



TUGAS AKHIR - EC234801

PENGEMBANGAN KURSI RODA OTONOM BERBASIS YOLOV8 UNTUK PENGHINDARAN OBSTACLE

I Gst Ngr Agung Hari Vijaya Kusuma
NRP 07211940000073

Dosen Pembimbing
Dr. Eko Mulyanto Yuniarno, S.T.,M.T.
NIP 19680601199512 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 - EC234801

**PENGEMBANGAN KURSI RODA OTONOM BERBASIS
YOLOV8 UNTUK PENGHINDARAN OBSTACLE**

I Gst Ngr Agung Hari Vijaya Kusuma
NRP 07211940000073

Dosen Pembimbing
Dr. Eko Mulyanto Yuniarno, S.T.,M.T.
NIP 19680601199512 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 - EC234801

Development of YOLOv8-based Autonomous Wheelchair for Obstacle Avoidance.

I Gst Ngr Agung Hari Vijaya Kusuma

NRP 07211940000073

Advisor

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

NIP 19680601199512 1 009

Undergraduate Study Program of Computer Engineering

Department of Computer Engineering

Faculty of Intelligence Electrics and Informatics Technology

Sepuluh Nopember Institute of Technology

Surabaya

2024

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENGEMBANGAN KURSI RODA OTONOM BERBASIS YOLOV8 UNTUK PENGHINDARAN OBSTACLE

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: **I Gst Ngr Agung Hari Vijaya Kusuma**
NRP. 07211940000073

Disetujui oleh Tim Penguji Tugas Akhir:

Dr. Eko Mulyanto Yuniarno, S.T.,M.T.
NIP: 19680601199512 1 009

(Pembimbing I)

.....

Wernher von Braun, S.T., M.T.
NIP:

(Pembimbing II)

.....

Dr. Galileo Galilei, S.T., M.Sc.
NIP:

(Penguji I)

.....

Friedrich Nietzsche, S.T., M.Sc.
NIP:

(Penguji II)

.....

Alan Turing, ST., MT.
NIP:

(Penguji III)

.....

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

Prof. Albus Percival Wulfric Brian Dumbledore, S.T., M.T.
NIP.

SURABAYA
Mei, 2024

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

Development of YOLOv8-based Autonomous Wheelchair for Obstacle Avoidance.

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 Intelligence Electrics and Informatics Technology
Sepuluh Nopember Institute of Technology

By: **I Gst Ngr Agung Hari Vijaya Kusuma**
NRP. 07211940000073

Approved by Final Project Examiner Team:

Dr. Eko Mulyanto Yuniarno, S.T.,M.T. (Advisor I)
NIP: 19680601199512 1 009

.....

Wernher von Braun, S.T., M.T. (Co-Advisor II)
NIP:

.....

Dr. Galileo Galilei, S.T., M.Sc. (Examiner I)
NIP:

.....

Friedrich Nietzsche, S.T., M.Sc. (Examiner II)
NIP:

.....

Alan Turing, ST., MT. (Examiner III)
NIP:

.....

Acknowledged,
Head of Computer Engineering Department ELECTICS - ITS

Prof. Albus Percival Wulfric Brian Dumbledore, S.T., M.T.
NIP.

SURABAYA
May, 2024

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa / NRP : I Gst Ngr Agung Hari Vijaya Kusuma / 0721194000073
Departemen : Teknik Komputer
Dosen Pembimbing / NIP : Dr. Eko Mulyanto Yuniarno, S.T.,M.T. / 19680601199512 1 009

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "PENGEMBANGAN KURSI RODA OTONOM BERBASIS YOLOV8 UNTUK PENGHINDARAN OBSTACLE" 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, May 2024

Mengetahui
Dosen Pembimbing

Mahasiswa

Dr. Eko Mulyanto Yuniarno, S.T.,M.T.
NIP. 19680601199512 1 009

I Gst Ngr Agung Hari Vijaya Kusuma
NRP. 0721194000073

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : I Gst Ngr Agung Hari Vijaya Kusuma / 07211940000073
Department : Computer Engineering
Advisor / NIP : Dr. Eko Mulyanto Yuniarno, S.T.,M.T. / 19680601199512 1 009

Hereby declared that the Final Project with the title of "*Development of YOLOv8-based Autonomous Wheelchair for Obstacle Avoidance.*" 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, May 2024

Acknowledged

Advisor

Student

Dr. Eko Mulyanto Yuniarno, S.T.,M.T.
NIP. 19680601199512 1 009

I Gst Ngr Agung Hari Vijaya Kusuma
NRP. 07211940000073

[Halaman ini sengaja dikosongkan]

ABSTRAK

Nama Mahasiswa : I Gst Ngr Agung Hari Vijaya Kusuma
Judul Tugas Akhir : PENGEMBANGAN KURSI RODA OTONOM BERBASIS *YOLOV8*
UNTUK PENGHINDARAN *OBSTACLE*
Pembimbing : 1. Dr. Eko Mulyanto Yuniarno, S.T.,M.T.
2. Wernher von Braun, S.T., M.T

Pengembangan kursi roda otonom telah menjadi semakin penting dalam memberikan mobilitas dan kemandirian yang ditingkatkan bagi individu dengan mobilitas terbatas. Studi ini mengusulkan pengembangan sistem kursi roda otonom berbasis YOLOv8 untuk menghindari obstacle, khususnya fokus pada deteksi obstacle manusia. Dengan memanfaatkan kemampuan deteksi objek yang canggih dari YOLOv8, sistem yang diusulkan bertujuan untuk mendeteksi dan menghindari obstacle manusia secara efektif. Sistem tersebut mendeteksi manusia melalui video menggunakan Single Based Computer Intel NUC dan Kamera. Obstacle yang dideteksi akan membuat SBC mengirim perintah ke ESP32 untuk menjalankan motor untuk melakukan manuver penghindaran. Evaluasi sistem meliputi pengujian komprehensif di berbagai lingkungan dan konteks navigasi untuk memvalidasi efektivitas dan kehandalan dalam aplikasi dunia nyata. Penelitian ini berkontribusi pada kemajuan teknologi kursi roda otonom, dan keselamatan bagi individu dengan keterbatasan mobilitas

Kata Kunci: Kursi Roda Otonom, *YOLOv8*, Intel NUC, ESP32, Deteksi Manusia, Bantuan Mobilitas

[Halaman ini sengaja dikosongkan]

ABSTRACT

*Name : I Gst Ngr Agung Hari Vijaya Kusuma
Title : Development of YOLOv8-based Autonomous Wheelchair for Obstacle Avoidance.
Advisors : 1. Dr. Eko Mulyanto Yuniarno, S.T.,M.T.
2. Wernher von Braun, S.T., M.T*

The development of autonomous wheelchairs has become increasingly vital in providing enhanced mobility and independence for individuals with limited mobility. This study proposes the development of an autonomous wheelchair system based on YOLOv8 for obstacle avoidance, specifically focusing on human obstacle detection. Utilizing the advanced object detection capabilities of YOLOv8, the proposed system aims to effectively detect and avoid human obstacles. The system detects humans through video using an Intel NUC Single Board Computer and camera. Detected obstacles trigger the SBC to send commands to the ESP32 to drive motors for evasion maneuvers. System evaluation includes comprehensive testing in various environments and navigation contexts to validate effectiveness and reliability in real-world applications. This research contributes to the advancement of autonomous wheelchair technology and safety for individuals with mobility limitations

Autonomous Wheelchair, YOLOv8, Intel NUC, ESP32, Human Detection, Mobility Aid.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kehadirat Tuhan Yang Maha Esa, atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penelitian ini yang berjudul "PENGEMBANGAN KURSI RODA OTONOM BERBASIS YOLOV8 UNTUK PENGHINDARAN OBSTACLE".

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

1. Bapak Dr.Supeno Mardi Susiko Nugroho, ST.,MT, selaku Kepala Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember
2. Bapak Dr. Eko Mulyanto Yuniarno, S.T., M.T. selaku Dosen Pembimbing telah memberikan arahan selama pengerjaan tugas akhir ini
3. Bapak/ibu selaku dosen penguji I, Bapak/ibu selaku dosen penguji II dan Bapak/ibu selaku dosen penguji III yang telah memberikan saran dan revisi agar pengerjaan Buku Tugas Akhir ini dapat menjadi lebih baik
4. Bapak-Ibu dosen pengajar Departemen Teknik Komputer, atas ilmu dan pengajaran yang telah diberikan kepada penulis selama ini
5. I Gusti Ngurah Bagus Kusuma Dewa, S.Si., Apt., MPPM. dan Ida Ayu Sekar Wathi, S.Si., Apt., M.Si. Ayah dan Ibu yang selalu mendukung dan membiayai saya dari kecil hingga sarjana.
6. Muhammad Khaeral Azzam dan I Putu Haris Setiadi Ekatama S.T, Teman seperjuangan yang telah membimbing saya secara teknis dan spiritual dalam pengerjaan tugas akhir ini.
7. Teman - teman lab B300 dan B201 serta teman - teman Departemen Teknik Komputer lainnya

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

Surabaya, Mei 2024

I Gst Ngr Agung Hari Vijaya Kusuma

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Permasalahan	1
1.3 Tujuan	2
1.4 Batasan Masalah atau Ruang Lingkup	2
1.5 Manfaat	2
2 TINJAUAN PUSTAKA	3
2.1 Hasil penelitian terdahulu	3
2.1.1 Deep Learning-Based Object Detection, Localisation and Tracking for Smart Wheelchair Healthcare Mobility	3
2.1.2 Deteksi Objek Menggunakan YOLOv3 Untuk Keamanan Pada Pergerakan Kursi Roda Elektrik	3
2.1.3 Deteksi Plat Nama Ruangan untuk Kendali Kursi Roda Pintar menggunakan YOLOv5 dan EasyOCR berbasis TX2	4
2.2 Object Detection	4
2.3 YOLO (<i>You Only Look Once</i>)	5
2.3.1 YOLOv8	5
2.4 Pose Estimation	6
2.5 MediaPipe	7
2.5.1 MediaPipe Pose	7
2.6 Classification Performance	8

2.7	Evaluation Metrics	8
2.8	Intersection over Union (IoU)	10
2.9	Single Board Computer (SBC)	11
2.10	Roboflow	12
2.11	Intel NUC	13
2.12	ESP32 Devkit V1	13
2.13	Motor Driver H-Bridge	14
2.14	Kursi Roda Elektrik KY-123	14
3	DESAIN DAN IMPLEMENTASI	17
3.1	Deskripsi Sistem	17
3.1.1	Kamera	17
3.1.2	Intel NUC	17
3.1.3	ESP32	18
3.1.4	Skematik Alat	19
3.2	Software	20
3.2.1	Pengumpulan Dataset citra	21
3.2.2	Labeling Menggunakan Roboflow	21
3.2.3	Object Detection YoloV8	24
3.2.4	Estimasi Pose MediaPipe	27
3.2.5	Perhitungan Rumus	28
3.2.6	Visualisasi hasil deteksi dalam Grid	31
3.2.7	Output Hasil deteksi	35
4	PENGUJIAN DAN ANALISIS	41
4.1	Pengujian	41
4.1.1	Hasil Pengujian Performa menggunakan YOLOV8	41
4.2	Deployment Pada Hardware	47
4.2.1	Performa Model Pada Intel NUC	47
4.2.2	Pengujian Berdasarkan FPS	48
4.2.3	Pengujian Berdasarkan Hasil Response Time	51
4.2.4	Performa Keberhasilan	53
5	PENUTUP	55
5.1	Kesimpulan	55
5.2	Saran	55

