



TUGAS AKHIR - TD123456

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

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024



TUGAS AKHIR - TD123456

**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

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

[Halaman ini sengaja dikosongkan]



FINAL PROJECT - TD123456

***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

Undergraduate Study Program of Computer Engineering

Department of Computer Engineering

Faculty of Intelligent Electrical and Informatics Technology

Sepuluh Nopember Institute of Technology

Surabaya

2024

[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)

NIP:

(Pembimbing II)

NIP:

(Penguji I)

NIP:

(Penguji II)

NIP:

(Penguji III)

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

NIP.

SURABAYA
Okttober, 2024

[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
NIP: 19680601 1995121 1 009

(Advisor I)

.....
(Co-Advisor II)

NIP:

.....
(Examiner I)

NIP:

.....
(Examiner II)

NIP:

.....
(Examiner III)

NIP:

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

.....
NIP.

SURABAYA
October, 2024

[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, October 2024

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, October 2024

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 : Dr. Eko Mulyanto Yuniarno, 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 mendeteksi 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 : Dr. Eko Mulyanto Yuniarno, S.T., M.T*

This study aims to develop an autonomous wheelchair control system that can follow human movement in real-time. This system integrates the object detection algorithm YOLOv11 and body motion tracking using MediaPipe Pose, utilizing the optimal viewing angle as the basis for visual data collection. The camera is placed on the user's glasses to obtain the optimal viewing angle in detecting and tracking user movement. This system is designed to operate wirelessly using the ESP32 module, providing 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 showed that the system can follow user movement accurately and responsively, making a significant contribution to the development of smarter and more 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 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

Penelitian ini disusun dalam rangka Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara tercinta yang telah Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero.
2. Bapak Nikola Tesla, S.T., M.T., selaku Quisque ullamcorper placerat ipsum. Cras nibh.
3. Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl.

Akhir kata, semoga Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus.

Surabaya, Oktober 2024

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>)	4
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	5
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	21
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	Hasil Pengujian Performa Menggunakan Confusion Matrix	35
4.2	Pengujian Berdasarkan FPS	35
4.3	Pengujian Berdasarkan Response Time	35
4.4	Performa Keberhasilan Tracking	35
4.5	Pengujian Kesesuaian Jarak Deteksi	35
4.6	Performa Pergerakan Mengikuti Objek	35
4.6.1	Belok Kiri saat berada didalam frame	36
4.6.2	Belok Kiri saat berada diluar frame	36
4.6.3	Belok Kanan saat berada didalam frame	36
4.6.4	Belok Kanan saat berada diluar frame	36
4.7	Performa Keberhasilan Mengikuti	36
4.8	Pembahasan Hasil	36
4.8.1	Performa Deteksi Objek	36
4.8.2	Kecepatan Pemrosesan (FPS)	36
4.8.3	Response Time	36
4.8.4	Performa Keberhasilan Tracking	36
4.8.5	Kesesuaian Jarak Deteksi	36
4.8.6	Performa Pergerakan Mengikuti Objek	37
4.8.7	Performa Keberhasilan Mengikuti	37
5	PENUTUP	39
5.1	Kesimpulan	39
5.2	Saran	39
DAFTAR PUSTAKA		41
LAMPIRAN		43
Kode Program		43

DAFTAR GAMBAR

2.1	Arsitektur YOLOv8	6
2.2	YOLOv8 Pose.	8
2.3	Arsitektur YOLOv10.	9
2.4	Arsitektur Yolov11	10
2.5	Modul Bottleneck, C3, C2F, dan C3K2	10
2.6	MediaPipe 3D.	11
2.7	MediaPipe Pose.	12
2.8	Matrix Konfusi.	12
2.9	Intersection over Union.	15
2.10	Kalman Filter bbox.	16
2.11	Kompensasi Gerakan Kamera	18
2.12	Interface RoboFlow	19
2.13	Interface OBS Studio	20
2.14	Gambar ESP32 Devkit V1	21
2.15	Gambar Motor Driver H-Bridge	21
2.16	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

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

3.1 Kode Instruksi dari Hasil Klasifikasi	25
3.2 Tabel Keypoint yang digunakan	29

[Halaman ini sengaja dikosongkan]

DAFTAR PROGRAM

5.1	Program Deteksi Fusi pada Unit Kontrol.	43
5.2	Program Stream Frame dan Forward Kode Instruksi.	48
5.3	Program Pengolahan Kode Instruksi Pada Motor.	52

[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).

YOLO, salah satu metode deteksi objek paling populer, mampu melakukan deteksi dengan kecepatan tinggi dan akurasi yang baik, menjadikannya sangat cocok untuk aplikasi *real-time* seperti kursi roda otonom. *YOLOv11*, versi terbaru dari keluarga *YOLO*, menawarkan peningkatan signifikan dalam kecepatan dan akurasi, serta tambahan fitur seperti *YOLO pose* untuk deteksi pose manusia. Selain itu, metode pelacakan objek seperti *DeepSORT* yang menggunakan *Kalman Filter* dan *Hungarian Algorithm*, serta teknologi lainnya seperti *MediaPipe Pose*, semakin memperkuat kemampuan sistem kursi roda otonom dalam melacak dan mengikuti gerakan manusia dengan presisi tinggi.

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.

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.

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 da-

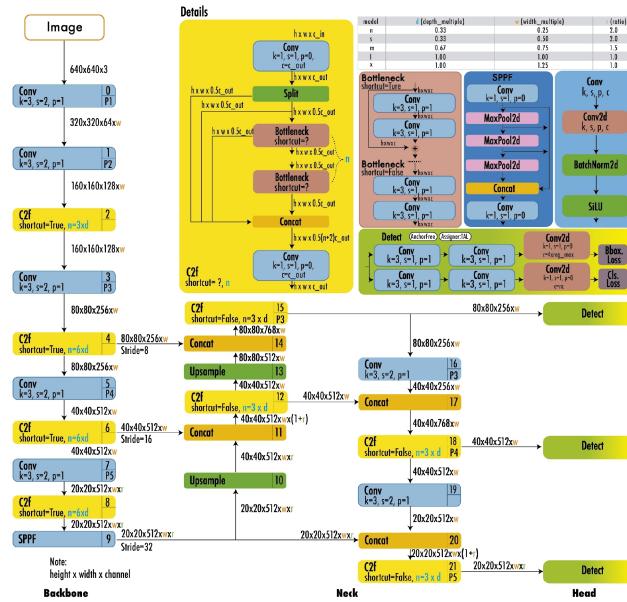
pat diikuti dengan akurasi tinggi. Dengan memanfaatkan *pose estimation*, sistem dapat memahami arah dan intensitas gerakan pengguna, yang memungkinkan kursi roda untuk merespons secara tepat dan efisien.

