n'	Turn Left
12:55:19	0.3399	1.1515	1.3422	165	b'A\n'	b'A\n'	Turn Left
12:55:25	0.4715	1.1164	1.4924	170	b'A\n'	b'C\n'	Turn Left
12:58:05	0.4054	1.3446	1.3906	174	b'A\n'	b'A\n'	Turn Left
12:58:10	1.2931	1.4123	1.6479	320	b'A\n'	b'C\n'	Turn Left
12:58:25	4.1776	1.1047	1.2300	197	b'A\n'	b'C\n'	Turn Left
12:58:26	0.6310	1.1284	1.2368	195	b'A\n'	b'A\n'	Turn Left
13:00:55	1.9408	1.0975	1.0855	185	b'A\n'	b'C\n'	Turn Left
13:00:55	0.4380	1.0975	1.1142	177	b'A\n'	b'C\n'	Turn Left
13:00:56	0.3263	1.0960	1.1078	167	b'A\n'	b'A\n'	Turn Left
13:00:58	0.2737	1.6903	1.0003	215	b'A\n'	b'B\n'	Turn Left
13:00:59	0.2535	1.6869	1.4439	228	b'A\n'	b'C\n'	Turn Left
13:00:59	0.3323	1.7536	1.3650	225	b'A\n'	b'A\n'	Turn Left
13:01:00	0.3113	1.2734	1.1814	173	b'A\n'	b'B\n'	Turn Left
13:01:01	0.4180	1.6188	1.0805	170	b'A\n'	b'C\n'	Turn Left
13:01:02	0.2600	1.5882	1.5507	222	b'A\n'	b'A\n'	Turn Left
13:01:08	0.4923	1.6188	1.3519	211	b'A\n'	b'C\n'	Turn Left
13:01:09	0.3287	1.7039	1.7476	202	b'A\n'	b'A\n'	Turn Left

Table 4.9 menunjukkan data performa sistem saat berbelok ke kiri. Data ini mencakup waktu deteksi, jarak deteksi dari YOLOv11 dan MediaPipe Pose, posisi objek dalam frame, hasil deteksi, instruksi yang dikirimkan, dan keterangan mengenai arah belok. Data ini memberikan gambaran kinerja sistem dalam mengikuti objek yang berbelok ke kiri, termasuk ke-

cepatan respons, akurasi deteksi, dan ketepatan instruksi yang dikirimkan.



Gambar 4.12: Left Turn Plots

Visualisasi pada gambar 4.12 berperan penting dalam mendeteksi adanya anomali atau kesalahan yang mungkin terjadi selama proses pengujian. Data yang dipilih untuk divisualisasikan merupakan salah satu hasil terbaik dari serangkaian percobaan terpisah yang telah dilakukan. Hal ini menunjukkan bahwa data tersebut merepresentasikan kondisi sistem yang optimal dan relevan untuk dianalisis lebih lanjut.

Tabel 4.10: Summary of Data During Left Turn

	Percentage
Transmitted as C	10.5
Same Register	6.7
Different Register	2.0

Tabel 4.10 merangkum data transmisi dan deteksi selama belok kiri. Persentase data yang dikirim sebagai C adalah 10.5%, yang menunjukkan bahwa sistem sering mengirim instruksi untuk berhenti. Persentase data di mana deteksi dan data yang dikirim sama adalah 6.7%, yang menunjukkan bahwa sistem berjalan sesuai dengan deteksi yang dilakukan. Persentase data di mana deteksi dan data yang dikirim berbeda adalah 2.0%, yang menunjukkan adanya galat atau kesalahan dalam sistem.

4.8.4 Percobaan Belok Kanan

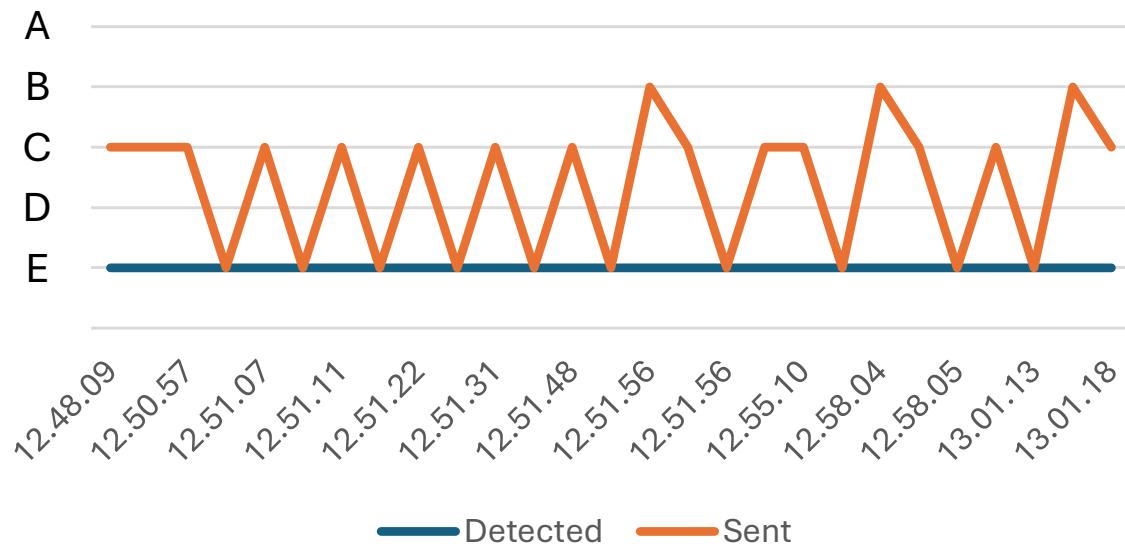
Bagian ini membahas kemampuan sistem dalam mengikuti objek saat berbelok ke kanan, yang pada dasarnya serupa dengan kondisi belok kanan. Tantangan yang dihadapi adalah bagaimana sistem mempertahankan objek dalam frame sambil melakukan penyesuaian sudut belok yang diperlukan. Solusinya meliputi pemanfaatan fusi data dari YOLOv11 dan MediaPipe Pose, serta penerapan algoritma kontrol yang mampu melakukan koreksi arah tepat waktu, sehingga sistem dapat mengikuti objek yang berbelok ke kanan dengan stabil dan akurat.