DAFTAR PUSTAKA **57**

BIOGRAFI PENULIS **67**

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

2.1	Arsitektur YOLOv8	5
2.2	Pose MediaPipe	8
2.3	Confusion Matrix	8
2.4	Perbandingan Performa Berdasarkan IoU	11
2.5	Interface Roboflow	12
2.6	Gambar Intel NUC	13
2.7	Gambar ESP32 Devkit V1	14
2.8	Gambar Motor Driver H-Bridge	14
2.9	Gambar Kursi Roda Elektrik KY-123	15
3.1	Blok Diagram Hardware	17
3.2	Skematik kontrol motor kursi roda	19
3.3	Flowchart kontrol kursi roda	20
3.4	Diagram Blok Software	21
3.5	Contoh Hasil data dari Citra.	21
3.6	Contoh upload dataset ke roboflow.	22
3.7	Contoh anotasi dataset ke roboflow.	22
3.8	Contoh Dataset.	23
3.9	Contoh Augmentasi Dataset.	24
3.10	Contoh Hasil Deteksi Menggunakan Model YoloV8 yang telah di train.	25
3.11	Contoh Confusion Matrix model.	25
3.12	Contoh grafik loss dan mAP model.	26
3.13	Contoh hasil deteksi pose.	27
3.14	Contoh hasil deteksi pose seluruh tubuh.	28
3.15	Contoh Penerapan hasil perhitungan jarak dan pemetaan grid.	29
3.16	Grid 10x10.	32
3.17	Parameter untuk setiap kotak grid.	32
3.18	Posisi Kursi Pada grid.	33
3.19	Kategori Posisi berdasarkan Index.	33
3.20	Variabel Dalam perpindahan Posisi terhadap Grid.	34
3.21	Lebar Objek dalam grid.	34

3.22 Contoh valid object diatas 0.8 Meter.	35
3.23 Contoh valid object dibawah 0.8 Meter.	36
3.24 Contoh lain valid object dibawah 0.8 Meter condong kiri.	36
3.25 Contoh lain valid object dibawah 0.8 Meter condong kanan.	37
3.26 Contoh kondisi Near Object Index Kiri >Kanan	37
3.27 Contoh kondisi Near Object Index Kanan >kiri	38
3.28 Contoh kondisi Near Object Liniear.	39
4.1 Visualisasi Hasil training	42
4.2 Confusion Matrix Hasil Training	43
4.3 Tes Inferensi Menggunakan Model Terlatih	43
4.4 Tes Inferensi Menggunakan Model Terlatih	44
4.5 Visualisasi Hasil training	45
4.6 Confusion Matrix Hasil Training	45
4.7 Tes Inferensi Menggunakan Model Terlatih	46
4.8 Tes Inferensi Menggunakan Model Terlatih	46

DAFTAR TABEL

3.1	Kode instruksi dari hasil klasifikasi	18
3.2	Tabel Keypoint yang digunakan	27
4.1	Spesifikasi Laptop	49
4.2	Tabel FPS pada Laptop	50
4.3	Spesifikasi NUC	50
4.4	Tabel FPS pada NUC	51
4.5	Hasil Pengujian Response Time	53

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

1.1 Latar Belakang

Lumpuh, sebagaimana didefinisikan oleh Kamus Besar Bahasa Indonesia, adalah keadaan di mana fungsi anggota badan melemah sehingga tidak bertenaga atau tidak dapat digerakkan lagi sebagaimana mestinya (Daring, 2016). Otot, tulang, saraf, dan jaringan penghubung antara ketiganya memiliki peran yang krusial dalam mengendalikan gerak tubuh manusia. Gangguan pada salah satu dari komponen ini bisa menyebabkan kelumpuhan, baik yang bersifat sementara maupun permanen.

Beberapa penyakit dan kondisi dapat memicu kelumpuhan, termasuk stroke yang menyebabkan kelumpuhan pada bagian wajah, lengan, dan kaki sebelah, *Bell's Palsy* yang menyebabkan kelumpuhan pada satu sisi wajah tanpa melibatkan anggota tubuh lainnya, trauma kepala yang menyebabkan kelumpuhan pada area tubuh yang sesuai dengan kerusakan otak, serta polio yang menyerang lengan, kaki, dan otot pernapasan, di antara banyak kondisi lain yang dapat memicu kelumpuhan (Pansawira, 2022).

Individu yang mengalami kelumpuhan sering kali menghadapi tantangan mobilitas dalam kehidupan sehari-hari. Mereka umumnya membutuhkan alat bantu, seperti kursi roda, untuk beraktivitas. Saat ini, kursi roda elektrik yang dioperasikan dengan *joystick* telah tersedia, namun alat ini sering kali tidak memadai untuk orang yang mengalami kelumpuhan lengan (Choi et al., 2019).

Pengembangan kursi roda otonom menjadi langkah vital untuk meningkatkan kemandirian dan kualitas hidup individu dengan keterbatasan mobilitas. Dengan perkembangan teknologi sensor dan pemrosesan gambar, aplikasi metode deteksi objek seperti YOLO (You Only Look Once) menjadi inti dari inovasi solusi mobilitas otonom. Penelitian oleh Lecrosnier mengenai aplikasi YOLOv3 dalam pengembangan kursi roda otonom untuk menghindari pintu menujukkan keberhasilan metode ini dalam memperbaiki navigasi dan keamanan pengguna (Lecrosnier et al., 2021).

YOLOv8, dengan kemampuan deteksi objek yang ditingkatkan dan akurasi yang tinggi dalam real-time, menawarkan kemungkinan besar untuk pengembangan lebih lanjut dalam kursi roda otonom. Oleh karena itu, penelitian dengan judul "Pengembangan Kursi Roda Otonom Berbasis YOLOv8 untuk Penghindaran Obstacle" diharapkan bisa mengurangi risiko kecelakaan yang berhubungan dengan kursi roda, memberikan solusi efektif untuk navigasi dan keamanan.

1.2 Permasalahan

Berdasarkan latar belakang yang telah dipaparkan, agar kursi roda otonom dapat menghindari obstacle, maka diperlukan suatu sistem yang dapat mendeteksi manusia.

1.3 Tujuan

Tujuan dari Tugas Akhir ini ialah untuk mengembangkan sistem yang dapat mendeteksi manusia pada kursi roda otonom untuk menghindari Obstacle.

1.4 Batasan Masalah atau Ruang Lingkup

Terdapat beberapa Batasan masalah untuk memperjelas penelitian yang dilakukan. Batasan batasannya adalah sebagai berikut :

1. Penelitian ini terbatas pada pengembangan system deteksi dan kendali untuk kursi roda berbasis computer vision dengan focus pada mendeteksi keberadaan manusia di sekitarnya.
2. System yang dikembangkan akan menggunakan deteksi objek YOLOv8 (You Only Look Once) untuk mendeteksi manusia dalam gambar.
3. Pengontrol kursi roda akan diimplementasikan untuk memberikan respon berbelok secara otomatis Ketika mendeteksi keberadaan manusia yang diam di dekatnya.
4. Penelitian ini tidak mencakup pengembangan perangkat keras kendali kursi roda, melainkan hanya focus pada pengembangan algoritma perangkat lunak untuk pengendalian otomatis.
5. Mediapipe tidak dilakukan *training* namun, digunakan untuk pengambilan landmark pada tubuh manusia.
6. Evaluasi kinerja system akan dilakukan melalui pengujian yang menggunakan dataset gambar yang telah dilabeli dengan benar, dengan focus pada metrik precision, recall, dan mean average precision (mAP)

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

2.1.1 Deep Learning-Based Object Detection, Localisation and Tracking for Smart Wheelchair Healthcare Mobility

Pada 24 December 2020 telah dilakukan penelitian mengenai “Deep Learning-Based Object Detection, Localisation and Tracking for Smart Wheelchair Healthcare Mobility” oleh Lecrosnier, L. (Lecrosnier et al., 2021)

Penelitian ini bertujuan untuk mengembangkan Sistem Bantuan Pengemudi Lanjutan untuk kursi roda listrik pintar guna meningkatkan kemandirian orang-orang dengan disabilitas. Penelitian ini berfokus pada deteksi, lokalisasi, dan pelacakan objek di dalam ruangan untuk kursi roda, khususnya pintu dan pegangan pintu.

Tujuan utama dari penelitian ini adalah meningkatkan kemampuan kursi roda otomatis, yang memungkinkan deteksi objek-objek tersebut di sekitarnya secara tepat. Langkah pertama dalam penelitian ini adalah mengadaptasi algoritma deteksi objek YOLOv3 ke dalam kasus penggunaannya. Selanjutnya, penelitian ini menggunakan kamera Intel RealSense untuk melakukan estimasi kedalaman. Terakhir, sebagai langkah ketiga dan terakhir, penelitian ini menggunakan pendekatan pelacakan objek 3D berbasis algoritma SORT.

Untuk memvalidasi semua pengembangan tersebut, penelitian dilakukan dengan melakukan berbagai eksperimen di lingkungan dalam yang terkontrol. Deteksi, estimasi jarak, dan pelacakan objek diuji menggunakan dataset yang disiapkan sendiri, yang mencakup pintu dan pegangan pintu. Dengan demikian, penelitian ini bertujuan untuk meningkatkan kemampuan kursi roda pintar dalam memahami dan berinteraksi dengan lingkungannya, khususnya dalam mengenali objek-objek penting seperti pintu dan pegangannya. (Lecrosnier et al., 2021)

2.1.2 Deteksi Objek Menggunakan YOLOv3 Untuk Keamanan Pada Pergerakan Kursi Roda Elektrik

Pada 12 Desember 2022 telah dilakukan penelitian mengenai “Deteksi Objek Menggunakan YOLOv3 Untuk Keamanan Pada Pergerakan Kursi Roda Elektrik” oleh Wahyu Krisna Wijaya dan I Komang Somawirata. (Wijaya et al., 2022)

Penelitian ini bertujuan untuk mengembangkan sistem pengolahan citra menggunakan algoritma You Only Look Once (YOLO), untuk meningkatkan keamanan pergerakan kursi roda elektrik. Sistem ini dirancang untuk mendeteksi objek di depan kursi roda, termasuk halangan di permukaan jalan dan objek yang dapat menghalangi pergerakan kursi roda tersebut contohnya seperti meja, kursi dan barang - barang yang ada di lingkungan sehari-hari.

Dengan menggunakan teknologi pengolahan citra, sistem ini dapat secara otomatis mendeteksi objek-objek tersebut saat kursi roda bergerak. Hasil pendekripsi tersebut kemudian ditampilkan pada layar monitor di depan pengemudi, memungkinkan mereka untuk merespons

dengan cepat terhadap potensi bahaya di jalur perjalanan.

Tujuan utama dari penelitian ini adalah untuk meningkatkan keamanan pengguna kursi roda elektrik dengan fokus pada pendekripsi objek di sekitar kursi roda. Dengan demikian, sistem ini memberikan kontribusi dalam mendukung sistem keamanan kursi roda elektrik, khususnya dalam menghadapi situasi di mana ada halangan di depan atau di sekitar kursi roda yang dapat membahayakan pengguna.

2.1.3 Deteksi Plat Nama Ruangan untuk Kendali Kursi Roda Pintar menggunakan YOLOv5 dan EasyOCR berbasis TX2

Pada 2 Maret 2023 telah dilakukan penelitian mengenai "Deteksi Plat Nama Ruangan untuk Kendali Kursi Roda Pintar menggunakan YOLOv5 dan EasyOCR berbasis TX2" oleh Muhammad Fadhel Haidar dan Fitri Utaminingrum.

Penelitian ini bertujuan untuk Mengembangkan sistem deteksi nama ruangan otomatis untuk kursi roda pintar untuk membantu penyandang disabilitas mendapatkan mobilitas yang lebih baik. Menggunakan kamera, sistem ini dapat mengenali plat nama ruangan melalui metode YOLOv5 dan EasyOCR.

Penelitian ini menunjukkan bahwa metode YOLOv5 memiliki akurasi 60% dalam mendekripsi nama ruangan, sementara EasyOCR mencapai 100%. Hasil ini memvalidasi kemungkinan integrasi sistem deteksi objek dan pengenalan karakter untuk meningkatkan navigasi kursi roda otonom.