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 2.1: Arsitektur YOLOv8.

Gambar menunjukkan arsitektur lengkap *YOLOv8*, dimulai dari input gambar dengan resolusi $640 \times 640 \times 3$, yang diproses melalui serangkaian lapisan konvolusi dalam *Backbone* untuk

mengekstraksi fitur penting. Kemudian, fitur-fitur ini diteruskan melalui *Neck* yang menggabungkan dan mengolah informasi dari berbagai level resolusi, menghasilkan representasi multi-skala yang kaya. Akhirnya, *Head* bertanggung jawab untuk melakukan prediksi akhir, seperti *bounding boxes*, label kelas, dan *confidence score*.

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*.

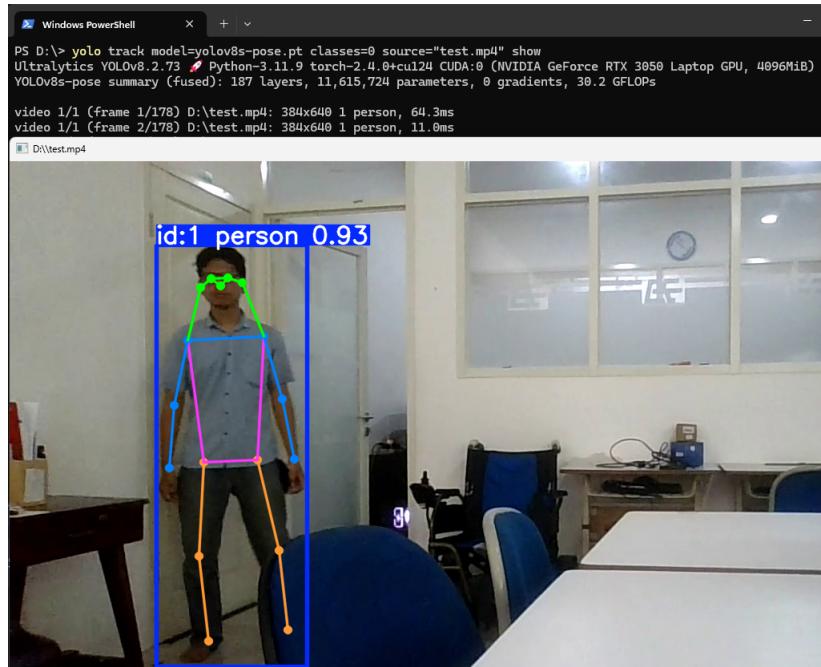
Fungsi *loss* dalam *YOLOv8* terdiri dari beberapa komponen utama yang mengukur perbedaan antara prediksi model dan *ground truth*, serta menyeimbangkan pentingnya prediksi koordinat, *confidence score*, dan klasifikasi objek. Rumus fungsi *loss*-nya adalah:

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

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).

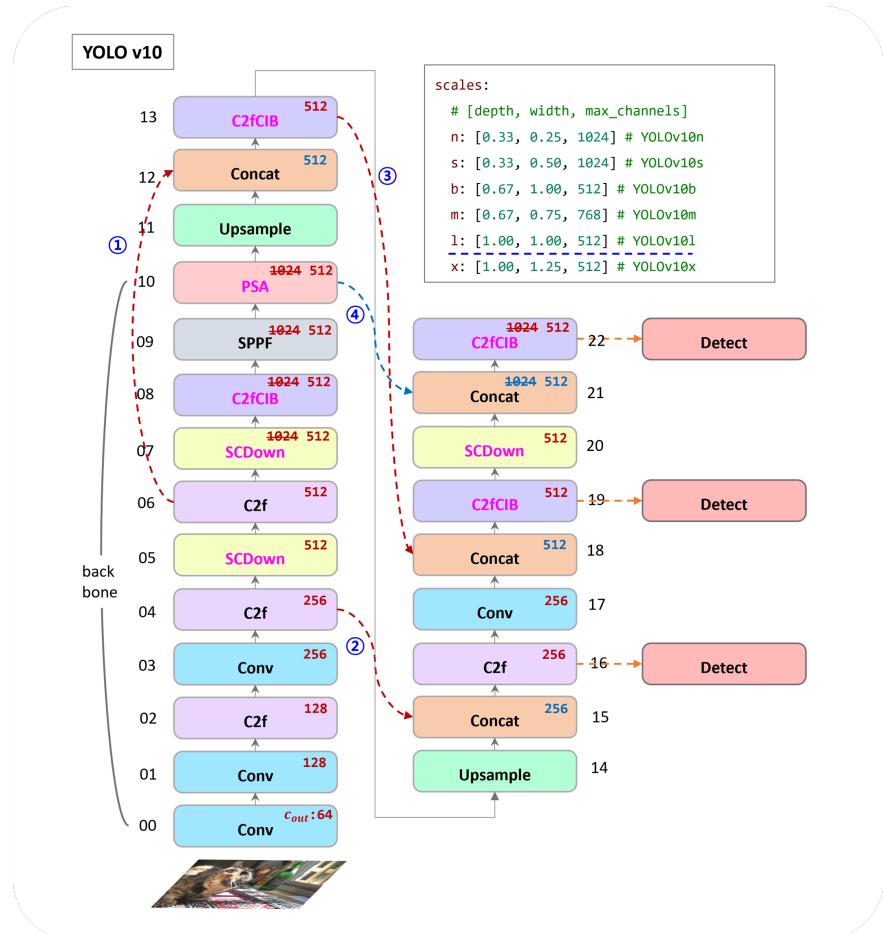


Gambar 2.2: YOLOv8 Pose.

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.