Tabel 4.11: Data Performa Belok (Kanan)

Waktu	Reg (s)	YOLO (m)	MP (m)	x (px)	Deteksi	Terkirim	Keterangan
12:48:09	0.2960	2.4469	1.6214	765	b'E\n'	b'C\n'	Turn Right
12:50:51	0.4610	1.4219	1.0142	830	b'E\n'	b'C\n'	Turn Right
12:50:56	0.6617	2.0234	1.6721	801	b'E\n'	b'C\n'	Turn Right
12:50:07	0.3573	1.4291	1.4911	795	b'E\n'	b'B\n'	Turn Right
12:50:07	0.3236	1.4588	1.7653	820	b'E\n'	b'C\n'	Turn Right
12:50:57	0.2675	2.0939	1.4524	825	b'E\n'	b'E\n'	Turn Right
12:51:07	3.9922	1.7796	1.3830	896	b'E\n'	b'C\n'	Turn Right
12:51:07	0.3518	2.0345	1.5442	883	b'E\n'	b'E\n'	Turn Right
12:51:11	2.3641	1.5332	1.5242	850	b'E\n'	b'C\n'	Turn Right
12:51:11	0.2897	2.0453	1.7421	792	b'E\n'	b'E\n'	Turn Right
12:51:22	0.2721	2.6141	1.7926	769	b'E\n'	b'C\n'	Turn Right
12:51:22	0.2947	2.7330	1.5570	867	b'E\n'	b'E\n'	Turn Right
12:51:31	0.2615	2.5602	1.3594	879	b'E\n'	b'C\n'	Turn Right
12:51:31	0.2711	2.2042	1.3829	827	b'E\n'	b'E\n'	Turn Right
12:51:41	2.8133	1.7721	1.4401	766	b'E\n'	b'C\n'	Turn Right
12:51:47	0.3573	1.4291	1.4911	795	b'E\n'	b'B\n'	Turn Right
12:51:48	0.3236	1.4588	1.7653	820	b'E\n'	b'C\n'	Turn Right
12:51:48	0.2701	1.9485	1.6638	792	b'E\n'	b'E\n'	Turn Right
12:51:56	0.2506	1.9262	1.7147	880	b'E\n'	b'C\n'	Turn Right
12:51:56	0.2716	1.9759	1.6276	871	b'E\n'	b'E\n'	Turn Right
12:55:09	1.9217	1.5004	1.8914	861	b'E\n'	b'C\n'	Turn Right
12:55:10	0.3128	2.1414	1.8698	858	b'E\n'	b'C\n'	Turn Right
12:55:10	0.3967	2.7356	2.0821	856	b'E\n'	b'E\n'	Turn Right
12:58:04	0.3573	1.4291	1.4911	795	b'E\n'	b'B\n'	Turn Right
12:58:04	0.3236	1.4588	1.7653	820	b'E\n'	b'C\n'	Turn Right
12:58:05	0.3560	1.6768	1.8145	795	b'E\n'	b'E\n'	Turn Right
13:01:13	1.1224	1.8459	1.5848	850	b'E\n'	b'C\n'	Turn Right
13:01:13	0.2936	1.5942	1.0293	832	b'E\n'	b'E\n'	Turn Right
13:01:17	0.3835	1.1238	1.0723	862	b'E\n'	b'B\n'	Turn Right
13:01:18	0.3058	1.1253	1.0494	845	b'E\n'	b'C\n'	Turn Right

Tabel 4.11 menunjukkan data performa sistem saat berbelok ke kanan. Data ini mencakup waktu deteksi, jarak deteksi dari YOLOv11 dan MediaPipe Pose, posisi objek dalam frame, hasil deteksi, instruksi yang dikirimkan, dan keterangan mengenai arah belok. Data ini mem-

berikan gambaran kinerja sistem dalam mengikuti objek yang berbelok ke kanan, termasuk kecepatan respons, akurasi deteksi, dan ketepatan instruksi yang dikirimkan.



Gambar 4.13: Right Turn Plots

Visualisasi pada gambar 4.13 berperan penting dalam mendeteksi adanya anomali atau kesalahan yang mungkin terjadi selama proses pengujian. Data yang dipilih untuk divisualisasikan merupakan salah satu hasil terbaik dari serangkaian percobaan terpisah yang telah dilakukan. Hal ini menunjukkan bahwa data tersebut merepresentasikan kondisi sistem yang optimal dan relevan untuk dianalisis lebih lanjut.

Tabel 4.12: Summary of Data During Right Turn

	Percentage
Transmitted as C	6.7
Same Register	5.0
Different Register	1.5

Tabel 4.12 merangkum data transmisi dan deteksi selama belok kanan. Persentase data yang dikirim sebagai C adalah 6.7%, yang menunjukkan bahwa sistem sering mengirim instruksi untuk berhenti. Persentase data di mana deteksi dan data yang dikirim sama adalah 5.0%, yang menunjukkan bahwa sistem berjalan sesuai dengan deteksi yang dilakukan. Persentase data di mana deteksi dan data yang dikirim berbeda adalah 1.5%, yang menunjukkan adanya galat atau kesalahan dalam sistem.

4.8.5 Percobaan Luar Frame

Bagian ini membahas kemampuan sistem dalam menghadapi kondisi ketika objek keluar dari bidang pandang kamera (luar frame). Tantangan yang muncul adalah hilangnya data visual tentang posisi dan gerakan objek, sehingga sistem harus mampu memprediksi posisi berikutnya atau melakukan penyesuaian strategi pencarian dengan mengintegrasikan algoritma prediksi lintasan serta inisialisasi ulang posisi yang membantu sistem untuk kembali melacak objek setelah objek kembali ke dalam frame.

Tabel 4.13: Data Status Frame (Luar Frame)

Waktu	Reg (s)	YOLO (m)	MP (m)	x (px)	Deteksi	Terkirim	Keterangan
12:48:08	0.2758	0.0	0.0	0	b'B\n'	b'B\n'	Forward
12:48:08	0.3151	0.0	0.0	0	b'B\n'	b'B\n'	Forward
12:48:15	0.2931	0.0	0.0	0	b'A\n'	b'A\n'	Turn Left
12:48:18	0.3001	0.0	0.0	0	b'B\n'	b'B\n'	Forward
12:50:43	0.7366	0.0	0.0	0	b'C\n'	b'C\n'	Stop
12:50:43	0.3732	0.0	0.0	0	b'C\n'	b'C\n'	Stop
12:50:43	0.2571	0.0	0.0	0	b'C\n'	b'C\n'	Stop
12:50:44	0.2395	0.0	0.0	0	b'E\n'	b'E\n'	Turn Right
12:50:44	0.3522	0.0	0.0	0	b'C\n'	b'C\n'	Stop
12:50:44	0.3647	0.0	0.0	0	b'C\n'	b'C\n'	Stop
12:50:45	0.3504	0.0	0.0	0	b'C\n'	b'C\n'	Stop
12:50:49	0.3535	0.0	0.0	0	b'A\n'	b'A\n'	Turn Left
12:51:07	0.3518	0.0	0.0	0	b'E\n'	b'E\n'	Turn Right
12:51:11	0.2897	0.0	0.0	0	b'E\n'	b'E\n'	Turn Right
12:51:26	0.3185	0.0	0.0	0	b'B\n'	b'B\n'	Forward
12:51:30	0.2687	0.0	0.0	0	b'B\n'	b'B\n'	Forward
12:51:31	0.2615	0.0	0.0	0	b'E\n'	b'E\n'	Turn Right
12:51:31	0.2711	0.0	0.0	0	b'E\n'	b'E\n'	Turn Right
12:51:38	0.2502	0.0	0.0	0	b'A\n'	b'A\n'	Turn Left
12:51:38	0.2617	0.0	0.0	0	b'A\n'	b'A\n'	Turn Left
12:52:05	0.6562	0.0	0.0	0	b'B\n'	b'B\n'	Forward
12:58:09	0.3319	0.0	0.0	0	b'B\n'	b'B\n'	Forward
12:58:11	0.2506	0.0	0.0	0	b'A\n'	b'A\n'	Turn Left
13:00:57	0.3574	0.0	0.0	0	b'C\n'	b'C\n'	Stop
13:01:07	0.2856	0.0	0.0	0	b'A\n'	b'A\n'	Turn Left
13:01:12	0.2639	0.0	0.0	0	b'B\n'	b'B\n'	Forward
13:01:16	1.3214	0.0	0.0	0	b'C\n'	b'C\n'	Stop
13:01:16	0.3159	0.0	0.0	0	b'C\n'	b'C\n'	Stop
13:01:17	0.3360	0.0	0.0	0	b'C\n'	b'C\n'	Stop
13:01:19	0.2638	0.0	0.0	0	b'C\n'	b'C\n'	Stop