Kesimpulan dari penelitian tersebut adalah bahwa sistem deteksi nama ruangan otomatis pada kursi roda pintar yang menggunakan metode YOLOv5 dan EasyOCR terbukti efektif dalam membantu navigasi penyandang disabilitas. Meskipun metode YOLOv5 menunjukkan akurasi yang lebih rendah (60%) dibandingkan EasyOCR (100%), integrasi kedua metode tersebut meningkatkan kemampuan kursi roda untuk bergerak secara otonom menuju tujuan yang diinginkan berdasarkan pengenalan plat nama ruangan. Ini membuka peluang untuk pengembangan lebih lanjut dalam teknologi bantu mobilitas bagi penyandang disabilitas.

2.2 Object Detection

Deteksi objek secara real-time telah muncul sebagai komponen kritis dalam berbagai aplikasi, meliputi berbagai bidang seperti kendaraan otonom, robotika, pengawasan video, dan realitas tertambah. Di antara berbagai algoritma deteksi objek, kerangka kerja YOLO (*You Only Look Once*) telah menonjol karena keseimbangan kecepatan dan akurasi yang luar biasa, memungkinkan identifikasi objek yang cepat dan dapat diandalkan dalam gambar. Sejak diperkenalkan, keluarga YOLO telah berkembang melalui beberapa iterasi, setiap versi membangun atas versi sebelumnya untuk mengatasi keterbatasan dan meningkatkan kinerja.

Selain kerangka kerja YOLO, bidang deteksi objek dan pemrosesan gambar telah mengembangkan beberapa metode lain yang terkenal. Teknik seperti R-CNN (*Region-based Convolutional Neural Networks*) dan penerusnya, Fast R-CNN dan Faster R-CNN, telah memainkan peran penting dalam memajukan akurasi deteksi objek. Metode ini mengandalkan proses dua tahap, di mana pencarian selektif menghasilkan usulan wilayah, dan jaringan saraf konvolisional mengklasifikasi dan menyempurnakan wilayah-wilayah ini. Pendekatan signifikan lainnya adalah Single-Shot MultiBox Detector (SSD), yang, serupa dengan YOLO, berfokus pada kecepatan dan efisiensi dengan menghilangkan kebutuhan untuk langkah usulan wilayah ter-

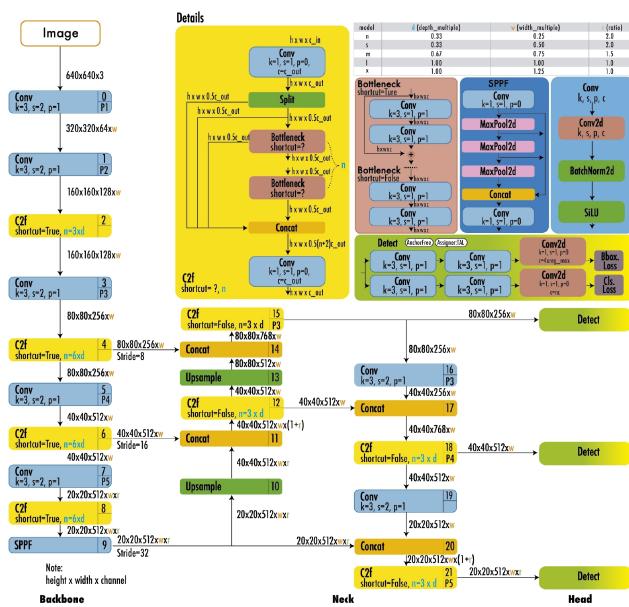
pisah. Selain itu, metode seperti Mask R-NN telah memperluas kemampuan ke segmentasi instans, memungkinkan lokalisasi objek yang tepat dan segmentasi tingkat piksel. Perkembangan ini, bersama dengan lainnya seperti RetinaNet dan EfficientDet, telah secara kolektif berkontribusi pada lanskap beragam algoritma deteksi objek. Setiap metode menawarkan kompromi unik antara kecepatan, akurasi, dan kompleksitas, melayani kebutuhan aplikasi yang berbeda dan kendala komputasi.(Terven et al., 2023)

2.3 YOLO (*You Only Look Once*)

YOLO oleh Redmon et al., 2016. Joseph Redmon untuk pertama kalinya memperkenalkan pendekatan *End to end* pada deteksi objek secara *Real Time*. Nama YOLO, yang merupakan singkatan dari "*You Only Look Once*" (Anda Hanya Melihat Sekali), merujuk pada kemampuannya untuk menyelesaikan tugas deteksi dengan hanya satu kali laluan jaringan, berbeda dengan pendekatan sebelumnya yang menggunakan jendela geser diikuti oleh pengklasifikasi yang harus dijalankan ratusan atau ribuan kali per gambar atau metode yang lebih canggih yang membagi tugas menjadi dua langkah, di mana langkah pertama mendekripsi kemungkinan daerah dengan objek atau proposal daerah dan langkah kedua menjalankan pengklasifikasi pada proposal tersebut. Selain itu, YOLO menggunakan keluaran yang lebih sederhana berdasarkan hanya regresi untuk memprediksi keluaran deteksi sebagai lawan dari Fast R-CNN yang menggunakan dua keluaran terpisah, sebuah klasifikasi untuk probabilitas dan regresi untuk *box* koordinat(Redmon et al., 2016)

2.3.1 YOLOv8

YOLOv8 diluncurkan pada Januari 2023 oleh *Ultralytics*, perusahaan yang mengembangkan YOLOv5. YOLOv8 menyediakan lima versi skala: YOLOv8n (nano), YOLOv8s (kecil), YOLOv8m (sedang), YOLOv8l (besar), dan YOLOv8x (ekstra besar). YOLOv8 mendukung berbagai tugas visi seperti deteksi objek, segmentasi, estimasi pose, pelacakan, dan klasifikasi.(Terven et al., 2023)



Gambar 2.1: Arsitektur YOLOv8.

Pada Gambar 2.1 merupakan gambar detail Arsitektur Yolov8.YOLOv8 menggunakan backbone yang serupa dengan YOLOv5 dengan beberapa perubahan pada CSPLayer, yang kini disebut modul C2f. Modul C2f (*cross-stage partial bottleneck with two convolutions*) menggabungkan fitur tingkat tinggi dengan informasi kontekstual untuk meningkatkan akurasi deteksi.

YOLOv8 menggunakan model *anchor-free* dengan *decoupled head* untuk memproses tugas klasifikasi , dan regresi secara independen. Desain ini memungkinkan setiap cabang untuk fokus pada tugasnya dan meningkatkan akurasi model secara keseluruhan. Di output layer dari YOLOv8, mereka menggunakan fungsi aktivasi sigmoid untuk objectness score, yang mewakili probabilitas bahwa *bounding box* mengandung suatu objek. Model ini menggunakan fungsi softmax untuk *class probabilities*, yang mewakili probabilitas objek yang termasuk dalam setiap kelas yang mungkin.

YOLOv8 menggunakan *loss functions* CIoU dan DFL untuk *bounding box loss* dan *binary cross-entropy* untuk *classification loss*. Loss ini telah meningkatkan kinerja *object detection*, terutama saat berhadapan dengan objek yang lebih kecil.

YOLOv8 juga menyediakan semantic *segmentation model* yang disebut YOLOv8-Seg model. Backbone adalah CSPDarknet53 *feature extractor*, diikuti oleh modul C2f bukan traditional YOLO neck architecture. Modul C2f diikuti oleh dua segmentation heads, yang belajar memprediksi semantic *segmentation masks* untuk input image. Model ini memiliki *detection heads* yang mirip dengan YOLOv8, terdiri dari lima detection modules dan prediction layer. YOLOv8-Seg model telah mencapai *state-of-the-art results* pada berbagai *object detection* dan *semantic segmentation benchmarks* sambil menjaga *high speed* dan efisiensi.

YOLOv8 dapat dijalankan dari *command line interface* (CLI), atau juga dapat dipasang sebagai PIP package. Selain itu, dilengkapi dengan berbagai integrations untuk labeling, training, dan deploying.

Nilai Evaluasi pada MS COCO dataset test-dev 2017, YOLOv8x mencapai AP sebesar 53.9% dengan image size 640 pixels (dibandingkan dengan 50.7% dari YOLOv5 pada size input yang sama) dengan speed 280 FPS pada NVIDIA A100 dan TensorRT.

2.4 Pose Estimation

Estimasi pose adalah tugas menggunakan model pembelajaran mesin (*ML*) untuk memperkirakan pose seseorang dari gambar atau video dengan mengestimasi lokasi spasial sendi tubuh utama (titik kunci atau *keypoints*). Estimasi pose merujuk pada teknik visi komputer yang mendeteksi sosok manusia dalam gambar dan video, sehingga seseorang dapat menentukan, misalnya, di mana siku seseorang muncul dalam gambar. Penting untuk menyadari bahwa estimasi pose hanya memperkirakan di mana sendi tubuh kunci berada dan tidak mengenali siapa yang ada dalam gambar atau video.(Martín Abadi et al., 2015)

Model estimasi pose mengambil gambar kamera yang telah diproses sebagai input dan mengeluarkan informasi tentang titik kunci. Titik kunci yang terdeteksi diindeks oleh ID bagian, dengan skor kepercayaan antara 0,0 dan 1,0. Skor kepercayaan menunjukkan probabilitas bahwa titik kunci ada di posisi tersebut.(Martín Abadi et al., 2015)

2.5 MediaPipe

MediaPipe adalah kerangka kerja untuk membangun pipa yang melakukan inferensi dari data sensorik apa pun. Dengan MediaPipe, pipa persepsi dapat dibangun sebagai graf dari komponen modular, termasuk inferensi model, algoritma pengolahan media, dan transformasi data, dll. Data sensorik seperti aliran audio dan video memasuki graf, dan deskripsi yang dirasakan seperti aliran lokalizasi objek dan landmark wajah keluar dari graf. MediaPipe dirancang untuk praktisi pembelajaran mesin (ML) (*machine learning*), termasuk peneliti, mahasiswa, dan pengembang perangkat lunak, yang mengimplementasikan aplikasi ML yang siap produksi, menerbitkan kode yang menyertai karya penelitian, dan membangun prototipe teknologi. Kasus penggunaan utama untuk MediaPipe adalah prototipe cepat dari pipa persepsi dengan model inferensi dan komponen yang dapat digunakan kembali lainnya. MediaPipe juga memfasilitasi penyebaran teknologi persepsi ke dalam demo dan aplikasi di berbagai platform perangkat keras yang berbeda. MediaPipe memungkinkan peningkatan bertahap pada pipa persepsi melalui bahasa konfigurasi yang kaya dan alat evaluasi. (Lugaresi et al., 2019)

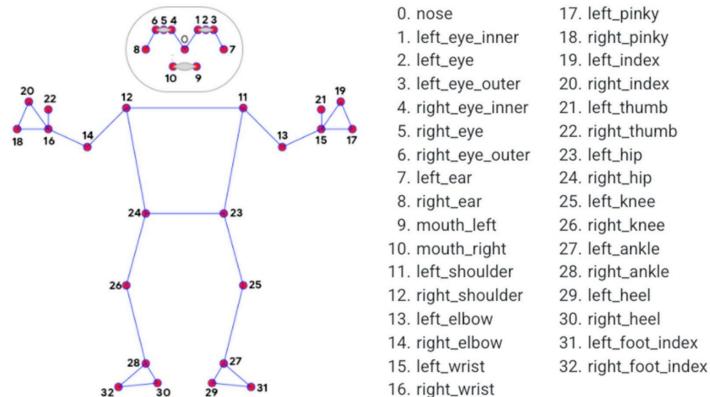
Memodifikasi aplikasi persepsi untuk menggabungkan langkah-langkah pemrosesan tambahan atau model inferensi dapat sulit, karena penggabungan berlebihan antar langkah. Selain itu, mengembangkan aplikasi yang sama untuk platform yang berbeda memakan waktu dan biasanya melibatkan pengoptimalan langkah inferensi dan pemrosesan untuk berjalan dengan benar dan efisien pada perangkat target.