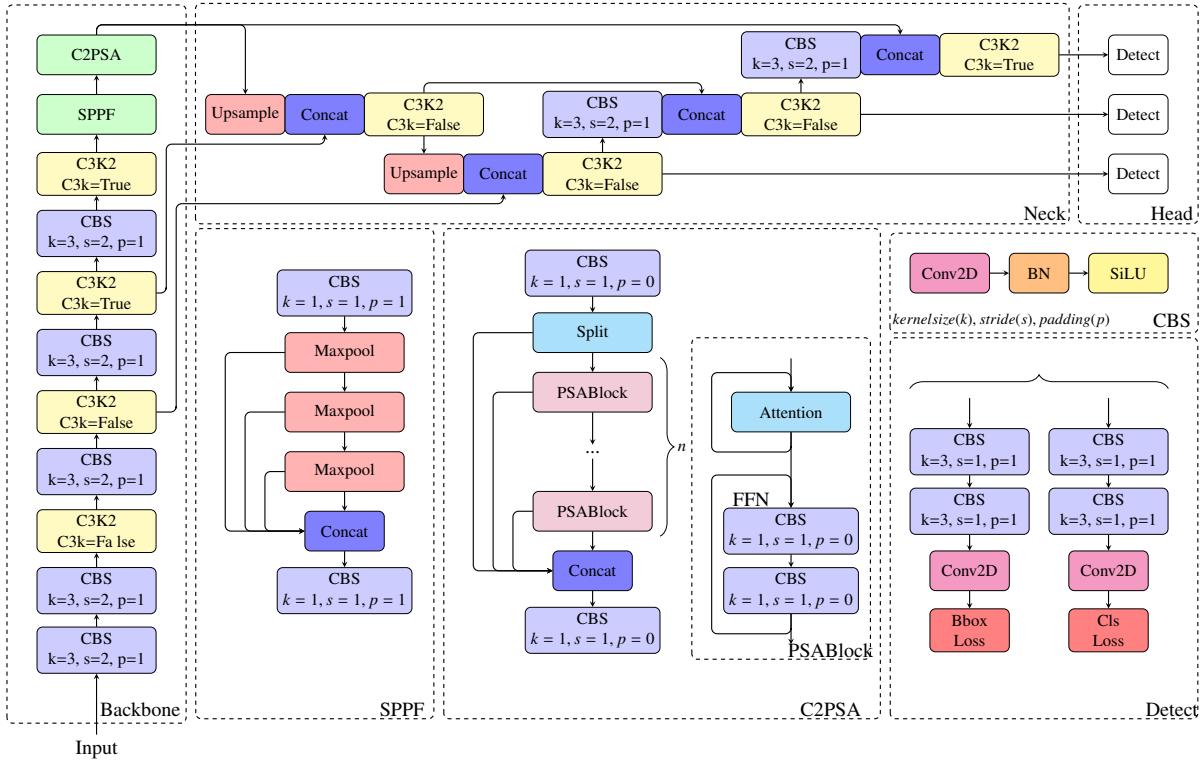


Gambar 2.3: Arsitektur YOLOv10.

2.5.3 YOLOv11

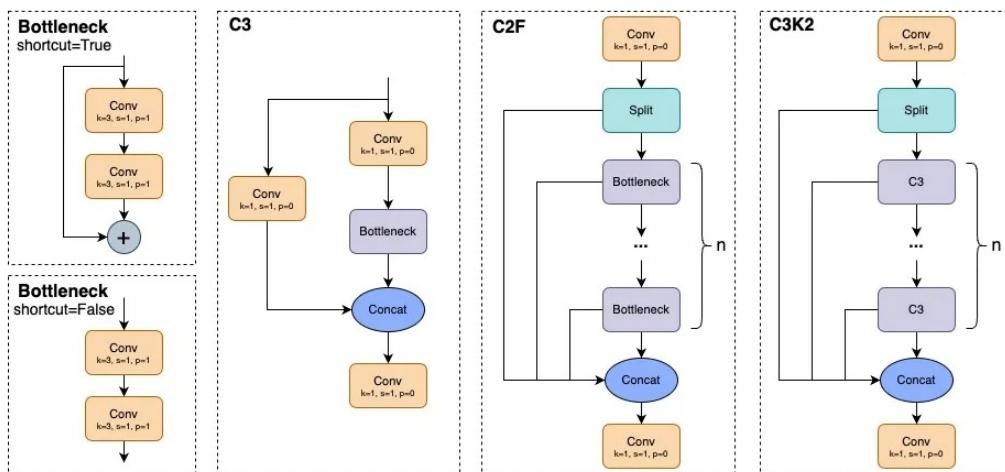
YOLOv11 adalah terobosan terbaru dalam seri detektor objek real-time dari Ultralytics. Meneruskan kemajuan dari pendahulunya, YOLOv11 menghadirkan peningkatan signifikan pada arsitekturnya, menjadikannya solusi yang kuat dan adaptif untuk berbagai aplikasi *computer vision*. Ultralytics memperkenalkan 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 2.4: 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 2.5: 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.

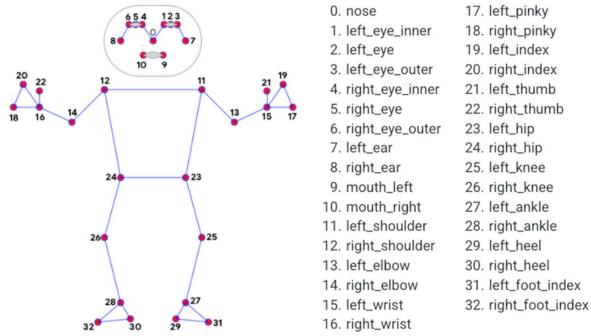


Gambar 2.6: MediaPipe 3D.

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 2.7: 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 2.8: 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 benar. 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*.

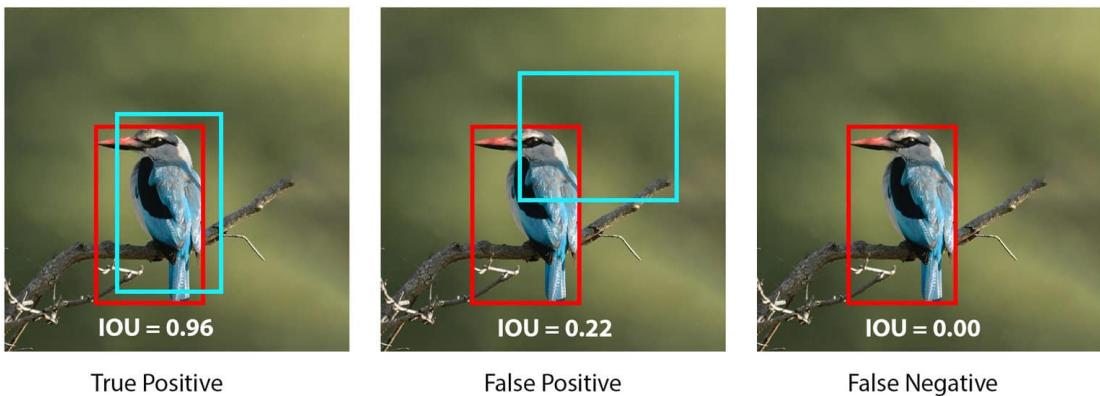
$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.9)$$

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 2.9: 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 [4]:

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 ($\dot{x}_c, \dot{y}_c, \dot{w}, \dot{h}$) [5], [6]. Vektor status ini didefinisikan sebagai:

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

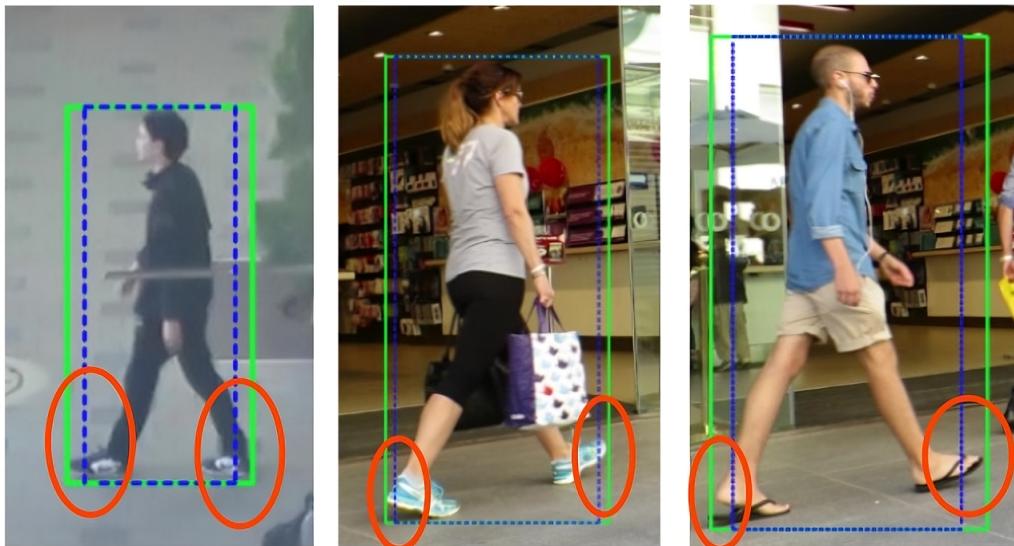
$$\mathbf{z}_k = [z_{x_c}(k), z_{y_c}(k), z_w(k), z_h(k)]^\top \quad (2.12)$$

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 2.10: Kalman Filter bbox.

Visualisasi bentuk *bounding box* dibandingkan dengan *Kalman filter* yang banyak digunakan [6] (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 [7], kemudian menggunakan sparse optical flow [8] 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 [9]. 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 [10], yang mencakup pergerakan kamera akibat manuver kendaraan yang berbelok ke arah kanan.



Gambar 2.11: 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 [11], [12], dan mekanisme exponential moving average (EMA) digunakan untuk memperbarui status appearance tracklet [13]. 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:

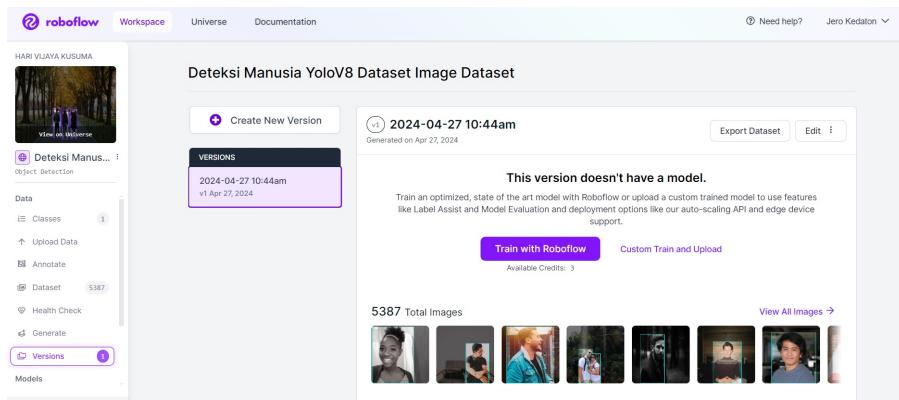
$$\hat{d}_{i,j}^{cos} = \begin{cases} 0.5 \cdot d_{i,j}^{cos}, (d_{i,j}^{cos} < \theta_{emb}) \wedge (d_{i,j}^{iou} < \theta_{iou}) \\ 1, \text{ otherwise} \end{cases} \quad C_{i,j} = \min\{d_{i,j}^{iou}, \hat{d}_{i,j}^{cos}\} \quad (2.21)$$

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

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 [14] 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 2.12: 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*

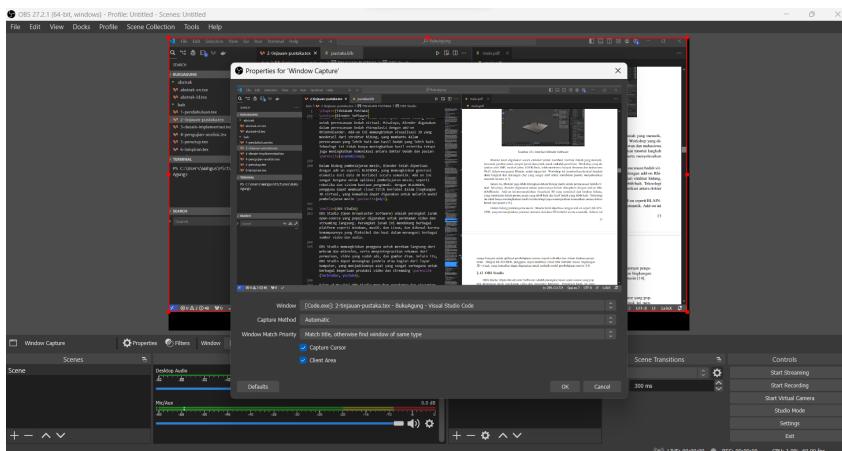
populer juga disediakan, memudahkan persiapan data untuk berbagai algoritma pembelajaran mesin. Selain itu, fungsi untuk membagi *dataset* menjadi set pelatihan, validasi, dan pengujian untuk validasi model.

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

- *Eksport Data*: *Dataset* dapat dengan mudah diekspor 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 2.13: Interface OBS Studio

OBS Studio adalah perangkat lunak *open-source* yang dirancang untuk streaming dan perekaman video. Dalam proyek ini, *OBS Studio* digunakan terutama untuk mendokumentasikan dan menganalisis performa kursi roda otonom selama fase pengujian dengan menggunakan *webcam* sebagai sumber utama input data. *Webcam* yang terhubung ke *OBS Studio* memungkinkan perekaman seluruh proses pengujian secara visual dan *real-time*. Penggunaan *webcam* sebagai input streaming memberikan opsi untuk melangsungkan data secara langsung menuju kursi roda otonom. Dengan demikian, seluruh proses pengujian dari awal hingga akhir dapat terdokumentasi secara akurat, sekaligus mendukung evaluasi dan pengambilan keputusan secara *real-time* berdasarkan data yang diterima.

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.