Tabel 4.13 menunjukkan data status sistem saat objek berada di luar frame. Data ini memperlihatkan saat nilai dari deteksi objek keduanya adalah 0 dan keterangan mengenai status objek. Data ini memberikan gambaran kinerja sistem dalam menghadapi kondisi ketika objek keluar dari bidang pandang kamera, termasuk kecepatan respons, akurasi deteksi, dan ketepatan

instruksi yang dikirimkan.

Beberapa data telah difilter untuk menampilkan data yang menunjukkan sistem bekerja dengan sempurna ketika tidak ada objek yang terdeteksi dan mengirimkan kode instruksi sebelumnya tanpa penundaan yang lama ke mesin. Namun, ada masalah serius di mana sistem terus-menerus mengirimkan kode instruksi maju tanpa jaminan bahwa objek akan terdeteksi untuk mengubah kode instruksi ke mesin. Hal ini dapat menyebabkan potensi risiko keselamatan dan ketidakefisienan dalam operasi sistem.

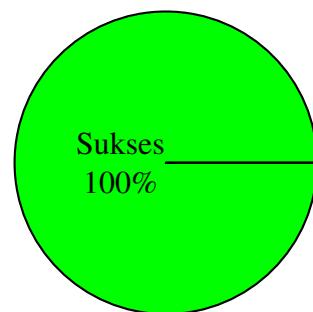
4.9 Performa Keberhasilan Mengikuti

Bagian ini menguji apakah sistem dapat mengikuti objek pada jalur khusus, seperti koridor, area dalam ruangan yang luas, dan jalur landai di sekitar Tower 2 di ITS. Gambar 4.14 merupakan contoh foto sampel yang akan digunakan.



Gambar 4.14: Dokumentasi Jalur

Pada pengujian ini hasil pengamatan kemudian dirangkum dalam bentuk visual untuk menunjukkan tingkat keberhasilan sistem secara keseluruhan.



Gambar 4.15: Pie Chart Performa Keberhasilan Sistem

Hasil pengujian menunjukkan bahwa sistem mampu mengikuti objek dengan baik di seluruh kondisi jalur yang diuji. Visualisasi dalam bentuk pie chart pada Gambar 4.15 mengonfirmasi tingkat keberhasilan sistem mencapai 100%. Hal ini mencerminkan bahwa sistem memiliki performa yang stabil dan dapat diandalkan untuk beroperasi pada berbagai jenis lingkungan yang telah disimulasikan selama pengujian. Keberhasilan ini menegaskan efektivitas sistem dalam memenuhi tujuan pengembangan untuk mengikuti objek dengan konsisten.

4.10 Pembahasan Hasil

Pada bagian ini, akan dibahas hasil pengujian yang telah dilakukan terhadap sistem kursi roda otonom berbasis teknik fusi gabungan YOLOv11 dan MediaPipe Pose untuk mendeteksi dan mengikuti manusia. Sistem ini dirancang untuk memanfaatkan kekuatan YOLOv11 dalam mendeteksi objek dengan presisi tinggi serta MediaPipe Pose untuk melacak posisi tubuh manusia secara real-time.

4.10.1 Performa Deteksi Objek

Berdasarkan hasil pengujian performa model menggunakan confusion matrix, terlihat bahwa model YOLOv11 mampu mendeteksi objek manusia dengan tingkat akurasi yang baik. Pada pelatihan dengan 100 epoch, model memiliki box loss sebesar 0.82978 pada tahap training dan 1.199 pada tahap validasi. Skor mAP (mean Average Precision) pada threshold 0.5 mencapai 81.85%, menunjukkan kemampuan model untuk mendeteksi objek dengan ketepatan yang tinggi.

Visualisasi hasil training dengan confusion matrix menunjukkan bahwa dari 4359 data train dan 1023 data validasi, model berhasil mendeteksi 1806 citra sebagai true positive dan 483 citra sebagai false positive. Pengujian ini mengindikasikan bahwa model cukup efektif dalam mendeteksi manusia, meskipun masih terdapat beberapa kesalahan deteksi.

4.10.2 Kecepatan Pemrosesan (FPS)

Pengujian kecepatan pemrosesan (FPS) dilakukan pada perangkat laptop sebagai unit komputasi menghasilkan data bahwa sistem mampu memproses frame secara real-time dengan mencatat resolusi frame, jumlah objek yang terdeteksi, dan waktu pemrosesan per frame. Dalam pengujian, sistem mencapai FPS tertinggi sebesar 62,5, dengan rata-rata FPS sekitar 40. Fluktuasi FPS dipengaruhi oleh kondisi lingkungan, kompleksitas objek, dan intensitas perhitungan.

4.10.3 Regulator Time

Pengujian ini mengukur waktu yang dibutuhkan sistem untuk meregulasi kode isntruksi sesuai dengan perubahan lingkungan, dengan mencatat data arah dan timestamp menggunakan Arduino IDE dan Visual Studio Code. Data yang dihasilkan, berupa Timestamp Receive ESP dan Timestamp Receive Motor, diolah menjadi plot distribusi waktu respons. Hasil pengujian menunjukkan bahwa 66,17% respons berada dalam waktu yang relatif singkat, sedangkan 34% lainnya melebihi 0,5 detik, menunjukkan adanya variasi kecil akibat kondisi lingkungan atau performa sistem.

4.10.4 Keberhasilan Tracking

Pengujian ini mengevaluasi keberhasilan sistem dalam melacak target menggunakan Bot-SORT dan YOLO pada 15 kali percobaan. Sistem berhasil mengunci target (Locked Detection) sebanyak 4 kali, sementara 11 kali terjadi kehilangan fokus (Distracted Detection), menghasilkan tingkat keberhasilan sebesar 26,67%. Ketidakstabilan dalam pelacakan terlihat dari peningkatan nomor track pada setiap objek yang terdeteksi, yang disebabkan oleh kurang optimalnya fungsi track(). Ketidakmampuan model untuk menjaga identitas ID secara konsisten juga menjadi faktor yang mengurangi akurasi pelacakan secara keseluruhan.