MediaPipe mengatasi tantangan ini dengan mengabstraksi dan menghubungkan model persepsi individu ke dalam pipa yang dapat dipelihara. Semua langkah yang diperlukan untuk mengambil inferensi dari data sensorik dan mendapatkan hasil yang dirasakan ditentukan dalam konfigurasi pipa. Mudah untuk menggunakan kembali komponen MediaPipe dalam pipa yang berbeda di seluruh aplikasi berturut-turut karena komponen ini berbagi antarmuka umum yang berorientasi pada data seri waktu. Setiap pipa kemudian dapat berjalan dengan perilaku yang sama di berbagai platform, memungkinkan praktisi untuk mengembangkan aplikasi di stasiun kerja, dan kemudian menerapkannya di mobile, misalnya. (Lugaresi et al., 2019)

MediaPipe terdiri dari tiga bagian utama: kerangka kerja untuk inferensi dari data sensorik, seperangkat alat untuk evaluasi kinerja, dan koleksi komponen inferensi dan pemrosesan yang dapat digunakan kembali yang disebut kalkulator.

2.5.1 MediaPipe Pose

MediaPipe Pose (MPP), sebuah kerangka kerja lintas platform sumber terbuka yang disediakan oleh Google, digunakan untuk mendapatkan perkiraan koordinat sendi manusia 2D dalam setiap bingkai gambar. MediaPipe Pose membangun pipa dan memproses data kognitif dalam bentuk video menggunakan pembelajaran mesin (*machine learning* - ML). MPP menggunakan BlazePose yang mengekstrak 33 landmark 2D pada tubuh manusia seperti yang ditunjukkan pada Gambar 2.2. BlazePose adalah arsitektur pembelajaran mesin ringan yang mencapai kinerja real-time pada telepon seluler dan PC dengan inferensi CPU. Ketika menggunakan koordinat yang dinormalisasi untuk estimasi pose, rasio invers harus dikalikan dengan nilai piksel sumbu-y. (Kim et al., 2023)



Gambar 2.2: Pose MediaPipe

2.6 Classification Performance

Dalam melakukan proses pengklasifikasian, diperlukan perhitungan efektifitas model yang telah dibuat berdasarkan beberapa pengukuran menggunakan set data pengetesan. Dalam hal ini, *confusion matrix* merupakan salah satu perhitungan yang sering digunakan dalam kasus pengklasifikasian.

Confusion matrix merupakan salah satu pengukuran yang paling mudah dilakukan dalam mencari nilai tingkat kebenaran dan juga akurasi dari model. *Confusion matrix* adalah sebuah tabel berbentuk dua dimensi yang terdiri dari data aktual dan data prediksi yang masing-masing memiliki kelas. Data aktual terletak pada bagian kolom tabel, sedangkan data prediksi terletak pada bagian baris dari tabel. Gambar 2.3 merupakan representasi visual dari perhitungan *confusion matrix*.

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

Gambar 2.3: Confusion Matrix

2.7 Evaluation Metrics

Penggunaan metrik evaluasi dalam konteks Deteksi Objek adalah sangat penting, karena ini berperan sebagai dasar kritis untuk membandingkan efektivitas berbagai algoritma dan skenario yang berbeda. Evaluasi yang teliti memungkinkan para peneliti untuk membuat perbandingan yang akurat antara berbagai teknik deteksi objek, serta menilai seberapa tepat tingkat keakuratan yang dicapai. Ini menjadi sangat vital dalam memilih algoritma yang paling cocok untuk tugas deteksi spesifik. Metrik seperti akurasi, presisi, dan recall adalah kunci dalam

mengevaluasi seberapa efektif suatu model dalam mendekripsi dan mengidentifikasi objek dengan benar. Metrik-metrik ini penting untuk diimplementasikan demi menentukan model yang paling efisien.

Lebih lanjut, analisis akurasi memberikan wawasan kuantitatif yang signifikan mengenai kinerja algoritma deteksi objek, memberikan detail lebih lanjut mengenai kemampuan algoritma dalam menghasilkan deteksi yang akurat. Mengenali kesalahan melalui metrik evaluasi adalah langkah kritis dalam penelitian deteksi, seperti dalam kasus deteksi asap yang kami teliti. Langkah ini memfasilitasi identifikasi dan pemahaman tentang potensi kesalahan dalam algoritma, yang dapat membuka jalan untuk perbaikan dan peningkatan metode deteksi. Selanjutnya, metrik evaluasi juga mendukung pengoptimalan hiperparameter algoritma.

Dalam konteks penelitian ini, berbagai metrik evaluasi telah diterapkan, termasuk presisi, recall, dan Mean Average Precision (mAP). Dengan menggabungkan metode evaluasi ini, penelitian bertujuan untuk menyajikan analisis komprehensif mengenai kinerja dari algoritma deteksi objek yang ditinjau. Adapaun penjelasan konsep sederhana mengenai metrik evaluasi yang akan dijabarkan sebagai berikut :

1. Precision

Precision merupakan rasio dimana TP (true positive) dimana jumlah positif benar dan FP (false positive) jumlah positif palsu sebagai metrik evaluasi dalam konteks machine learning, memberikan ukuran terhadap rasio prediksi positif yang tepat dibandingkan dengan seluruh prediksi positif yang diberikan oleh model. Dengan kata lain, precision memberikan wawasan seberapa akurat model dalam membuat prediksi positif. Lebih rinci, precision mencerminkan seberapa sering model berhasil mengklasifikasikan instance sebagai positif dengan benar dalam keseluruhan dataset. Nilai precision dapat mengindikasikan sejauh mana model mampu memberikan prediksi yang benar dalam konteks positif. Rentang nilai precision berada antara 0 dan 1, di mana nilai 1 menunjukkan bahwa semua prediksi positif model adalah benar, sementara nilai 0 menunjukkan bahwa tidak ada prediksi positif yang benar.

$$\frac{TP}{TP + FP} \quad (2.1)$$

Pada persamaan diatas menyajikan perbandingan antara prediksi positif yang tepat dengan total prediksi positif yang diberikan oleh model, memberikan pandangan yang lebih mendalam terkait kemampuan model dalam menghasilkan hasil yang benar dalam kategori yang diinginkan

2. Recall

Recall merupakan metrik yang digunakan untuk mengukur rasio dari data positif yangbenar yang ditemukan dari seluruh data positif. Recall memberikan informasi tentang seberapa baik model machine learning menemukan semua data positif. Nilai recall berkisar antara 0 dan 1. Recall yang tinggi menunjukkan bahwa kelas yang dikenali dengan benar banyak, atau false negative yang didapatkan sedikit. Rumus dari recall dapat dilihat pada persamaan dibawah

$$\frac{TP}{TP + FN} \quad (2.2)$$

Recall merupakan rasio dimana TP (true positif) adalah jumlah positif benar dan FN (false

negative) jumlah negatif palsu

3. Mean Average Precision (mAP)

Mean Average Precision (mAP) adalah sebuah metrik akurasi yang dihasilkan dari menghitung rata-rata dari Average Precision (AP) atau presisi rata-rata. AP sendiri diperoleh melalui perhitungan precision dan recall. Oleh karena itu, mAP dapat dianggap sebagai metrik evaluasi yang sangat informatif dalam mengevaluasi kinerja suatu sistem.

$$AP = \sum ((Recall_{n+1} - Recall_n) \times Precision_{interp} \times Recall_{n+1}) \quad (2.3)$$

$$mAP = \frac{1}{n} \sum_n^{i=1} AP_i \quad (2.4)$$

2.8 Intersection over Union (IoU)

Intersection over Union, atau IoU, adalah metrik yang digunakan untuk mengevaluasi keakuratan posisi objek yang dideteksi oleh model dalam pemrosesan gambar. Prinsipnya adalah dengan menghitung area persinggungan antara kotak deteksi yang dihasilkan oleh model dengan kotak referensi yang merupakan standar emas atau Ground Truth. Rasio ini didapat dengan membandingkan area irisan kedua kotak tersebut terhadap keseluruhan area yang mereka cakup secara bersamaan. Jika kita membayangkan kedua kotak tersebut sebagai satu kesatuan, maka IoU memberikan kita sebuah skor yang mengukur seberapa baik model kita dalam memprediksi lokasi objek sebenarnya. Semakin besar area persinggungan relatif terhadap total area gabungan, semakin tinggi nilai IoU, yang menandakan keakuratan prediksi yang lebih baik. Secara Sistematis, hal ini dituliskan sebagai :

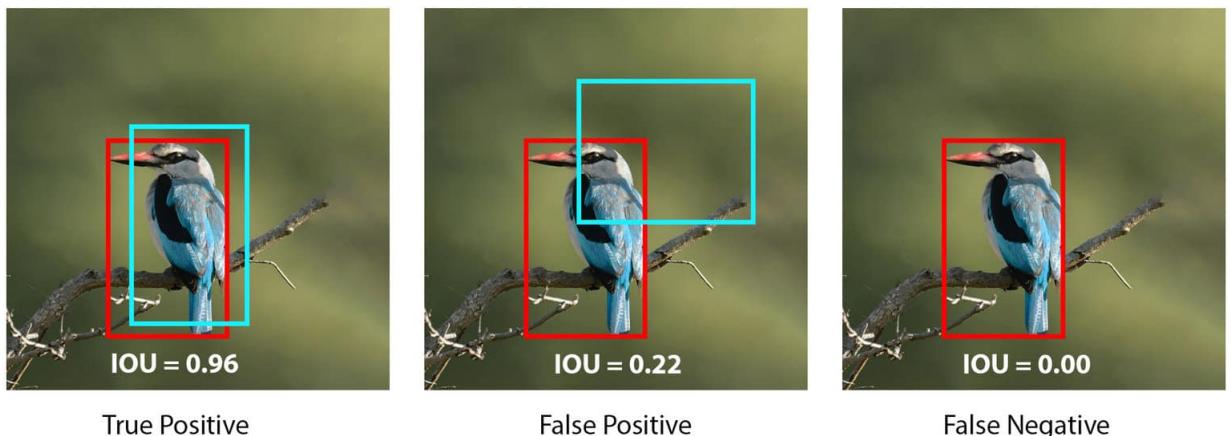
$$Intersection over Union (IoU) = \frac{|A \cap B|}{|A \cup B|}. \quad (2.5)$$

IoU, atau Intersection over Union, merupakan metode penilaian yang mengukur efektivitas model deteksi objek dengan membandingkan area overlap antara prediksi model dan posisi objek aktual. Skala nilai IoU berada antara 0 hingga 1, di mana nilai yang lebih dekat ke 1 menandakan prediksi yang sangat akurat terhadap objek sebenarnya. Nilai IoU yang lebih tinggi menunjukkan bahwa model tersebut lebih tepat dalam mengidentifikasi dan menentukan lokasi objek.

Dalam skenario penelitian, misalnya pengenalan otomatis seekor hewan dalam sebuah gambar, model pembelajaran mendalam akan menciptakan sebuah kotak pembatas sebagai prediksi lokasi hewan tersebut. Kotak pembatas ini lalu dibandingkan dengan kotak kebenaran dasar—area yang telah ditentukan secara manual sebagai lokasi sebenarnya dari hewan dalam gambar. IoU kemudian dihitung untuk menilai seberapa baik kotak prediksi menutupi kotak kebenaran dasar, dengan mempertimbangkan area bersama dan area gabungan dari kedua kotak tersebut.

Dengan demikian, IoU berfungsi sebagai alat ukur yang penting dalam mengevaluasi kemampuan sebuah model deteksi objek untuk secara akurat menentukan posisi objek dalam berbagai kondisi, seperti variasi ukuran, orientasi, dan konteks objek dalam gambar. Sebuah nilai IoU yang tinggi menunjukkan bahwa model tersebut dapat diandalkan dalam mendeteksi

dan mengidentifikasi objek dengan presisi yang tinggi.