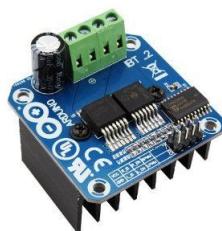


Gambar 2.14: Gambar ESP32 Devkit V1

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 2.15: 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 2.16: 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.

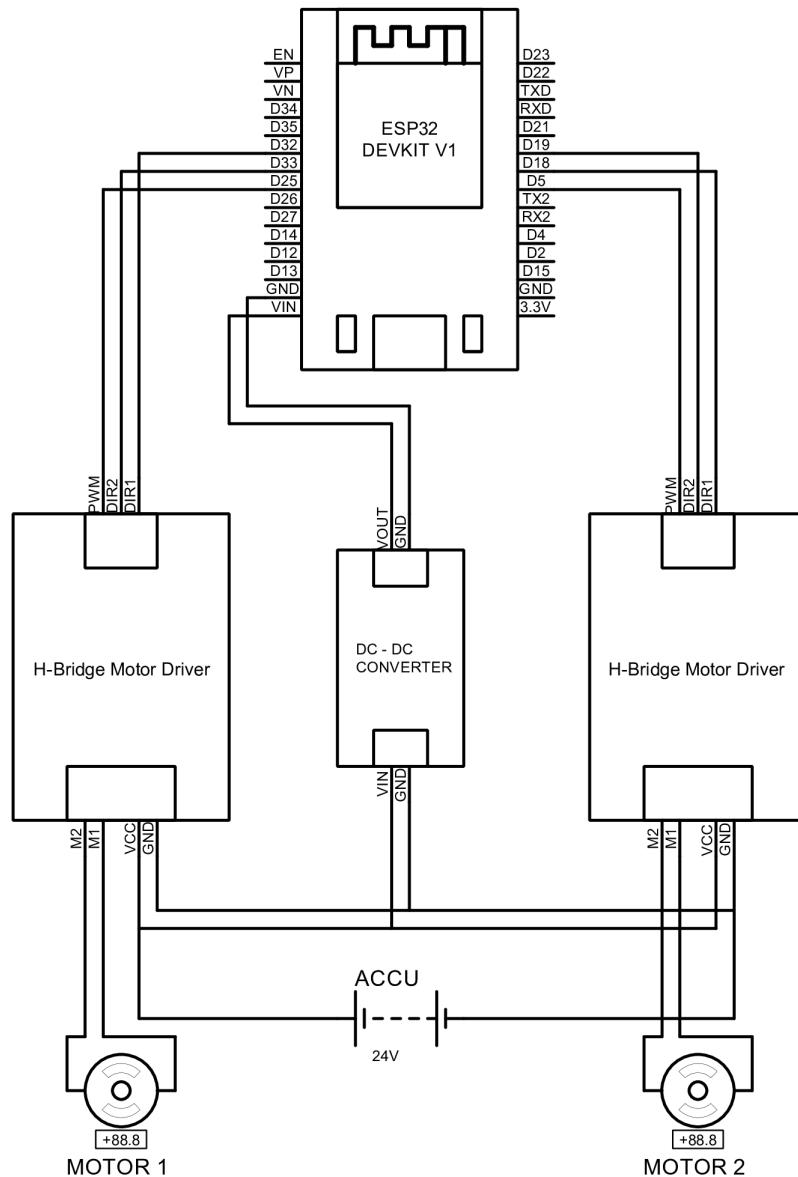
3.2.5 Skematik Alat

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.

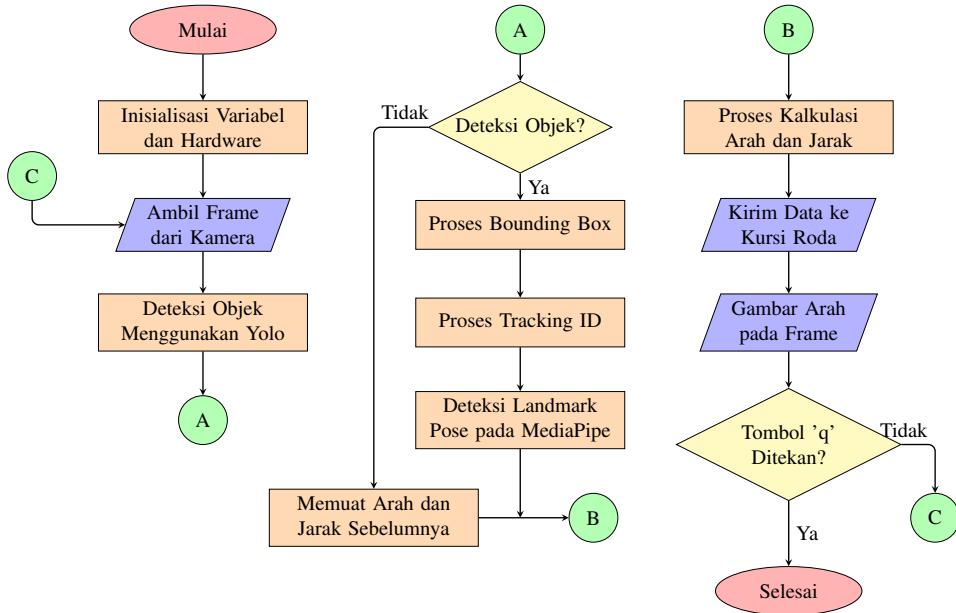
3.3.7 Kontrol Motor

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.

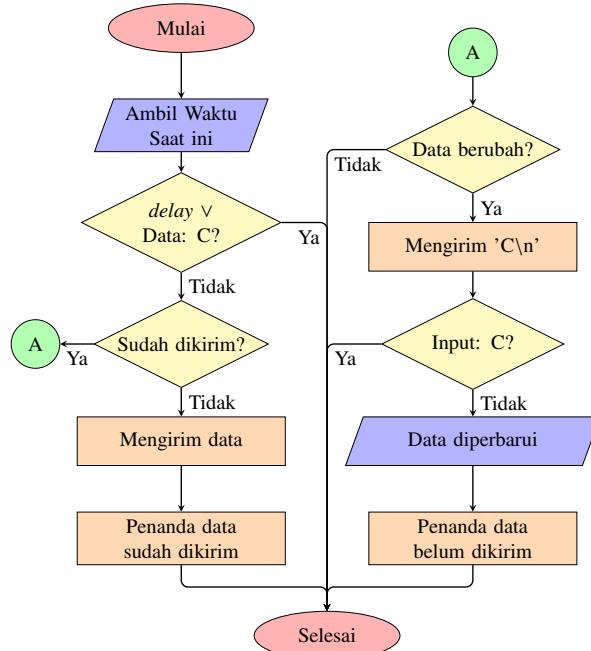
3.3.8 Kode Program

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.