Meskipun demikian, ID yang terdeteksi tetap dapat dimanfaatkan sebagai referensi untuk verifikasi lebih lanjut atau sebagai dasar analisis data. Hal ini memungkinkan sistem untuk

tetap memberikan nilai praktis, seperti mendukung identifikasi pola pergerakan atau pengambilan keputusan berbasis data di lingkungan tertentu. Hasil ini menunjukkan bahwa meskipun sistem masih memerlukan peningkatan dalam hal kestabilan pelacakan, termasuk akurasi deteksi, kecepatan respons, dan pengelolaan ID objek, sistem tetap dapat digunakan untuk aplikasi lapangan yang relevan dan memberikan manfaat signifikan.

4.10.5 Kesesuaian Tingkat Pencahayaan

Pengujian ini mengevaluasi kemampuan sistem dalam mendeteksi objek pada kondisi pencahayaan yang berbeda, sistem menunjukkan kinerja terbaik pada siang hari ketika tingkat pencahayaan berada pada kondisi optimal, menghasilkan nilai kepercayaan yang lebih tinggi. Sebaliknya, kinerja sistem cenderung menurun pada malam hari akibat kurangnya pencahayaan, yang menyebabkan penurunan nilai kepercayaan pada deteksi objek.

Namun, penggunaan pencahayaan tambahan, seperti senter atau lampu, tidak direkomendasikan karena dapat menyebabkan overexposure pada gambar. Kondisi ini dapat mengurangi akurasi deteksi sistem secara keseluruhan. Oleh karena itu, untuk mencapai performa optimal, penting untuk memastikan bahwa tingkat pencahayaan lingkungan berada dalam rentang yang sesuai tanpa menggunakan pencahayaan buatan yang berlebihan. Hasil pengujian ini menegaskan pentingnya faktor pencahayaan dalam mendukung akurasi sistem deteksi real-time.

4.10.6 Kesesuaian Jarak Deteksi

Pengujian menunjukkan bahwa sistem mampu mendeteksi objek dengan jarak kurang dari 1 (satu) meter secara konsisten. Deteksi dilakukan menggunakan YOLOv11 dan MediaPipe Pose, dengan hasil berupa kode instruksi untuk menghentikan sistem ("Stop") guna mencegah tabrakan. Data mencakup waktu respons, jarak deteksi, dan ketepatan instruksi yang dikirimkan, menunjukkan bahwa sistem bekerja sesuai dengan parameter yang ditentukan. Verifikasi tambahan dilakukan dengan pengukuran manual dan perbandingan dengan alat ukur untuk memastikan akurasi deteksi.

4.10.7 Performa Pergerakan Mengikuti Objek

Evaluasi dilakukan untuk menilai kemampuan sistem dalam mendeteksi objek secara akurat, mengirimkan instruksi yang sesuai, serta menjaga konsistensi kinerja di berbagai kondisi dinamis. Hasil pengujian menunjukkan bahwa:

- Pada pengujian dalam frame, sistem mencatat sebagian besar data terkirim sebagai register 'C' dengan persentase sebesar 55.4%, sementara kesalahan transmisi hanya sebesar 6.3%.
- Pada kondisi bergerak maju, instruksi 'Forward' terkirim dengan persentase kesalahan transmisi yang sangat kecil, yaitu hanya 1.5%.
- Pada kondisi belok kiri, sistem mencatat data terkirim sebagai register 'C' sebesar 10.5%, dengan kesalahan transmisi sebesar 2.0%.
- Pada kondisi belok kanan, sistem mencatat data terkirim sebagai register 'C' sebesar 6.7%, dengan kesalahan transmisi sebesar 1.5%.
- Saat objek keluar dari frame, sistem mencatat sebagian besar data dengan nilai deteksi nol, dan prioritas diberikan untuk mengirimkan instruksi sebelumnya.

Hasil pengujian menunjukkan bahwa sistem memiliki performa yang sangat baik dalam mendeteksi dan mengikuti objek di berbagai skenario, baik saat dalam frame, bergerak maju, maupun berbelok. Sistem mampu menunjukkan stabilitas tinggi dalam transmisi data dan respons yang sesuai terhadap deteksi. Meski demikian, diperlukan perbaikan lebih lanjut dalam menangani kondisi luar frame, terutama untuk mencegah pengiriman instruksi yang tidak relevan atau tidak aman. Dengan pengoptimalan tersebut, sistem akan semakin andal dan aman untuk diterapkan pada berbagai kondisi operasional.

4.10.8 Performa Keberhasilan Mengikuti

Hasil pengujian menunjukkan bahwa sistem mampu mengikuti objek dengan baik di seluruh kondisi jalur yang diuji, termasuk koridor, area dalam ruangan, dan jalur landai, dengan tingkat keberhasilan mencapai 100%. Keberhasilan ini menunjukkan efektivitas sistem dalam memenuhi tujuan pengembangan untuk mengikuti objek secara akurat dan stabil, memberikan dasar yang kuat untuk implementasi lebih lanjut pada aplikasi nyata.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Model YOLOv11 menunjukkan performa deteksi yang tinggi dengan skor mAP mencapai 81.85% pada threshold 0.5.
2. Sistem mampu memproses frame secara real-time dengan rata-rata FPS sekitar 40, menunjukkan kecepatan pemrosesan yang memadai untuk aplikasi lapangan.
3. Pengujian regulator time menunjukkan bahwa 66.17% respons berada dalam waktu yang relatif singkat, namun 34% lainnya melebihi 0.5 detik.
4. Sistem berhasil mengunci target (Locked Detection) sebanyak 4 kali dari 15 percobaan, menghasilkan tingkat keberhasilan sebesar 26.67%.
5. Sistem menunjukkan kinerja terbaik pada kondisi pencahayaan optimal, namun kinerja menurun pada kondisi pencahayaan rendah.
6. Sistem mampu mendeteksi objek dengan jarak kurang dari 1 meter secara konsisten dan mengirimkan instruksi "Stop" untuk mencegah tabrakan.
7. Sistem memiliki performa yang sangat baik dalam mendeteksi dan mengikuti objek di berbagai skenario, baik saat dalam frame, bergerak maju, maupun berbelok.
8. Sistem mampu mengikuti objek dengan baik di seluruh kondisi jalur yang diuji, dengan tingkat keberhasilan mencapai 100%.

5.2 Saran

Untuk pengembangan lebih lanjut pada penelitian selanjutnya, adapun saran yang bisa diberikan antara lain:

1. Meningkatkan fungsi `track()` untuk menjaga konsistensi identitas ID objek yang terdeteksi, guna meningkatkan akurasi pelacakan.
2. Menggunakan pencahayaan tambahan yang tidak menyebabkan overexposure dapat diperimbangkan untuk meningkatkan kinerja deteksi pada kondisi pencahayaan rendah.
3. Memberikan optimisasi lebih lanjut pada sistem untuk mengurangi waktu respons yang melebihi 0.5 detik, guna meningkatkan stabilitas dan konsistensi kinerja.
4. Menuliskan algoritma yang lebih adaptif untuk menangani kondisi luar frame, sehingga dapat mencegah pengiriman instruksi yang tidak relevan atau tidak aman.
5. Implementasi sistem pada skenario nyata dengan berbagai kondisi operasional untuk menguji dan memastikan keandalan serta keamanan sistem secara menyeluruh.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