Gambar 2.4: Perbandingan Performa Berdasarkan IoU

IoU menawarkan metode kuantitatif untuk mengevaluasi seberapa akurat model dalam mengenali dan menandai objek dalam gambar. Dalam proses pelatihan, nilai IoU minimal yang ditetapkan memungkinkan penetapan batas bagi kotak prediksi untuk dianggap cocok dengan deteksi yang benar, memfasilitasi penyesuaian antara tingkat akurasi deteksi dan insiden false positif. Penentuan batas IoU tidak bersifat tetap dan dapat disesuaikan, dengan 0,5 yang menjadi standar patokan awal. Kotak prediksi dengan IoU minimal 0,5 terhadap deteksi positif dianggap valid. Penyesuaian ambang nilai ini mempengaruhi balance antara presisi dan daya tangkap, dengan peningkatan nilai ambang cenderung mengurangi kesalahan positif namun berpotensi mengabaikan beberapa deteksi valid.

Penggunaan nilai kebenaran dasar dalam IoU merupakan perbandingan standar antara prediksi dan kondisi aktual objek yang diidentifikasi. Penandaan kotak kebenaran dasar, dilakukan secara manual oleh pakar, menentukan batas pasti objek dalam gambar. Skor IoU yang dihasilkan dari perbandingan antara prediksi model dengan batas ini memberikan wawasan terhadap efektivitas model dalam deteksi objek. Dataset kebenaran dasar, yang meliputi kotak pembatas yang ditandai secara manual, menjadi kunci dalam proses evaluasi ini, memberikan dasar objektif untuk mengukur kinerja algoritme deteksi objek.

Dengan demikian, IoU bukan hanya sekedar metrik evaluasi tetapi juga alat penting dalam pengembangan dan penyesuaian model deteksi objek, memastikan bahwa model dapat dianalkan dan akurat dalam berbagai kondisi dan skenario pengujian. (Shah, 2023)

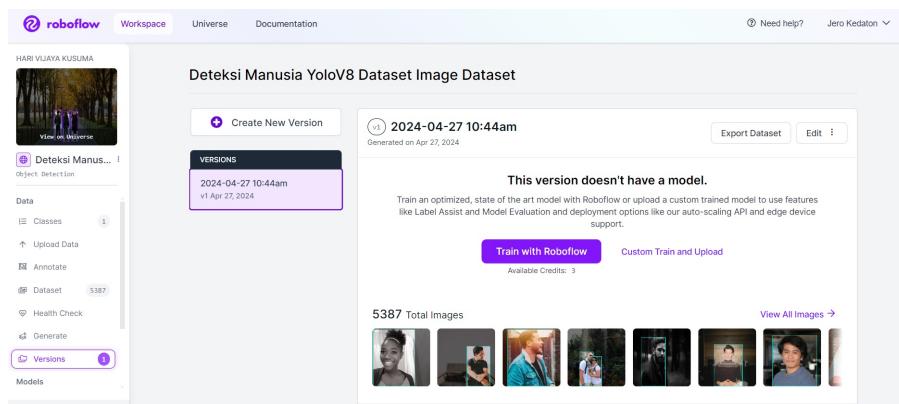
2.9 Single Board Computer (SBC)

Single Board Computer (SBC) adalah komputer lengkap yang dibangun pada satu papan sirkuit tunggal, dengan mikroprosesor, memori, input/output (I/O), dan fitur lain yang diperlukan untuk fungsi komputer penuh. SBC dirancang untuk menjadi kompak dan efisien, sering digunakan dalam aplikasi tertanam, perangkat IoT, dan proyek-proyek pengembangan karena kemudahan penggunaan dan ukurannya yang kecil. SBC menyediakan platform yang ideal untuk pengembangan prototipe dan implementasi sistem kontrol dalam berbagai aplikasi, termasuk kursi roda otonom.

Dalam perkembangan teknologi , penggunaan Single Board Computer (SBC) seperti Intel NUC, NVIDIA Jetson Nano Developer Kit dan Rasberry Pi 4 telah berperan sangat penting terutama dalam Deteksi Real-time. Kecepatan yang ditawarkan SBC dalam memproses data dari sensor dan kamera , serta menjalankan model visi komputer berbasis deep learning membuat alat ini menjadi standar dalam memilih alat untuk implementasi model Deteksi secara real time. Selain itu SBC juga biasanya dilengkapi dengan konektifitas yang sangat baik sehingga dapat terhubung dengan banyak IO. (Elliot et al., 2023)

2.10 Roboflow

RoboFlow adalah platform yang mendukung pengembangan dan penyebaran aplikasi visi komputer dengan menyediakan alat-alat canggih untuk manajemen dan peningkatan dataset. Platform ini dirancang untuk memudahkan pengolahan, analisis, dan augmentasi data visual, sehingga mempercepat siklus pengembangan dan peningkatan model pembelajaran mesin.



Gambar 2.5: Interface Roboflow

RoboFlow menyediakan serangkaian fungsi yang membantu dalam proses pengembangan model visi komputer, termasuk:

- **Annotasi Data :** RoboFlow memungkinkan pengguna untuk menandai gambar dengan alat bantu yang intuitif, mempercepat proses pembuatan label untuk dataset.
- **Augmentasi Data:** Melalui augmentasi data, RoboFlow dapat secara otomatis memodifikasi gambar dalam dataset untuk menciptakan variasi, yang membantu dalam meningkatkan robustness model yang dilatih.
- **Konversi Format Data:** Platform ini mendukung konversi antara berbagai format dataset yang populer, memudahkan pengguna dalam mempersiapkan data untuk berbagai jenis algoritma pembelajaran mesin.
- **Pemisahan Dataset:** RoboFlow menyediakan fungsi untuk membagi dataset menjadi set pelatihan, validasi, dan pengujian, yang merupakan langkah penting dalam validasi model.

RoboFlow menawarkan integrasi yang mulus dengan banyak kerangka kerja pembelajaran mesin populer seperti TensorFlow, PyTorch, dan YOLO. Integrasi ini memungkinkan pengembang:

- Ekspor Data: Pengguna dapat dengan mudah mengekspor dataset mereka dalam format yang siap digunakan oleh kerangka kerja pembelajaran mesin pilihan mereka.
- Pelatihan Model: Platform ini menyediakan alat yang memungkinkan pengguna untuk langsung melatih model mereka menggunakan dataset yang telah disiapkan dan dioptimalkan.
- Evaluasi Model: RoboFlow menyediakan metrik untuk mengukur kinerja model, membantu pengguna memahami efektivitas model mereka dan membuat penyesuaian yang diperlukan.

2.11 Intel NUC

Intel NUC (Next Unit of Computing) adalah solusi komputasi yang kompak dan kuat yang dirancang oleh Intel untuk memenuhi berbagai kebutuhan komputasi, mulai dari hiburan rumah hingga gaming dan tugas profesional. Intel NUC dengan fitur prosesor Intel Core generasi dalam form factor kompak 4x4 inci. Dirancang untuk menawarkan kombinasi ukuran, kinerja, keberlanjutan, dan keandalan yang dibutuhkan oleh bisnis modern. Intel NUC model tertentu juga menyertakan teknologi Intel vPro® Enterprise dengan keamanan yang ditingkatkan. Mini PC ini dapat diupgrade dan diperbaiki, menjadikannya pilihan serbaguna untuk berbagai aplikasi bisnis termasuk komputasi klien, komputasi edge, dan digital signage.



Gambar 2.6: Gambar Intel NUC

2.12 ESP32 Devkit V1

ESP32 Devkit V1 adalah salah satu development board yang dibuat oleh DOIT untuk menjalankan modul ESP-WROOM-32 buatan Espressif. ESP32 Devkit dikenal dengan Development board yang kaya fitur dengan konektivitas Wi-Fi dan Bluetooth terintegrasi untuk beragam aplikasi. Devkit ini memiliki banyak pin yang memungkinkannya untuk diprogram dengan banyak tugas.



Gambar 2.7: Gambar ESP32 Devkit V1

2.13 Motor Driver H-Bridge

Driver motor H-Bridge adalah rangkaian elektronik yang digunakan untuk mengontrol arah dan kecepatan motor DC. Cara kerjanya berdasarkan empat switch yang membentuk jembatan H (H-Bridge), yang mana dengan mengatur pembukaan dan penutupan switch-switch ini, kita dapat mengatur arah arus yang mengalir ke motor. Dengan demikian, kita bisa mengubah arah putaran motor DC. Driver Motor H-Bridge tersusun oleh sekumpulan transistor yang berfungsi sebagai pengendali motor, terutama yang memerlukan arus serta tegangan yang cukup besar. selain itu, Rangkaian H-Bridge juga dapat memberikan fungsi pengereman pada motor dengan menghubungkan kedua terminal motor sehingga motor dapat berhenti lebih cepat. (Muhammad, 2018)



Gambar 2.8: Gambar Motor Driver H-Bridge

2.14 Kursi Roda Elektrik KY-123

Kursi roda bertenaga listrik merupakan alat bantu mobilitas yang terdiri dari struktur dasar kursi roda, sistem pengendalian gerak, mesin elektrik, dan modul baterai. Keunggulan alat ini terletak pada kemampuannya untuk dikendalikan dengan mudah dan nyaman, meminimalkan usaha fisik yang diperlukan pengguna dibandingkan dengan kursi roda manual. Ini sangat bermanfaat bagi individu dengan kondisi hemiplegia, memungkinkan pengoperasian dengan satu tangan. Selain itu, kursi roda elektrik ini juga memberikan solusi mobilitas yang lebih baik bagi lansia yang mengalami keterbatasan dalam bergerak.



Gambar 2.9: Gambar Kursi Roda Elektrik KY-123

[Halaman ini sengaja dikosongkan]

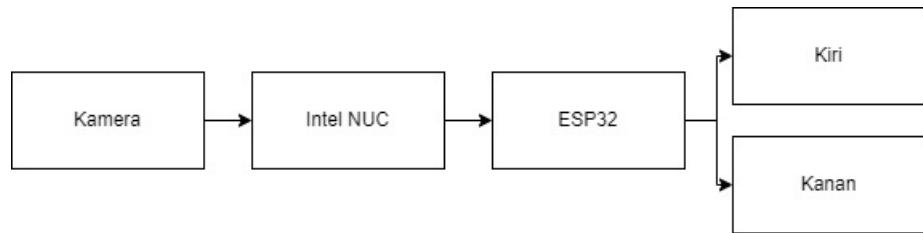
BAB III

DESAIN DAN IMPLEMENTASI

Penelitian ini dilakukan sesuai dengan desain sistem berikut beserta implementasinya. Desain sistem adalah konsep dari pembuatan dan perancangan infrastruktur dan kemudian diwujudkan dalam bentuk alur yang harus dikerjakan

3.1 Deskripsi Sistem

Penelitian dan pembuatan sistem ini diterapkan sesuai dengan desain dan implementasi pada bab ini. Desain sistem ini mencakup konsep pembuatan, perancangan, alur, dan implementasi infrastruktur yang dibuat dalam Blok Diagram. Desain dan penerapan diilustrasikan melalui penggunaan Gambar dan akan dijelaskan mulai dari pengumpulan data berupa citra, analisa dari model yang telah dibuat untuk mendeteksi objek Manusia, serta sistem yang menggunakan model tersebut seperti Gambar 3.1 berikut dan dirincikan pada tiap subbab.



Gambar 3.1: Blok Diagram Hardware

3.1.1 Kamera

Pada tugas akhir ini pendeksi menggunakan data citra yang diolah untuk mendapatkan output berupa deteksi manusia yang akan dianalisa jaraknya. Kamera menjadi input utama untuk mendapatkan citra yang akan dipasangkan dalam kursi roda otomotif. Posisi kamera yang digunakan harus berada pada posisi yang dapat menangkap citra manusia dengan jelas. Adapun kamera yang digunakan adalah kamera Logitech C920.