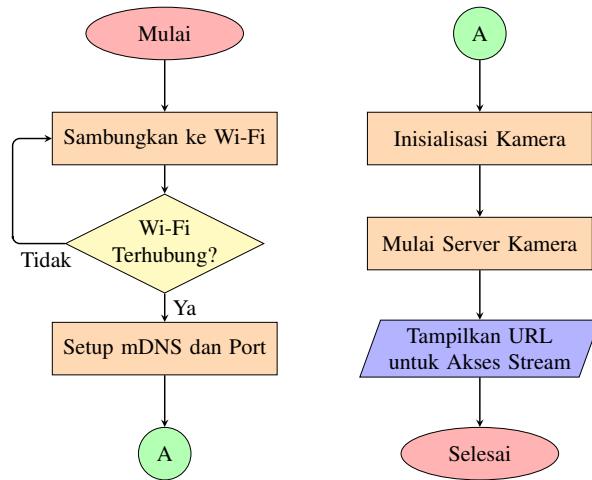


Gambar 3.3: Flowchart regulasi arah

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

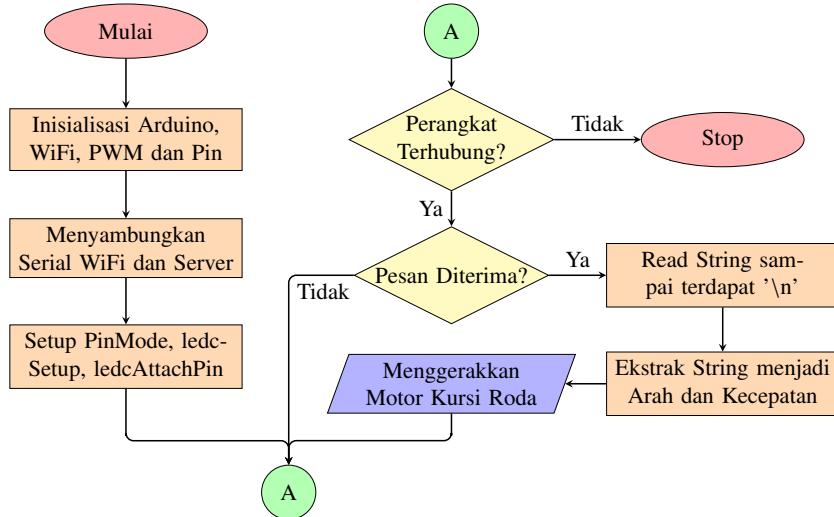
Sistem ini menerapkan protokol *mDNS (Multicast DNS)* untuk secara otomatis memberikan nama unik pada setiap kamera. Setiap kamera terhubung dengan nama berbeda, seperti *camera.local*, *camera2.local*, dan seterusnya. Alokasi port untuk streaming diatur secara otomatis, misalnya melalui port 81, 82, 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 penelitian ini dipaparkan skenario pengujian yang dilakukan, termasuk kondisi lingkungan pengujian, perangkat keras dan perangkat lunak yang digunakan, serta parameter-parameter yang diuji. Informasi detail mengenai konfigurasi dan prosedur pengujian disajikan agar hasil pengujian dapat direplikasi dengan konsisten.

4.1 Hasil Pengujian Performa Menggunakan Confusion Matrix

Bagian ini membahas hasil pengujian performa deteksi objek menggunakan Confusion Matrix. Nilai seperti True Positive, True Negative, False Positive, dan False Negative dianalisis untuk menilai akurasi dan efektivitas deteksi.

4.2 Pengujian Berdasarkan FPS

Pengujian ini dilakukan untuk menganalisis kecepatan pemrosesan sistem dalam satuan Frame per Second (FPS). FPS yang tinggi menunjukkan bahwa sistem dapat bekerja dengan baik secara real-time. Grafik hasil pengujian disertakan untuk memudahkan visualisasi performa.

4.3 Pengujian Berdasarkan Response Time

Bagian ini mengukur waktu yang dibutuhkan sistem untuk merespons setiap perubahan lingkungan atau perintah yang diterima. Response Time sangat penting untuk mengukur responsivitas sistem dalam skenario dinamis.

4.4 Performa Keberhasilan Tracking

Bagian ini mengevaluasi keberhasilan sistem dalam melakukan tracking terhadap objek target. Tingkat keberhasilan dianalisis untuk mengetahui keandalan sistem dalam berbagai skenario.

4.5 Pengujian Kesesuaian Jarak Deteksi

Pengujian dilakukan untuk mengukur kemampuan sistem dalam mendeteksi objek pada berbagai jarak dan menganalisis performa sistem pada jarak-jarak tersebut.

4.6 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.

4.6.1 Belok Kiri saat berada didalam frame

Bagian ini membahas kemampuan sistem dalam mengikuti objek saat berbelok ke kiri. Tantangan yang dihadapi dan solusi yang diterapkan diuraikan.

4.6.2 Belok Kiri saat berada diluar frame

Bagian ini membahas kemampuan sistem dalam mengikuti objek saat berbelok ke kiri. Tantangan yang dihadapi dan solusi yang diterapkan diuraikan.

4.6.3 Belok Kanan saat berada didalam frame

Bagian ini membahas kemampuan sistem dalam mengikuti objek saat berbelok ke kanan, mirip dengan bagian sebelumnya.

4.6.4 Belok Kanan saat berada diluar frame

Bagian ini membahas kemampuan sistem dalam mengikuti objek saat berbelok ke kanan, mirip dengan bagian sebelumnya.

4.7 Performa Keberhasilan Mengikuti

Bagian ini menguji apakah sistem dapat mengikuti objek pada lajur khusus, seperti jalur sempit atau berbelok yang membutuhkan manuver khusus.

4.8 Pembahasan Hasil

Bagian ini membahas hasil-hasil pengujian secara keseluruhan dan memberikan insight mengenai performa sistem.

4.8.1 Performa Deteksi Objek

Bagian ini menjelaskan kekuatan dan kelemahan deteksi objek yang ditemukan selama pengujian.

4.8.2 Kecepatan Pemrosesan (FPS)

Menganalisis kecepatan pemrosesan secara keseluruhan dan pengaruhnya terhadap performa sistem real-time.

4.8.3 Response Time

Membahas hasil pengujian response time dan faktor-faktor yang mempengaruhi waktu respons.

4.8.4 Performa Keberhasilan Tracking

Membahas tingkat keberhasilan sistem dalam melacak objek dan kondisi yang mempengaruhi performa.

4.8.5 Kesesuaian Jarak Deteksi

Diskusi tentang performa sistem dalam mendeteksi objek pada berbagai jarak yang telah diuji.