- [1] I. Ekatama, “Perancangan sistem kontrol motor kursi roda secara nirkabel berbasis esp32,” Ph.D. dissertation, Institut Teknologi Sepuluh Nopember, 2024.
- [2] W. K. Wijaya, I. K. Somawirata, and R. P. M. D. Labib, “Deteksi objek menggunakan yolo v3 untuk keamanan pada pergerakan kursi roda elektrik,” *Nucleus Journal*, vol. 1, no. 2, pp. 94–98, 2022.
- [3] R. P. Narwaria, A. Ahirwar, A. K. Prajapati, A. Kumar, and A. K. Tiwari, “Smart object detection using esp32-cam based on yolo algorithm,” in *2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*, 2024, pp. 817–820. doi: [10.1109/ICoICI62503.2024.10696374](https://doi.org/10.1109/ICoICI62503.2024.10696374).
- [4] K. O’Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: [1511.08458 \[cs.NE\]](https://arxiv.org/abs/1511.08458). [Online]. Available: <https://arxiv.org/abs/1511.08458>.
- [5] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, *Bot-sort: Robust associations multi-pedestrian tracking*, 2022. arXiv: [2206.14651 \[cs.CV\]](https://arxiv.org/abs/2206.14651). [Online]. Available: <https://arxiv.org/abs/2206.14651>.
- [6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *ICIP*, IEEE, 2016, pp. 3464–3468.
- [7] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE international conference on image processing (ICIP)*, IEEE, 2017, pp. 3645–3649.
- [8] J. Shi *et al.*, “Good features to track,” in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, IEEE, 1994, pp. 593–600.
- [9] J.-Y. Bouguet, “Pyramidal implementation of the lucas kanade feature tracker,” 1999.
- [10] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [11] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, “Mot16: A benchmark for multi-object tracking,” *arXiv preprint arXiv:1603.00831*, 2016.
- [12] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, and T. Mei, “Fastreid: A pytorch toolbox for general instance re-identification,” *arXiv preprint arXiv:2006.02631*, 2020.
- [13] H. Zhang, C. Wu, Z. Zhang, *et al.*, “Resnest: Split-attention networks,” *arXiv preprint arXiv:2004.08955*, 2020.
- [14] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, “Towards real-time multi-object tracking,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, Springer, 2020, pp. 107–122.
- [15] H. W. Kuhn, “The hungarian method for the assignment problem,” *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Kode Program

Program 5.1: Program Deteksi Fusi pada Unit Kontrol.

```
1 import cv2
2 import math
3 import mediapipe as mp
4 import socket
5 import torch
6 import time
7 import csv
8 from enum import Enum
9 from ultralytics import YOLO
10
11 source = "http://camera.local:81/stream"
12 # source = 0
13 host = "192.168.4.1"
14 port = 80
15
16 # Initialize MediaPipe Pose
17 mp_pose = mp.solutions.pose
18 pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.5)
19 mirror = True
20
21 fp = 481 # Focal length in pixelsCV2
22 tb = 175 # Real-world height of the object in cm
23 k = 24.422 # Calibration factor for MediaPipe
24
25 # Load YOLO model
26 model = YOLO("best.pt")
27
28 def hitung_jarak(tinggi_bounding_box, focal_length_pixel, tinggi_objek_nyata):
29     if tinggi_bounding_box == 0:
30         return float('inf')
31     jarak = (tinggi_objek_nyata * focal_length_pixel) / tinggi_bounding_box
32     return jarak / 100 # Convert to meters
33
34 def hitung_jarak_euclidean(landmark1, landmark2, lebar_img):
35     return math.sqrt((landmark1.x - landmark2.x) ** 2 + (landmark1.y - landmark2.y) ** 2) * lebar_img
36
37 def hitung_lebar_mediapipe(pose_results, lebar_img):
38     if pose_results.pose_landmarks:
39         bahu_kiri = pose_results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_SHOULDER]
40         bahu_kanan = pose_results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_SHOULDER]
41         jarak_pix = hitung_jarak_euclidean(bahu_kiri, bahu_kanan, lebar_img)
42         return (k / jarak_pix) * 10
43     return 0
44
```

```

45 class SocketCommunicator:
46     def __init__(self, host, port):
47         self.socket = None
48         self.interval = 0
49         self.data = 'C\n'.encode()
50         self.sent = True
51         self.sentr = True
52         self.host = host
53         self.port = port
54         self.ex_time = 0
55         self.ex_data = 'C\n'.encode()
56         self.create_socket()
57
58     def create_socket(self):
59         print("Loading...", end="", flush=True)
60         s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
61         s.settimeout(1)
62         try:
63             s.connect((self.host, self.port))
64             print()
65             self.socket = s
66         except socket.error:
67             print(".", end="", flush=True)
68             time.sleep(1)
69
70     def send_data(self, data, regulator='C\n'):
71         verbose = False
72         over_delay = (time.time() - self.interval) > .25
73         if over_delay and self.data != regulator:
74             if not self.sent:
75                 verbose = True
76                 self.ex_time = self.interval
77                 self.interval = time.time()
78                 if self.socket:
79                     self.socket.send(self.data.encode())
80                     self.ex_data = self.data
81                     self.sent = True
82                     self.sentr = True
83             elif self.data != data:
84                 verbose = True
85                 self.ex_time = self.interval
86                 self.interval = time.time()
87                 if (data != regulator):
88                     self.data = data
89                     self.sent = False
90                 if self.sentr == True:
91                     if self.socket:
92                         self.socket.send(regulator.encode())
93                         self.ex_data = regulator
94                         self.sentr = False
95         return verbose
96
97 # Prepare CSV file
98 csv_file = open('log_belokan.csv', mode='w', newline='')
99 csv_writer = csv.writer(csv_file)
100 csv_writer.writerow(['Waktu', 'Inference Time',
101                      'Jarak YOLO', 'Jarak MediaPipe', 'x-mid', 'near'],

```

```

102         'Regulasi Waktu', 'Menyimpan', 'Terkirim',
103         'Mencari', 'Keterangan'])
104
105 # Initialize video capture
106 cap = cv2.VideoCapture(source)
107 cap.set(3, 1366)
108 cap.set(4, 768)
109
110 class Direction(Enum):
111     DIAM = 0
112     MAJU = 1
113     BELIKANAN = 2
114     BELIKIRI = 3
115
116 class PosisiManusiaTerakhir(Enum):
117     DIAM = 0
118     KIRI = 1
119     KANAN = 2
120     MAJU = 3
121
122 posisi_terakhir = PosisiManusiaTerakhir.DIAM
123
124 def draw_arrow(img, direction):
125     height, width, _ = img.shape
126     start_point = (width // 2, height - 50) # Start point of arrow
127     if direction == Direction.MAJU:
128         end_point = (width // 2, height - 150)
129     elif direction == Direction.BELIKANAN:
130         end_point = (width // 2 + 100, height - 100)
131     elif direction == Direction.BELIKIRI:
132         end_point = (width // 2 - 100, height - 100)
133     else:
134         end_point = start_point
135
136     color = (0, 255, 0)
137     thickness = 5
138     cv2.arrowedLine(img, start_point, end_point, color, thickness)
139
140
141 # Initialize the wheelchair controller
142 controller = SocketCommunicator(host, port)
143 controller.send_data('2\n') # Speed level .140
144
145 while True:
146     success, frame = cap.read()
147     if not success:
148         break
149
150     # frame = cv2.rotate(frame, cv2.ROTATE_180) if source else frame
151     img = cv2.flip(frame,1) if mirror else frame
152
153     img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
154     height, width, _ = img.shape
155
156     # Define screen partitions
157     left_bound = width // 3
158     right_bound = 2 * (width // 3)