Kamera Logitech C920 dilengkapi dengan lensa kaca Carl Zeiss dan sensor Gambar Full HD 1080p. Kamera ini juga mendukung video call dalam resolusi 720p yang berkualitas tinggi. Fitur autofocus otomatisnya memastikan bahwa Gambar tetap fokus bahkan ketika pengguna bergerak. Logitech C920 juga dilengkapi dengan dua mikrofon stereo yang dapat merespons suara dengan jelas dan natural, menghasilkan pengalaman audio yang memuaskan tanpa perlu penggunaan mikrofon eksternal. Dengan kombinasi fitur dan spesifikasi ini, Logitech C920 sudah lebih dari cukup untuk digunakan sebagai kamera untuk mendeksi obstacle yang hendak dideteksi.

3.1.2 Intel NUC

Intel NUC digunakan sebagai komputer pusat dalam kursi roda otomotif untuk memproses citra dari kamera dan menjalankan algoritma deteksi objek. Data yang dihasilkan dari deteksi

digunakan untuk mengambil keputusan navigasi. Misalnya, menghindari hambatan dan menentukan arah gerak. Perintah tersebut kemudian dikirim ke ESP32 yang mengontrol mekanisme penggerak kursi roda.

Penggunaan Intel NUC tidak langsung dapat menjalankan sistem yang dibuat, melainkan perlu diinstal library agar dapat menjalankan sistem yang telah dibuat.

```
1 pip install opencv-python
2 pip install ultralytics
3 pip install mediapipe
```

opencv-python digunakan untuk menangkap dan memproses citra dari kamera. Ultralytics digunakan untuk menjalankan deteksi objek menggunakan model YOLOv8. Mediapipe digunakan untuk menampilkan landmark pada manusia

3.1.3 ESP32

Untuk dapat menggerakkan kursi roda maka perlu mengirimkan perintah ke kontroler kursi roda. ESP32 menjadi akan menerima input perintah dasar untuk menggerakkan kursi roda, seperti maju, kanan, kiri. Perintah ini kemudian akan digabungkan dengan kecepatan maksimal menjadi satu command atau paket data seperti "Arah". Berikut merupakan tabel kode instruksi kursi roda berdasarkan hasil deteksi (Ekatama, 2024)

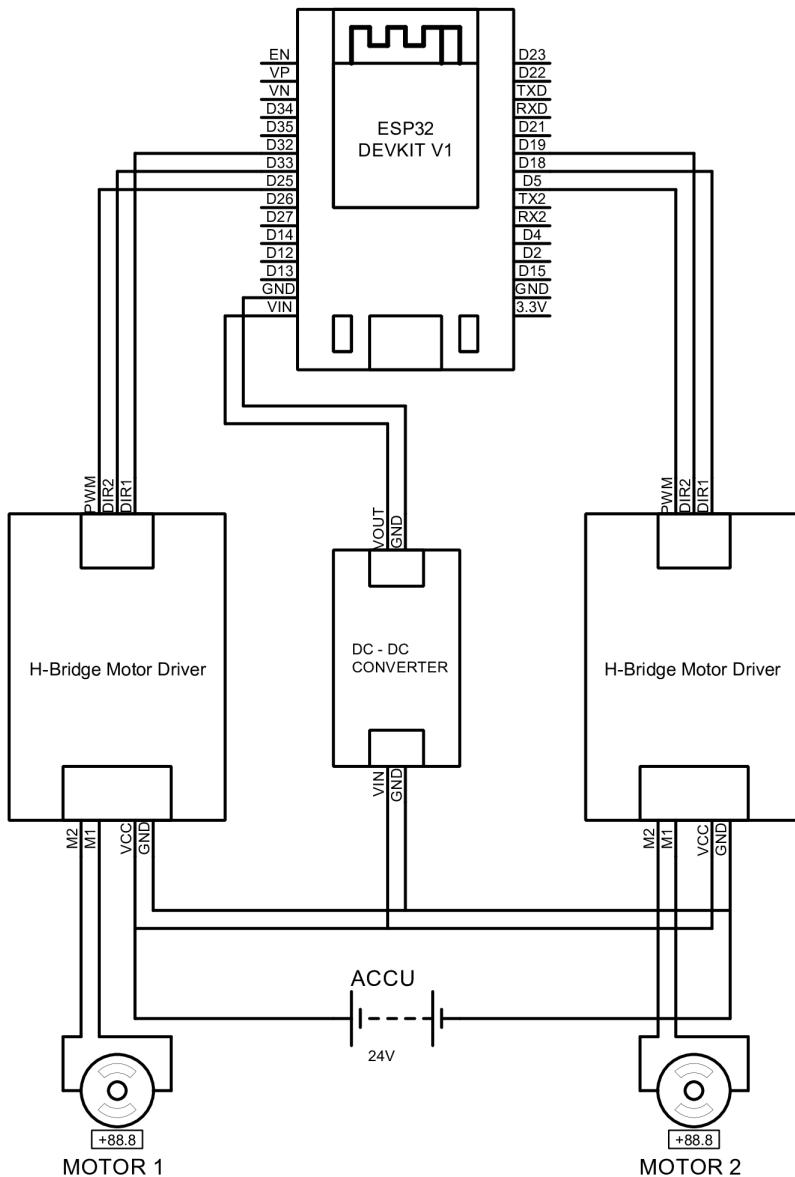
Tabel 3.1: Kode instruksi dari hasil klasifikasi

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

Setelah variabel tersebut dimasukkan maka akan dikirim secara nirkabel. Dalam tugas akhir ini menggunakan wifi dengan ssid Haris-Acess-Point setelah terkoneksi.

Paket data yang telah dikirimkan melalui NUC akan diterima oleh ESP32 menggunakan WiFi. Saat diterima oleh ESP32, data tersebut akan menjalani serangkaian proses yang melibatkan pemecahan paket data dan penyesuaian sesuai dengan variabel yang telah ditentukan sebelumnya. Pemecahan paket data ini memungkinkan ESP32 untuk mendekomposisi informasi yang terkandung dalam setiap paket dan memastikan bahwa setiap variabel terpisah dengan akurat. Dengan demikian proses ini akan mengorganisir dan menyusun kembali informasi serta memastikan bahwa setiap variabel telah benar sesuai dengan nama variabel dan tipe data yang disediakan.

3.1.4 Skematik Alat



Gambar 3.2: Skematik kontrol motor kursi roda

Gambar menampilkan skema detail dari perangkat yang dibahas. Sistem ini menggunakan kamera yang terhubung ke NUC, yang berfungsi sebagai perangkat utama untuk pengambilan gambar objek. Ketika kamera berhasil menangkap gambar objek, data citra tersebut kemudian diproses oleh Laptop atau Jetson Nano. Di dalam sistem, terdapat model klasifikasi yang telah diprogram untuk menganalisis data citra ini secara detail. Hasil analisis ini sangat penting, sebab menjadi dasar dalam pembuatan kode instruksi yang akan dieksekusi oleh sistem.(Ekatama, 2024)

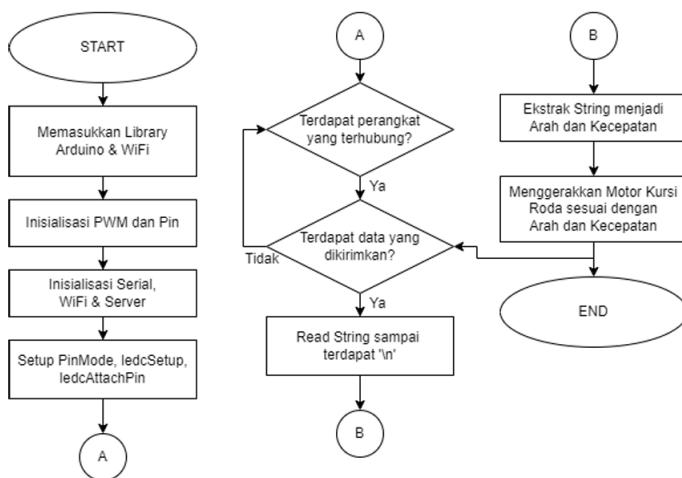
Kode instruksi yang dihasilkan kemudian diintegrasikan dengan parameter kecepatan maksimal yang telah ditetapkan oleh pengguna. Integrasi antara kode instruksi tersebut kemudian dikemas dalam bentuk paket data yang digunakan untuk mengatur dan mengendalikan perg

erakan kursi roda. Paket data ini selanjutnya ditransmisikan secara nirkabel, menggunakan teknologi WiFi, menuju modul ESP32 Devkit V1. (Ekatama, 2024)

ESP32 memegang peranan kunci dalam mekanisme pengendalian motor kursi roda, berfungsi sebagai pusat pengendalian yang menerima paket data dari pengguna melalui koneksi nirkabel. Setelah menerima paket, ESP32 akan menguraikan paket data tersebut untuk menyesuaikan dengan variabel-variabel yang telah ditentukan. Proses dekripsi ini menghasilkan dua variabel utama yang akan diproses lebih lanjut oleh ESP32 untuk pengoperasian kursi roda. (Ekatama, 2024)

Variabel utama pertama adalah variabel arah, yang memainkan peran kritis dalam menentukan arah gerakan motor-motor pada kursi roda, memastikan bahwa gerakan motor selaras dengan arah yang dikehendaki berdasarkan data yang telah dianalisis dan diterima, sehingga pergerakan kursi roda dapat dikontrol dengan aman dan efisien.(Ekatama, 2024)

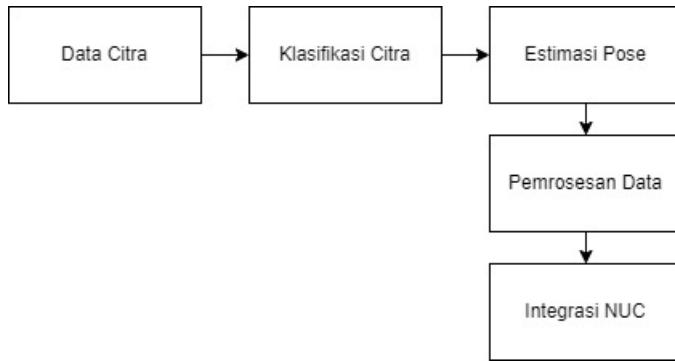
Untuk mengontrol pergerakan kursi roda, terdapat dua buah metode. Metode pertama adalah differential drive yang mana ketika kursi roda akan berbelok ke kanan, maka roda kanan akan bergerak mundur dan berbelok ke kiri sedangkan roda kirinya akan bergerak maju dan berbelok ke kanan. Ketika akan berbelok ke kiri, maka akan terjadi hal sebaliknya. Ketika akan bergerak maju atau mundur makan kedua roda akan bergerak secara bersamaan kearah yang diinginkan. Metode kedua adalah pergerakan biasa yang mana ketika kursi roda akan berbelok ke kanan, maka roda kanan akan diam sedangkan roda kirinya akan bergerak maju dan berbelok ke kanan. Ketika akan berbelok ke kiri, maka akan terjadi hal sebaliknya. Ketika akan bergerak maju atau mundur makan kedua roda akan bergerak secara bersamaan kearah yang diinginkan. Untuk penelitian kali ini akan digunakan metode kedua untuk menggerakan roda pada kursi roda(Ekatama, 2024)



Gambar 3.3: Flowchart kontrol kursi roda

3.2 Software

Perancangan software dilakukan sesuai dengan alur yang akan dideskripsikan pada subbab ini. Perangangan ini akan dipresentasikan dengan blok diagram alur yang telah merepresentasikan alur perancangan software ini. Gambar blok diagram alur akan ditampilkan sebagai berikut :



Gambar 3.4: Diagram Blok Software

3.2.1 Pengumpulan Dataset citra

Dalam pengembangan Tugas Akhir ini akan digunakan dataset citra berupa gambar, Dimana objek yang akan dideteksi pada gambar tersebut ialah Manusia yang fungsinya dalam tugas ini sebagai obstacle yang akan dihindari. citra manusia tersebut diambil melalui setiap frame citra pada video yang didapatkan menggunakan kamera webcam yang terhubung dengan komputer. Kemudian setiap citra yang didapatkan nantinya akan diproses untuk menentukan apakah terdapat manusia atau tidak pada citra tersebut. Dalam pendekalian, nantinya proses ini terjadi secara real-time guna mendapatkan keseluruhan citra yang dibutuhkan untuk menge-nali manusia atau tidak pada citra secara terus-menerus



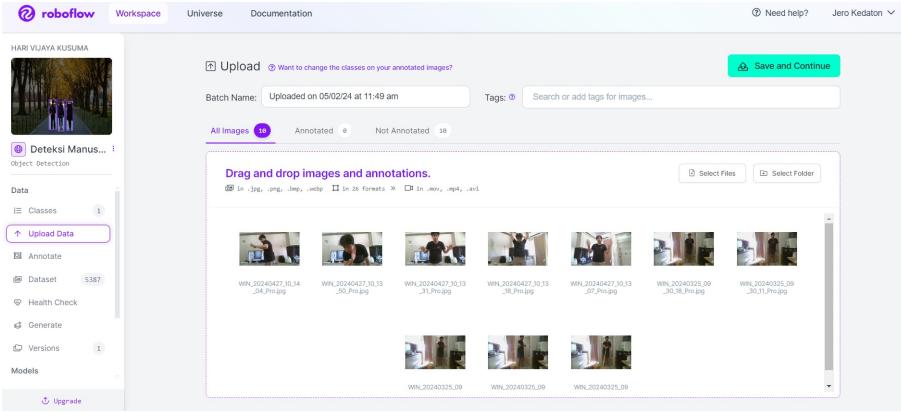
Gambar 3.5: Contoh Hasil data dari Citra.