4.8.6 Performa Pergerakan Mengikuti Objek

Evaluasi akurasi dalam mengikuti objek target, termasuk kondisi yang dapat menyebabkan kegagalan.

4.8.7 Performa Keberhasilan Mengikuti

Menguraikan keberhasilan sistem dalam mengikuti objek saat menghadapi belokan dan lajur khusus.

[Halaman ini sengaja dikosongkan]

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian yang Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. sebagai berikut:

1. Pembuatan Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus.
2. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa.
3. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna.

5.2 Saran

Untuk pengembangan lebih lanjut pada Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. antara lain:

1. Memperbaiki Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus.
2. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa.
3. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna.

[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] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, *Bot-sort: Robust associations multi-pedestrian tracking*, 2022. arXiv: 2206.14651 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2206.14651>.
- [5] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *ICIP*, IEEE, 2016, pp. 3464–3468.
- [6] 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.
- [7] 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.
- [8] J.-Y. Bouguet, “Pyramidal implementation of the lucas kanade feature tracker,” 1999.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] H. Zhang, C. Wu, Z. Zhang, *et al.*, “Resnest: Split-attention networks,” *arXiv preprint arXiv:2004.08955*, 2020.
- [13] 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.
- [14] 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 time
6 import csv
7 from enum import Enum
8 from ultralytics import YOLO
9
10 source = "http://192.168.4.1:81/stream"
11 # source = 0
12 host = "192.168.4.1"
13 port = 80
14
15 # Initialize MediaPipe Pose
16 mp_pose = mp.solutions.pose
17 pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.7)
18 mirror = True
19
20 fp = 481 # Focal length in pixelsCV2
21 tb = 175 # Real-world height of the object in cm
22 k = 24.422 # Calibration factor for MediaPipe
23
24 # Load YOLO model
25 model = YOLO("yolov10m.pt")
26
27 def hitung_jarak(tinggi_bounding_box, focal_length_pixel, tinggi_objek_nyata):
28     if tinggi_bounding_box == 0:
29         return float('inf')
30     jarak = (tinggi_objek_nyata * focal_length_pixel) / tinggi_bounding_box
31     return jarak / 100 # Convert to meters
32
33 def hitung_jarak_euclidean(landmark1, landmark2, lebar_img):
34     return math.sqrt((landmark1.x - landmark2.x) ** 2 + (landmark1.y - landmark2.y) ** 2) * lebar_img
35
36 def hitung_lebar_mediapipe(pose_results, lebar_img):
37     if pose_results.pose_landmarks:
38         bahu_kiri = pose_results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_SHOULDER]
39         bahu_kanan = pose_results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_SHOULDER]
40         jarak_pix = hitung_jarak_euclidean(bahu_kiri, bahu_kanan, lebar_img)
41         return (k / jarak_pix) * 10
42     return 0
43
44 class SocketCommunicator:
```

```

45     def __init__(self, host, port):
46         self.socket = None
47         self.interval = 0
48         self.data = 'C\n'.encode()
49         self.sent = True
50         self.host = host
51         self.port = port
52         self.create_socket()
53
54     def create_socket(self):
55         print("Loading...", end="", flush=True)
56         s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
57         s.settimeout(1)
58         try:
59             s.connect((self.host, self.port))
60             print()
61             self.socket = s
62         except socket.error:
63             print(".", end="", flush=True)
64             time.sleep(1)
65
66     def send_data(self, data, ms=500):
67         current_time = time.time()
68         if self.socket:
69             if data != self.data:
70                 self.interval = time.time()
71                 self.socket.send('C\n'.encode())
72                 self.sent = False
73                 self.data = data
74             elif (current_time - self.interval) * 1000 < ms and not self.sent:
75                 self.socket.send(self.data.encode())
76                 self.sent = True
77
78 # Prepare CSV file
79 csv_file = open('log_belokan.csv', mode='w', newline='')
80 csv_writer = csv.writer(csv_file)
81 csv_writer.writerow(['Waktu', 'Inference Time', 'Jarak YOLO', 'Jarak ←
MediaPipe', 'Arah'])
82
83 # Initialize video capture
84 cap = cv2.VideoCapture(source)
85 # \
86
87 class Direction(Enum):
88     DIAM = 0
89     MAJU = 1
90     BELIKANAN = 2
91     BELIKIRI = 3
92
93 class PosisiManusiaTerakhir(Enum):
94     DIAM = 0
95     KIRI = 1
96     KANAN = 2
97     MAJU = 3
98
99 posisi_terakhir = PosisiManusiaTerakhir.DIAM

```

```

100
101 def draw_arrow(img, direction):
102     height, width, _ = img.shape
103     start_point = (width // 2, height - 50) # Start point of arrow
104     if direction == Direction.MAJU:
105         end_point = (width // 2, height - 150)
106     elif direction == Direction.BELIKANAN:
107         end_point = (width // 2 + 100, height - 100)
108     elif direction == Direction.BELIKIRI:
109         end_point = (width // 2 - 100, height - 100)
110     else:
111         end_point = start_point
112
113     color = (0, 255, 0)
114     thickness = 5
115     cv2.arrowedLine(img, start_point, end_point, color, thickness)
116
117
118 # Initialize the wheelchair controller
119 controller = SocketCommunicator(host, port)
120 controller.send_data('2\n') # Speed level .140
121
122 while True:
123     success, frame = cap.read()
124     if not success:
125         break
126
127     frame = cv2.rotate(frame, cv2.ROTATE_180)
128     img = cv2.flip(frame, 1) if mirror else frame
129
130     img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
131     height, width, _ = img.shape
132
133     # Define screen partitions
134     left_bound = width // 3
135     right_bound = 2 * (width // 3)
136
137     # Draw partition lines
138     cv2.line(img, (left_bound, 0), (left_bound, height), (0, 255, 0), 2)
139     cv2.line(img, (right_bound, 0), (right_bound, height), (0, 255, 0), ←
140                 2)
141
142     start_inference_time = time.time()
143     results = model.track(img, classes=0, stream=True, persist=True, conf←
144                           =0.7)
145     inference_time = time.time() - start_inference_time
146
147     arah = 'C\n'
148     arah_log = "Stop" # Default value
149     jarak_mediapipe = 0 # Default value
150     jarak_yolo = 0
151     direction = Direction.DIAM # Default direction
152
153     for r in results:
154         min_id = None
155         min_box = None
156         boxes = r.boxes

```