```

```

159
160     # Draw partition lines
161     cv2.line(img, (left_bound, 0), (left_bound, height), (0, 255, 0), 2)
162     cv2.line(img, (right_bound, 0), (right_bound, height), (0, 255, 0), ←
163             2)
164
165     if torch.cuda.is_available(): torch.cuda.synchronize()
166     start_inference_time = time.time()
167     results = model.track(img, classes=0, stream=True, persist=True, conf←
168             =0.7, verbose=False)
169
170     if torch.cuda.is_available(): torch.cuda.synchronize()
171     inference_time = time.time() - start_inference_time
172
173     lf = False
174     arah = 'C\n'
175     center_x = 0
176     jarak_yolo = 0
177     jarak_mediapipe = 0
178     arah_log = "Menunggu jarak aman"
179     direction = Direction.DIAM
180
181     for r in results:
182         min_id = None
183         min_box = None
184         boxes = r.boxes
185         for box in boxes:
186             idex = box.id
187             if min_id is None or idex < min_id:
188                 min_id = idex
189                 min_box = box
190
191     if min_id:
192         x1, y1, x2, y2 = map(int, min_box.xyxy[0])
193         tinggi_bounding_box = y2 - y1
194
195         # Draw bounding box
196         cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
197         cv2.rectangle(img, (x1, y1), (x1+20, y1+30), (255, 0, 0), -1)
198         cv2.putText(img, str(int(min_id)), (x1, y1 + 25), cv2.←
199                         FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
200
201         # Calculate distance using YOLO
202         jarak_yolo = hitung_jarak(tinggi_bounding_box, fp, tb)
203
204         # Process with MediaPipe
205         person_img_rgb = cv2.cvtColor(img[y1:y2, x1:x2], cv2.←
206             COLOR_BGR2RGB)
207         pose_results = pose.process(person_img_rgb)
208
209         if pose_results.pose_landmarks:
210             # Draw pose landmarks
211             mp.solutions.drawing_utils.draw_landmarks(img[y1:y2, x1:←
212                 x2], pose_results.pose_landmarks, mp_pose.←
213                     POSE_CONNECTIONS)
214
215             lebar_img = person_img_rgb.shape[1]

```

```

210     jarak_mediapipe = hitung_lebar_mediapipe(pose_results, ←
211         lebar_img)
212
213     # print(f"YOLO Distance: {jarak_yolo:.2f} m, MediaPipe ←
214         Distance: {jarak_mediapipe:.2f} m")
215     cv2.putText(img, f"YOLO Distance: {jarak_yolo:.2f} m", ←
216         (10, 180), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), ←
217         2)
218     cv2.putText(img, f"MP Distance: {jarak_mediapipe:.2f} m", ←
219         (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), ←
220         2)
221
222     cp_Shape = lebar_img / img.shape[1]
223     center_x = (x1 + x2) // 2
224
225     if jarak_mediapipe > 1.0:
226         if center_x < left_bound:
227             arah = 'A\n'
228             arah_log = "Kursi roda belok Kiri"
229             direction = Direction.BELIKIRI
230             posisi_terakhir = PosisiManusiaTerakhir.KIRI
231         elif center_x > right_bound:
232             arah = 'E\n'
233             arah_log = "Kursi roda belok Kanan"
234             direction = Direction.BELIKANAN
235             posisi_terakhir = PosisiManusiaTerakhir.KANAN
236         else:
237             arah = 'B\n'
238             arah_log = "Kursi roda Maju"
239             direction = Direction.MAJU
240             posisi_terakhir = PosisiManusiaTerakhir.MAJU
241         else:
242             posisi_terakhir = PosisiManusiaTerakhir.DIAM
243
244     else:
245         lf = True
246         if posisi_terakhir == PosisiManusiaTerakhir.KIRI:
247             arah = 'A\n'
248             arah_log = "Mencari manusia ke Kiri"
249             direction = Direction.BELIKIRI
250             posisi_terakhir == PosisiManusiaTerakhir.KANAN:
251                 arah = 'E\n'
252                 arah_log = "Mencari manusia ke Kanan"
253                 direction = Direction.BELIKANAN
254             elif posisi_terakhir == PosisiManusiaTerakhir.MAJU:
255                 arah = 'B\n'
256                 arah_log = "Mencari manusia ke Depan"
257                 direction = Direction.MAJU
258             elif posisi_terakhir == PosisiManusiaTerakhir.DIAM:
259                 arah_log = "Menunggu jarak aman"
260                 direction = Direction.DIAM
261
262             cv2.putText(img, arah_log, (10, 50), cv2.FONT_HERSHEY_SIMPLEX, ←
263                 0.6, (0, 0, 255), 2)
264
265             # Send and Write log to CSV
266             if controller.send_data(arah):
267                 waktu = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())

```

```

260     text_lf = "Luar frame" if lf else "Dalam frame"
261     inter_send = time.time() - controller.ex_time
262     near_bound = 0 if not center_x else \
263         left_bound if abs(center_x-left_bound) \
264             < abs(center_x-right_bound) else right_bound
265     csv_writer.writerow([waktu, inference_time,
266         jarak_yolo, jarak_mediapipe, center_x, near_bound,
267         inter_send, arah.encode(), controller.ex_data.encode(),
268         text_lf, arah_log])
269     csv_file.flush() # Ensure data is written immediately
270
271     draw_arrow(img, direction)
272     cv2.imshow('Webcam', img)
273     if cv2.waitKey(1) & 0xFF == ord('q'):
274         break
275
276 cap.release()
277 cv2.destroyAllWindows()
278 csv_file.close()

```

Program 5.2: Program Stream Frame dan Forward Kode Instruksi.

```

1 #include <WiFi.h>
2 #include <esp_now.h>
3 #include "esp_camera.h"
4 #include "esp_http_server.h"
5
6 #define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
7 #define PWDN_GPIO_NUM -1
8 #define RESET_GPIO_NUM -1
9 #define XCLK_GPIO_NUM 15
10 #define SIOD_GPIO_NUM 4
11 #define SIOC_GPIO_NUM 5
12
13 #define Y2_GPIO_NUM 11
14 #define Y3_GPIO_NUM 9
15 #define Y4_GPIO_NUM 8
16 #define Y5_GPIO_NUM 10
17 #define Y6_GPIO_NUM 12
18 #define Y7_GPIO_NUM 18
19 #define Y8_GPIO_NUM 17
20 #define Y9_GPIO_NUM 16
21
22 #define VSYNC_GPIO_NUM 6
23 #define HREF_GPIO_NUM 7
24 #define PCLK_GPIO_NUM 13
25
26 // =====
27 // Enter your WiFi credentials
28 // =====
29 const char* ssid = "ESP32-Camera-AP";
30 const char* password = "123456789";
31
32 // Alamat MAC ESP B (penerima)
33 uint8_t broadcastAddress[] = {0xD0, 0xEF, 0x76, 0xEF, 0xE5, 0x4C};
34
35 // Struktur data yang akan dikirim melalui ESP-NOW