Dengan menggunakan perangkat kamera yang terhubung dengan komputer, perangkat akan menghubungkan interaksi antara pengguna dengan komputer untuk mendapatkan frame citra pada video agar nantinya dapat diproses. Gambar 3.2 merupakan contoh hasil dari data citra yang didapatkan melalui perangkat kamera pada komputer. Dalam hal ini, karena yang ingin dikenali adalah manusia, maka citra gambar citra yang didapatkan harus memiliki objek manusia didalamnya.

3.2.2 Labeling Menggunakan Roboflow

Dataset citra yang sudah didapatkan selanjutnya akan melalui proses labeling dan augmentasi. Dimana Roboflow memiliki tools yang mumpuni dalam melakukan labeling. Dimana

terdapat beberapa proses yang akan dilakukan yaitu, import dataset, pelabelan dataset dan augmentasi dataset



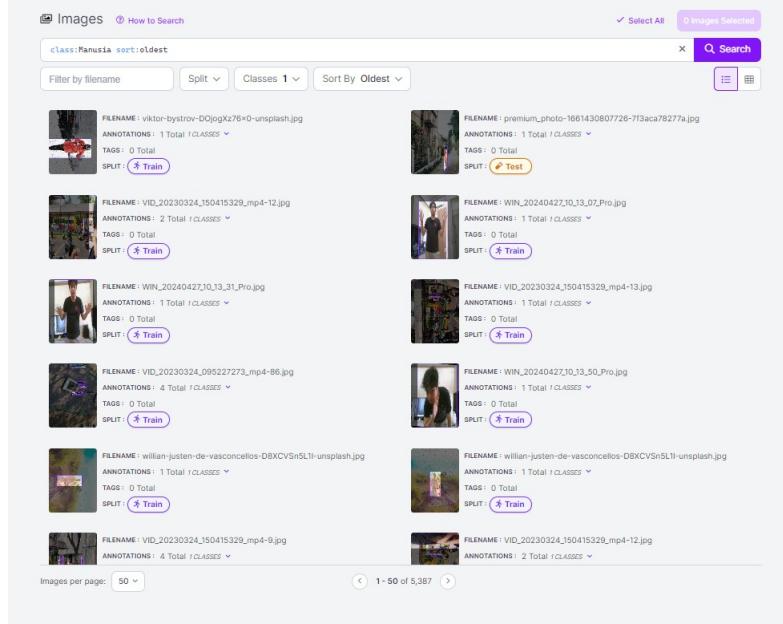
Gambar 3.6: Contoh upload dataset ke roboflow.

Dataset yang diupload haruslah mencakup beberapa skenario dan mencakup berbagai ukuran. Setelah itu melakukan praproses dataset dengan anotasi, yaitu mengubah ukuran Gambar, menormalkan nilai piksel, dan membaginya menjadi set training, validasi, dan test. Lalu menambah data dengan teknik seperti rotasi, penskalaan, atau membalik untuk meningkatkan keragaman sampel pelatihan.



Gambar 3.7: Contoh anotasi dataset ke roboflow.

Dalam proses anotasi penamaan class yang digunakan haruslah sesuai dengan objek yang akan dideteksi. Dalam konteks tugas akhir kali ini obstacle yang dimaksud adalah manusia, maka penamaan class adalah manusia. Selain itu dalam proses anotasi harus diperhatikan posisi objek yang akan dianotasi harus jelas agar tidak ada kesalahan model dalam mendekripsi.



Gambar 3.8: Contoh Dataset.

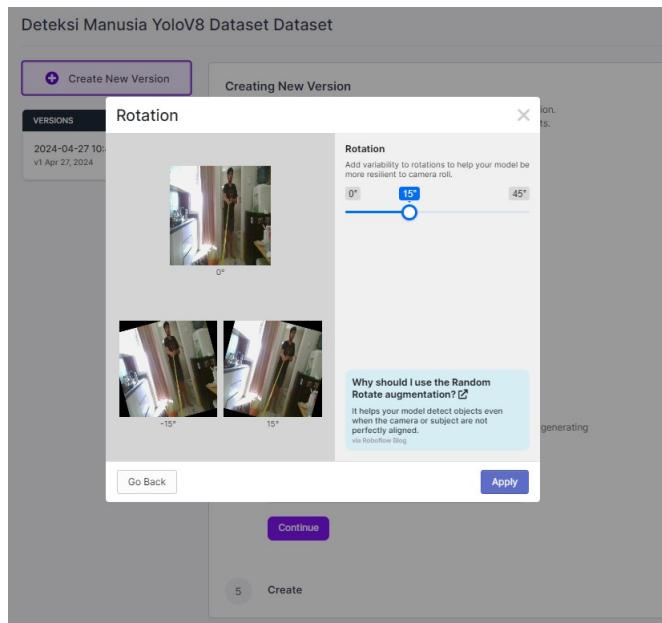
Setelah seluruh Gambar yang dipilih telah dianotasi, terdapat opsi membuat datasets dan Pemilihan Gambar yang dikelompokkan dari grup batch dataset tersebut diunggah. Pengguna dapat melakukan kombinasi dan Pemilihan secara mendetil tentang Gambar spesifik apa yang dibutuhkan dalam proses pembuatan model sehingga bisa merepresentasikan hasil deteksi objek yang hendak dilakukan. Jika Gambar yang telah dianotasi telah memenuhi kriteria yang ditetapkan, maka dibuatlah versi baru dataset dengan menekan tombol create new version berwarna biru. Kemudian pengguna dapat memilih Gambar dan mengatur konfigurasi split pada dataset, praproses dataset, serta augmentasi dataset. Berikut tampilan dari Datasets yang siap untuk dibuat yang diGambarkan pada Gambar Diatas.

Selain itu juga terdapat pula fitur preprocesssing datasets yang berasal dari roboflow dimana berguna untuk menstandarkan format Gambar (misalnya, semua Gambar berukuran sama). Langkah ini penting untuk memastikan kumpulan data konsisten sebelum melatih model. Berikut fitur tersebut.

- Auto - Orient
- Rezise
- Grayscale
- Auto Adjust Contrast
- Isolate Objects
- Static Crop
- Tile
- Modify Classes

- Filter Null

Adapula fitur Augmentasi data yaitu langkah di mana augmentasi diterapkan pada Gambar yang ada di kumpulan data . Proses ini dapat membantu meningkatkan kemampuan model untuk menggeneralisasi sehingga bekerja lebih efektif pada Gambar yang tidak terlihat. Pada awal proses training tidak digunakan proses augmentasi guna mengevaluasi kualitas datasets yang murni yaitu belum mengalami Augmentasi. Jika augmentasi ditambahkan dan dataset tidak berkinerja sebaik yang diharapkan, maka tidak akan ada dasar untuk membandingkan kinerja model.

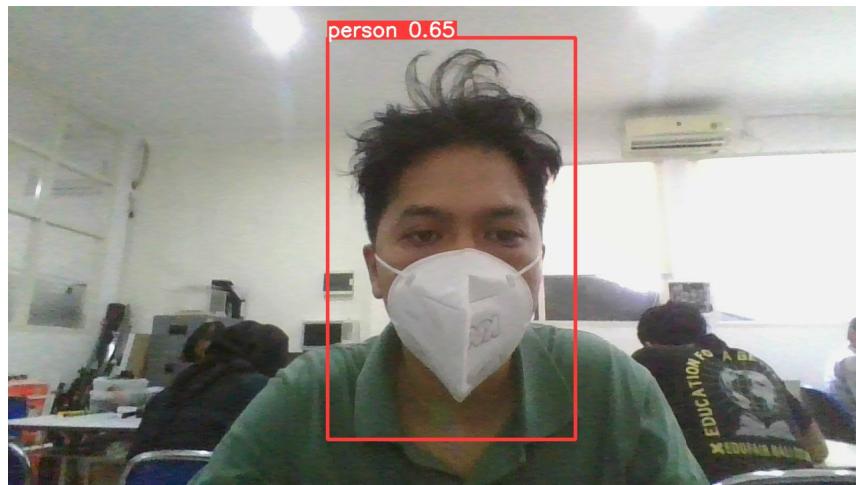


Gambar 3.9: Contoh Augmentasi Dataset.

Jika performa model tidak baik tanpa augmentasi, hal yang perlu dilakukan yaitu menyelidiki keseimbangan kelas, representasi data, dan ukuran kumpulan data. Jika terdapat kumpulan data yang modelnya telah berhasil dilatih tanpa augmentasi, maka proses penambahan augmentasi untuk lebih membantu meningkatkan performa model dapat dilakukan.

3.2.3 Object Detection YoloV8

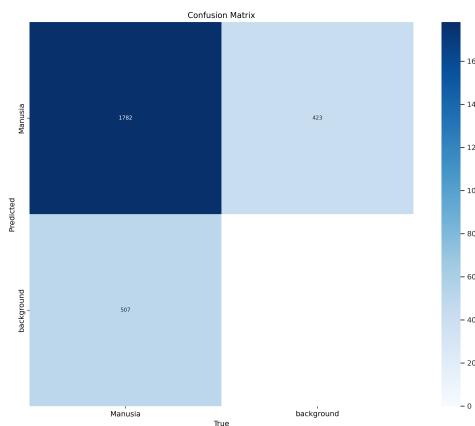
Deteksi Objek dilakukan dengan mendeteksi keberadaan Manusia pada citra yang didapatkan, kemudian keberadaan manusia yang terdeteksi pada citra, akan digambar bounding box pada area yang terdeteksi manusia. Dimana Bounding box yang terdeteksi tersebut akan memiliki nilai nilai x dan y dalam posisi pixel pada jendela kamera yang selanjutnya dapat diambil berupa tinggi dan lebar pada pixel yang terdeteksi pada jendela web kamera. Dimana nilai nilai tersebut akan menjadi acuan dalam menentukan jarak objek relatif terhadap kamera.



Gambar 3.10: Contoh Hasil Deteksi Menggunakan Model YoloV8 yang telah di train.

Bounding box yang didapatkan merupakan hasil dari training yang akan dilakukan sehingga model YoloV8 dapat mengidentifikasi kelas yang diinginkan yaitu kelas Manusia. Berikut merupakan contoh gambar hasil dari identifikasi objek yang dihasilkan dari model yang telah dilakukan training. Dapat dilihat hasil deteksi menghasilkan nilai, yang merupakan nilai confidence dari model dimana nilai ini menunjukkan seberapa yakin citra yang diinput merupakan bagian class yang terdeteksi.