```

155     for box in boxes:
156         idex = box.id
157         if min_id is None or idex < min_id:
158             min_id = idex
159             min_box = box
160
161     if min_id:
162         x1, y1, x2, y2 = map(int, min_box.xyxy[0])
163         tinggi_bounding_box = y2 - y1
164
165         # Draw bounding box
166         cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
167         cv2.rectangle(img, (x1, y1), (x1+20, y1+30), (255, 0, 0), -1)
168         cv2.putText(img, str(int(min_id)), (x1, y1 + 25), cv2.←
169                     FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
170
171         # Calculate distance using YOLO
172         jarak_yolo = hitung_jarak(tinggi_bounding_box, fp, tb)
173
174         # Process with MediaPipe
175         person_img_rgb = cv2.cvtColor(img[y1:y2, x1:x2], cv2.←
176                                     COLOR_BGR2RGB)
177         pose_results = pose.process(person_img_rgb)
178
179         if pose_results.pose_landmarks:
180             # Draw pose landmarks
181             mp.solutions.drawing_utils.draw_landmarks(img[y1:y2, x1:←
182                                         x2], pose_results.pose_landmarks, mp_pose.←
183                                         POSE_CONNECTIONS)
184
185             lebar_img = person_img_rgb.shape[1]
186             jarak_mediapipe = hitung_lebar_mediapipe(pose_results, ←
187                                         lebar_img)
188
189             print(f"YOLO Distance: {jarak_yolo:.2f} m, MediaPipe ←
190                   Distance: {jarak_mediapipe:.2f} m")
191             cv2.putText(img, f"YOLO Distance: {jarak_yolo:.2f} m", ←
192                         (10, 180), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), ←
193                         2)
194             cv2.putText(img, f"MP Distance: {jarak_mediapipe:.2f} m", ←
195                         (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), ←
196                         2)
197
198             cp_Shape = lebar_img / img.shape[1]
199             print(cp_Shape)
200             center_x = (x1 + x2) // 2
201
202             if jarak_mediapipe > 1.0:
203                 if cp_Shape > 0.5 :
204                     print("Kursi roda berhenti")
205                     # arah = 'C\n'
206                     cv2.putText(img, "Kursi roda berhenti", (10, 50), ←
207                                 cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), ←
208                                 2)
209                     arah_log = "BatasDekat"
210                     direction = Direction.DIAM
211                     posisi_terakhir = PosisiManusiaTerakhir.DIAM

```

```

200         elif center_x > right_bound:
201             print("Kursi roda belok Kiri")
202             arah = 'A\n'
203             cv2.putText(img, "Kursi roda belok Kiri", (10, ←
204                         50), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, ←
205                         255), 2)
206             arah_log = "BelokKiri"
207             direction = Direction.BELIKIRI
208             posisi_terakhir = PosisiManusiaTerakhir.KIRI
209             elif center_x < left_bound:
210                 print("Kursi roda belok Kanan")
211                 arah = 'E\n'
212                 cv2.putText(img, "Kursi roda belok Kanan", (10, ←
213                         50), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, ←
214                         255), 2)
215                 arah_log = "BelokKanan"
216                 direction = Direction.BELIKANAN
217                 posisi_terakhir = PosisiManusiaTerakhir.KANAN
218             else:
219                 arah = 'B\n'
220                 print("Kursi roda Maju")
221                 cv2.putText(img, "Kursi roda Maju", (10, 50), ←
222                         cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
223                 arah_log = "Maju"
224                 direction = Direction.MAJU
225                 posisi_terakhir = PosisiManusiaTerakhir.MAJU
226                 # delay(200)
227             else:
228                 # delay(200)
229                 arah_log = "Stop"
230                 direction = Direction.MAJU
231             else:
232                 if posisi_terakhir == PosisiManusiaTerakhir.KIRI:
233                     print("Mencari manusia ke Kiri")
234                     arah = 'A\n'
235                     cv2.putText(img, "Mencari manusia ke Kiri", (10, 50), cv2←
236                         .FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
237                     direction = Direction.BELIKIRI
238                 elif posisi_terakhir == PosisiManusiaTerakhir.KANAN:
239                     print("Mencari manusia ke Kanan")
240                     arah = 'E\n'
241                     cv2.putText(img, "Mencari manusia ke Kanan", (10, 50), ←
242                         cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
243                     direction = Direction.BELIKANAN
244                 elif posisi_terakhir == PosisiManusiaTerakhir.MAJU:
245                     print("Mencari manusia ke Depan")
246                     arah = 'B\n'
247                     cv2.putText(img, "Mencari manusia ke Depan", (10, 50), ←
248                         cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
249                     direction = Direction.MAJU
250                 elif posisi_terakhir == PosisiManusiaTerakhir.DIAM:
251                     print("Menunggu jarak aman")
252                     # arah = 'C\n'
253                     cv2.putText(img, "Menunggu jarak aman", (10, 50), cv2.←
254                         FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
255                     direction = Direction.MAJU

```

```

248
249     # Write log to CSV
250     waktu = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
251     csv_writer.writerow([waktu, inference_time, jarak_yolo, ←
252         jarak_mediapipe, arah_log])
253     csv_file.flush() # Ensure data is written immediately
254
255     controller.send_data(arah)
256     draw_arrow(img, direction)
257
258     cv2.imshow('Webcam', img)
259     if cv2.waitKey(1) & 0xFF == ord('q'):
260         break
261
262 cap.release()
263 cv2.destroyAllWindows()
264 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 ←

```

```

        kebutuhan
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 *) &←
115                                              msg, sizeof(msg));
116                if (result == ESP_OK) {
117                    Serial.println("Data dikirim ke ESP B melalui ESP-NOW");
118                } else {
119                    Serial.print("Gagal mengirim data ke ESP B. Error code: ");
120                    Serial.println(result); // Tampilkan error code untuk ←
121                                            debugging
122                }
123            }
124            client.stop();
125            Serial.println("Client terputus");
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=←
137                               frame");
138
139     if(res != ESP_OK){
140         return res;
141     }
142
143     while(true){
144         fb = esp_camera_fb_get();
145         if (!fb) {
146             Serial.println("Camera capture failed");
147             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             "));
151         if (res == ESP_OK){
152             res = httpd_resp_send_chunk(req, "Content-Type: image/jpeg\r\n←
153                 r\n",
153                                         strlen("Content-Type: image/jpeg\r\n←
153                 r\n"));
154         }
155         if (res == ESP_OK){
156             res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, ←
156                 _jpg_buf_len);
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
187 void setup() {
188     Serial.begin(115200);
189     Serial.setDebugOutput(true);
190     Serial.println();
191
192     initCamera();
193
194     // Set up WiFi Access Point
195     WiFi.mode(WIFI_AP_STA);
196     WiFi.softAP(ssid, password);
197     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, ←
68     int data_len) {
69     char macStr[18]; // Buffer untuk menyimpan string MAC address
70     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 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

[Halaman ini sengaja dikosongkan]