```

```

36 typedef struct struct_message {
37     char data[8]; // Data yang akan dikirimkan, bisa diubah sesuai ↵
38 } struct_message;
39
40 struct_message msg;
41
42 // Inisialisasi server Wi-Fi di port 80
43 WiFiServer server(80);
44
45 void initCamera() {
46     camera_config_t config;
47     config.ledc_channel = LEDC_CHANNEL_0;
48     config.ledc_timer = LEDC_TIMER_0;
49     config.pin_d0 = Y2_GPIO_NUM;
50     config.pin_d1 = Y3_GPIO_NUM;
51     config.pin_d2 = Y4_GPIO_NUM;
52     config.pin_d3 = Y5_GPIO_NUM;
53     config.pin_d4 = Y6_GPIO_NUM;
54     config.pin_d5 = Y7_GPIO_NUM;
55     config.pin_d6 = Y8_GPIO_NUM;
56     config.pin_d7 = Y9_GPIO_NUM;
57     config.pin_xclk = XCLK_GPIO_NUM;
58     config.pin_pclk = PCLK_GPIO_NUM;
59     config.pin_vsync = VSYNC_GPIO_NUM;
60     config.pin_href = HREF_GPIO_NUM;
61     config.pin_sscb_sda = SIOD_GPIO_NUM;
62     config.pin_sscb_scl = SIOC_GPIO_NUM;
63     config.pin_pwdn = PWDN_GPIO_NUM;
64     config.pin_reset = RESET_GPIO_NUM;
65     config.xclk_freq_hz = 20000000;
66     config.pixel_format = PIXFORMAT_JPEG;
67
68     // PSRAM handling
69     if (psramFound()) {
70         config.frame_size = FRAMESIZE_XGA;
71         config.jpeg_quality = 10;
72         config.fb_count = 2;
73     } else {
74         config.frame_size = FRAMESIZE_QVGA;
75         config.jpeg_quality = 12;
76         config.fb_count = 1;
77     }
78
79     // Inisialisasi kamera
80     esp_err_t err = esp_camera_init(&config);
81     if (err != ESP_OK) {
82         Serial.printf("Kamera gagal diinisialisasi dengan kode error 0x%x", ↵
83                     err);
84         return;
85     }
86     Serial.println("Kamera berhasil diinisialisasi");
87 }
88 void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
89     Serial.print("\nPengiriman data ke MAC: ");
90     for (int i = 0; i < 6; i++) {

```

```

91     Serial.printf("%02X", mac_addr[i]);
92     if (i < 5) Serial.print(":");
93 }
94 Serial.print("\nStatus pengiriman: ");
95 Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Terkirim" : "Gagal");
96 }
97
98 void sendData() {
99     WiFiClient client = server.available();
100    if (client) {
101        Serial.println("Client terhubung");
102        while (client.connected()) {
103            if (client.available()) {
104                // Membaca data yang diterima dari komputer
105                String request = client.readStringUntil('\n');
106                client.flush(); // Membersihkan buffer
107
108                // Menampilkan data yang diterima di serial monitor
109                Serial.print("Data dari port 80: ");
110                Serial.println(request);
111
112                // Mengirim data yang diterima ke ESP B melalui ESP-NOW
113                request.toCharArray(msg.data, sizeof(msg.data));
114                esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &msg, sizeof(msg));
115                if (result == ESP_OK) {
116                    Serial.println("Data dikirim ke ESP B melalui ESP-NOW");
117                } else {
118                    Serial.print("Gagal mengirim data ke ESP B. Error code: ");
119                    Serial.println(result); // Tampilkan error code untuk debugging
120                }
121            }
122        }
123        client.stop();
124        Serial.println("Client terputus");
125    }
126 }
127
128 // Server handler for streaming MJPEG
129 static esp_err_t stream_handler(httpd_req_t *req) {
130     camera_fb_t * fb = NULL;
131     esp_err_t res = ESP_OK;
132     size_t _jpg_buf_len;
133     uint8_t * _jpg_buf;
134     char * part_buf[64];
135
136     res = httpd_resp_set_type(req, "multipart/x-mixed-replace; boundary=--frame");
137
138     if(res != ESP_OK){
139         return res;
140     }
141
142     while(true){
143         fb = esp_camera_fb_get();
144         if (!fb) {

```

```

145     Serial.println("Camera capture failed");
146     res = ESP_FAIL;
147 } else {
148     _jpg_buf_len = fb->len;
149     _jpg_buf = fb->buf;
150     res = httpd_resp_send_chunk(req, "--frame\r\n", strlen("--frame\r\n←
151     "));
152     if (res == ESP_OK){
153         res = httpd_resp_send_chunk(req, "Content-Type: image/jpeg\r\n←
154         r\n",
155             strlen("Content-Type: image/jpeg\r\n←
156             r\n"));
157     }
158     if (res == ESP_OK){
159         res = httpd_resp_send_chunk(req, "\r\n", strlen("\r\n"));
160     }
161     esp_camera_fb_return(fb);
162 }
163 if(res != ESP_OK){
164     break;
165 }
166 }
167 return res;
168 }

169 void startCameraServer(){
170     httpd_config_t config = HTTPD_DEFAULT_CONFIG();
171     config.server_port = 81;
172
173     httpd_handle_t server = NULL;
174     if (httpd_start(&server, &config) == ESP_OK) {
175         httpd_uri_t stream_uri = {
176             .uri      = "/stream",
177             .method   = HTTP_GET,
178             .handler  = stream_handler,
179             .user_ctx = NULL
180         };
181         httpd_register_uri_handler(server, &stream_uri);
182     }
183 }
184 }

185

186 void setup() {
187     Serial.begin(115200);
188     Serial.setDebugOutput(true);
189     Serial.println();
190
191     initCamera();
192
193 // Set up WiFi Access Point
194 WiFi.mode(WIFI_AP_STA);
195 WiFi.softAP(ssid, password);
196 IPAddress IP = WiFi.softAPIP();

```

```

198 Serial.print("AP IP address: ");
199 Serial.println(IP);
200
201 // Start streaming
202 startCameraServer();
203 Serial.print("Camera Ready! Use 'http://");
204 Serial.print(IP);
205 Serial.println(":81/stream' to connect");
206
207 // Dapatkan channel Wi-Fi yang digunakan oleh AP
208 int channel = WiFi.channel();
209 Serial.print("Channel AP (ESP A): ");
210 Serial.println(channel);
211
212 // Mulai server Wi-Fi di port 80
213 server.begin();
214 Serial.println("Server berjalan di port 80");
215
216 if (esp_now_init() != ESP_OK) {
217     Serial.println("Error inisialisasi ESP-NOW");
218     return;
219 }
220 Serial.println("ESP-NOW berhasil diinisialisasi");
221
222 // Register callback untuk status pengiriman ESP-NOW
223 esp_now_register_send_cb(OnDataSent);
224
225 // Tambahkan peer ESP-NOW (ESP B)
226 esp_now_peer_info_t peerInfo;
227 memcpy(peerInfo.peer_addr, broadcastAddress, 6);
228 peerInfo.channel = channel; // Menggunakan channel yang sama dengan AP
229 peerInfo.encrypt = false;
230 if (esp_now_add_peer(&peerInfo) != ESP_OK) {
231     Serial.println("Error menambahkan peer ESP-NOW");
232     return;
233 }
234 Serial.println("Peer ESP B berhasil ditambahkan");
235 }
236
237 void loop() {
238     sendData();
239 }
```

Program 5.3: Program Pengolahan Kode Instruksi Pada Motor.

```

1 #include <WiFi.h>
2 #include <Arduino.h>
3 #include <esp_now.h>
4
5 // Motor Kiri
6 #define pwmpin1 5
7 #define dir1 18
8 #define dir2 19
9
10 // Motor kanan
11 #define pwmpin2 25
12 #define dir3 32
```

```

13 #define dir4 33
14
15 #define pwmChannel1 0
16 #define pwmChannel2 1
17 #define freq 15000
18 #define res 8
19
20 int PWM1_DutyCycle = 0;
21 int maxspeed = 70;
22 int turnspeed = 35;
23 int kecepatan = 0;
24
25 const char* ssid = "ESP32-Camera-AP";
26 const char* password = "123456789";
27
28 char arah[200];
29 bool _mac = true;
30
31 // Fungsi untuk membagi arah berdasarkan 5 level
32 void setMaxSpeed(char arah[]) {
33     int level = atoi(arah);
34     maxspeed = 70 + 37 * (level - 1);
35     Serial.print("Max Speed: ");
36     Serial.println(maxspeed);
37 }
38
39 // Fungsi untuk mengontrol motor dengan 4 parameter
40 void kontrolMotor(int d1, int d2, int d3, int d4) {
41     // Tentukan target speed
42     int targetSpeed;
43     d1 == HIGH && d3 == HIGH ?
44         targetSpeed = maxspeed : // Jika d1 dan d3 HIGH, gunakan maxspeed
45     d1 == HIGH || d2 == HIGH || d3 == HIGH || d4 == HIGH ?
46         targetSpeed = turnspeed : // Jika ada salah satu HIGH, gunakan ←
47             turnspeed
48         targetSpeed = 0; // Jika semuanya LOW, kecepatan 0
49
50     // Write pin direction
51     digitalWrite(dir1, d1);
52     digitalWrite(dir2, d2);
53     digitalWrite(dir3, d3);
54     digitalWrite(dir4, d4);
55
56     // Tentukan apakah kecepatan naik atau turun
57     if (PWM1_DutyCycle < targetSpeed) {
58         ledcWrite(pwmChannel1, PWM1_DutyCycle++);
59         ledcWrite(pwmChannel2, PWM1_DutyCycle++);
60     }
61     else {
62         ledcWrite(pwmChannel1, PWM1_DutyCycle--);
63         ledcWrite(pwmChannel2, PWM1_DutyCycle--);
64     }
65     delay(10);
66 }
67 void OnDataRecv(const esp_now_recv_info* recvInfo, const uint8_t *data, ←
    int data_len) {

```

```

68 char macStr[18]; // Buffer untuk menyimpan string MAC address
69 sprintf(macStr, sizeof(macStr), "%02X:%02X:%02X:%02X:%02X:%02X",
70     recvInfo->src_addr[0], recvInfo->src_addr[1], recvInfo->src_addr[2],
71     recvInfo->src_addr[3], recvInfo->src_addr[4], recvInfo->src_addr[5]);
72
73 Serial.print("Received data from: ");
74 Serial.println(macStr);
75 _mac = false;
76
77 if (data_len < sizeof(arah)) {
78     memcpy(arah, data, data_len);
79     arah[data_len] = '\0';
80 } else {
81     Serial.println("Received data is too large to fit in 'arah' buffer.") ←
82         ;
83 }
84 Serial.print("Arah: ");
85 Serial.println(arah);
86
87 strcmp(arah, "1") == 0 || strcmp(arah, "2") == 0 || strcmp(arah, "3") ←
88     == 0 ||
89 strcmp(arah, "4") == 0 || strcmp(arah, "5") == 0 ?
90     setMaxSpeed(arah):
91
92     strcmp(arah, "A") == 0 ?
93         kontrolMotor(LOW, LOW, HIGH, LOW):
94     strcmp(arah, "B") == 0 ?
95         kontrolMotor(HIGH, LOW, HIGH, LOW):
96     strcmp(arah, "C") == 0 ?
97         kontrolMotor(LOW, LOW, LOW, LOW):
98     strcmp(arah, "D") == 0 ?
99         kontrolMotor(LOW, HIGH, LOW, HIGH):
100    strcmp(arah, "E") == 0 ?
101        kontrolMotor(HIGH, LOW, LOW, LOW):
102    delay(10);
103 }
104 void setup() {
105     Serial.begin(115200);
106
107     WiFi.mode(WIFI_STA);
108     WiFi.begin(ssid, password);
109     while (WiFi.status() != WL_CONNECTED) {
110         delay(100);
111         Serial.print(".");
112     }
113     Serial.println("WiFi connected.");
114
115     pinMode(dir1, OUTPUT);
116     pinMode(dir2, OUTPUT);
117     pinMode(dir3, OUTPUT);
118     pinMode(dir4, OUTPUT);
119
120     // Penggunaan ledcAttach untuk mengatur PWM pada pin
121     ledcAttachChannel(pwmpin1, freq, res, pwmChannel1);
122     ledcAttachChannel(pwmpin2, freq, res, pwmChannel2);

```

```
123
124 // Inisialisasi ESP-NOW
125 if (esp_now_init() != ESP_OK) {
126     Serial.println("Error initializing ESP-NOW");
127     return;
128 }
129 Serial.print("Alamat MAC ESP32 B: ");
130 Serial.println(WiFi.macAddress());
131
132 // Register callback untuk menerima pesan ESP-NOW
133 esp_now_register_recv_cb(OnDataRecv);
134 }
135
136 void loop() {}
```

[Halaman ini sengaja dikosongkan]

BIOGRAFI PENULIS



Aldifahmi Sihotang, lahir pada 24 Mei 2000 di kota Padangsidimpuan. Penulis merupakan anak ketiga dari 3 bersaudara. Setelah lulus dari MA Negeri 2 Padangsidimpuan, Penulis kemudian melanjutkan pendidikan ke jenjang strata satu di Departemen Teknik Komputer ITS, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember mulai tahun 2018.

Penulis merupakan orang yang aktif berorganisasi, dan memiliki berbagai softskill serta hardskill. Hal ini dibuktikan dengan rekam jejak organisasi dan kepanitiaan dari penulis seperti Staff Hima Teknik komputer ITS, Bangkit Academy Mobile Development.

[Halaman ini sengaja dikosongkan]