Dalam proses training akan didapatkan beberapa nilai yang akan menjadi acuan dalam seberapa baik model dalam mendeteksi objek yang terdapat dari input citra, dimana nilai-nilai tersebut akan divisualisasikan melalui confusion matrix maupun grafik yang mengindikasikan performa model dalam mendeteksi. berikut contoh gambar confusion matrix

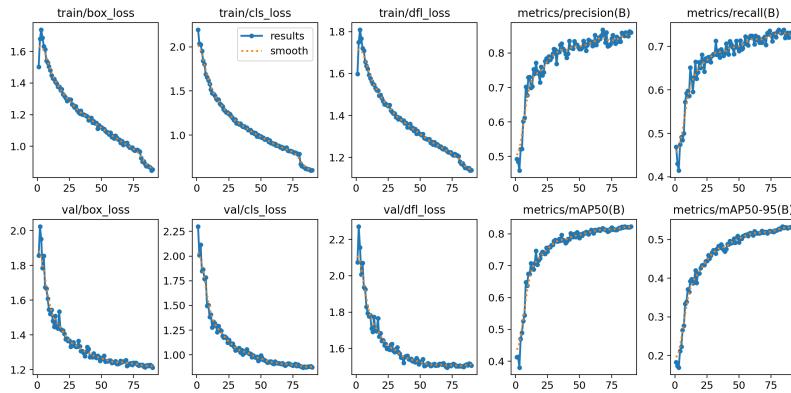


Gambar 3.11: Contoh Confusion Matrix model.

Dimana dapat dilihat pada confusion matrix dimana nilai hasil pojok kiri atas merupakan nilai true positive, sedangkan nilai pojok kanan atas merupakan false negatif, nilai pojok kiri bawah adalah false positive, dan terakhir nilai pojok kanan bawah merupakan true negative.

Selain confusion matrix juga terdapat classification loss. Classification Loss mengukur nilai loss model dalam memprediksi klasifikasi objek yang telah ditentukan melalui proses pelabelan

atau anotasi data. Sebuah nilai loss klasifikasi yang rendah menunjukkan bahwa model cenderung menjadi lebih optimal. Localization Loss, pada sisi lain, mencerminkan kinerja sistem dalam menentukan posisi suatu objek. Semakin rendah nilai loss lokalisisasi, semakin baik model yang dihasilkan. Contoh grafik tersebut dapat dilihat pada gambar berikut.



Gambar 3.12: Contoh grafik loss dan mAP model.

Penurunan nilai Localization loss mencerminkan bahwa model yang telah dilatih secara pra-terlatih mampu mengidentifikasi objek dalam Gambar dan menghasilkan koordinat bounding box yang mendekati koordinat bounding box yang diannotasikan. Regularization Loss merujuk pada serangkaian teknik yang bertujuan untuk mengurangi kompleksitas model jaringan saraf selama proses pelatihan, sehingga mencegah terjadinya overfitting. Penerapan berbagai teknik regularisasi dapat membantu mengatasi masalah overfitting dalam jaringan saraf, yang pada gilirannya meningkatkan akurasi model Deep Learning saat dihadapkan pada data baru dari domain permasalahan yang sama.

Nilai loss regularisasi mencerminkan tingkat kompleksitas model, dengan nilai yang lebih kecil mengindikasikan bahwa model memiliki kemungkinan lebih rendah untuk mengalami overfitting. Loss total mencakup seluruh proses pelatihan model dan dihitung berdasarkan tiga parameter utama, yaitu loss klasifikasi, loss lokalisisasi, dan loss regularisasi, memberikan pandangan menyeluruh terhadap proses pelatihan model. Oleh karena itu, evaluasi kinerja deteksi objek tidak hanya memperhatikan kemampuan model dalam mengklasifikasikan objek, tetapi juga dalam menentukan posisinya dengan akurasi tinggi dan menghindari overfitting yang berlebihan

Dalam penelitian ini, diperkenalkan tiga komponen utama dalam evaluasi kinerja model deteksi objek, yaitu classification loss, localization loss, dan regularization loss. Classification loss mengukur kesalahan model dalam memprediksi klasifikasi objek yang telah diidentifikasi melalui proses pelabelan atau anotasi data. Semakin rendah nilai classification loss, semakin optimal model dianggap. Di sisi lain, localization loss mengindikasikan kemampuan sistem dalam menentukan posisi objek dalam Gambar. Penurunan nilai localization loss mencerminkan peningkatan kinerja model, terutama dalam menghasilkan koordinat bounding box yang mendekati anotasi yang sebenarnya. Regularisasi, sebagai serangkaian teknik, digunakan untuk mengurangi kompleksitas model jaringan saraf selama pelatihan, dengan tujuan menghindari overfitting. Penerapan berbagai teknik regularisasi membantu meningkatkan generalisasi model terhadap data baru. Nilai regularization loss memberikan Gambaran tentang seberapa kompleks model tersebut, dan nilai yang lebih rendah menunjukkan kemungkinan lebih kecil terjadinya

overfitting.

Dapat dilihat pada bagian kanan terdapat grafik precision, dimana grafik ini menunjukkan presisi model, yaitu proporsi prediksi positif yang benar-benar positif. Grafik yang menunjukkan peningkatan secara bertahap menunjukkan bahwa model semakin jarang membuat prediksi positif yang salah. Peningkatan ini juga dialami oleh grafik recall yang mana grafik Recall ini mengukur proporsi positif aktual yang berhasil dideteksi oleh model. Grafik dengan peningkatan menunjukkan bahwa model menjadi lebih baik dalam mendeteksi semua kasus positif yang ada. Selanjutnya terdapat grafik mAP (*mean Average Precision*) pada threshold IoU (Intersection over Union) 50% adalah metrik yang menilai secara keseluruhan kinerja model dalam mendeteksi objek dengan keakuratan yang diberikan. Nilai yang lebih tinggi menunjukkan performa yang lebih baik. Dan terakhir mAP50-95 Grafik ini mirip dengan mAP50 tetapi dihitung sebagai rata-rata dari IoU yang berkisar dari 50% hingga 95%. Ini adalah metrik yang lebih ketat dan menunjukkan kinerja model secara lebih rinci dalam berbagai tingkat ketat deteksi.

3.2.4 Estimasi Pose MediaPipe

Deteksi pose dilakukan dengan menggunakan Python dengan library OpenCV dan framework MediaPipe menggunakan fungsi pose detection saat objek manusia terdeteksi dalam frame citra. Framework MediaPipe digunakan untuk mendapatkan landmark pada tubuh peraga, lalu landmark yang relevan akan digambarkan garis berbentuk kerangka yang sesuai dengan pose tubuh peraga. Dalam penelitian ini, landmark yang relevan yaitu keypoint siku atas hingga lengan bawah dan bahu kanan dan kiri. Titik keypoint yang digunakan pada estimasi pose dapat dilihat pada Tabel dibawah ini.

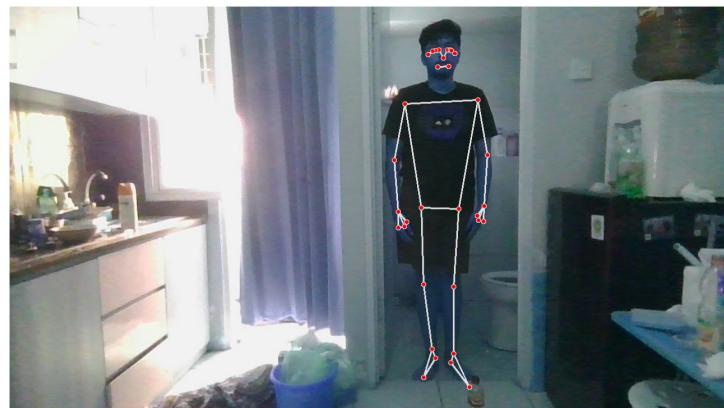
Tabel 3.2: Tabel Keypoint yang digunakan

Nomor Keypoint	Nama Keypoint
11	RIGHT_SHOULDER
12	LEFT_SHOULDER
14	RIGHT_ELBOW
16	RIGHT_WRIST

Dari titik landmark yang didapat akan diambil jarak terhadap pixelnya dimana nilai pixel yang didapatkan akan menjadi acuan dalam perhitungan yang akan digunakan untuk menghindari obstacle. Berikut contoh landmark pada data citra.



Gambar 3.13: Contoh hasil deteksi pose.



Gambar 3.14: Contoh hasil deteksi pose seluruh tubuh.

Penggunaan landmark lengan dan bahu sebagai acuan dalam perhitungan berdasar pada visibilitas, dimana visibilitas kedua landmark ini cenderung lebih baik ketimbang landmark kaki dan kepala terutama dalam kasus penghindaran manusia. Dimana dalam jarak dekat kedua landmark ini akan lebih terlihat ketimbang landmark lainnya. Selain itu penggunaan landmark ini juga dapat menghasilkan nilai yang lebih konsisten ketimbang landmark lainnya.

3.2.5 Perhitungan Rumus

Untuk dapat menentukan jarak, pertama tama model Yolo akan digunakan untuk mendeteksi class Manusia yang akan menjadi obstacle dalam tugas akhir ini. Dari hasil deteksi yang didapat akan digambar Bounding Box penanda kelas telah terdeteksi. Dimana dalam bounding box tersebut juga akan digambar Estimasi Pose dari MediaPipe. Kedua hasil deteksi ini yaitu Bounding Box dan Pose yang dihasilkan akan digunakan untuk mendapat estimasi jarak berdasarkan rumus yang akan dijabarkan, dimana hasil dari jarak yang didapatkan akan dipetakan perpindahannya dalam grid. Dimana grid ini akan menjadi acuan dari keputusan belok yang akan diambil. berikut merupakan contoh gambar pemetaan hasil deteksi dan pengambilan keputusan belok.



Gambar 3.15: Contoh Penerapan hasil perhitungan jarak dan pemetaan grid.

Dapat dilihat bahwa Terdapat banyak variabel pada jendela web kamera. Semua variabel tersebut memiliki peran masing masing, yang akan membantu mendapatkan nilai yang akan dipetakan dalam grid pada bagian kiri bawah.

Dalam Tugas Akhir ini terdapat beberapa variabel yang akan diukur. Variabel-variabel tersebut yaitu jarak manusia, lebar manusia, dan posisi manusia. Dimana Jarak manusia ini akan dibedakan menjadi 2 perhitungan, dimana 2 perhitungan ini berdasarkan 2 hasil deteksi yang didapatkan yaitu bounding box dan pose.

1. Perhitungan Jarak Obstacle berdasarkan Bounding Box

Salah satu cara untuk mendapatkan jarak melalui Bounding Box adalah dengan menggunakan *Focal Length Pixel*. Fokus panjang dalam piksel, atau *focal length pixel*, adalah sebuah konversi dari fokus panjang lensa kamera yang biasanya diukur dalam milimeter ke dalam satuan piksel. Ini adalah konsep kunci dalam fotogrametri dan visi komputer yang digunakan untuk menghubungkan informasi visual yang diperoleh dari kamera ke ukuran fisik dalam dunia nyata.

Dalam konteks tugas akhir ini Fokus panjang dalam piksel digunakan untuk mengkonversi ukuran halangan dari unit piksel menjadi unit meter. Ini penting karena kursi roda otonom perlu memahami jarak nyata ke halangan untuk mengambil keputusan navigasi yang tepat. Dalam penggunaannya dapat dilihat dalam rumus berikut.

$$focal_length_pixel = \left(\frac{jarak \times tinggi_bounding_box}{tinggi_objek_nyata} \right) \quad (3.1)$$

Fokus panjang dalam piksel adalah parameter penting dalam perhitungan jarak. Ini digunakan untuk mengonversi pengukuran dari ruang gambar (piksel) menjadi dimensi dunia nyata (seperti meter). Nilai ini mewakili fokus panjang lensa kamera dalam satuan piksel, yang diperoleh dari prosedur kalibrasi.

Saat menghitung jarak ke objek dengan menggunakan tinggi dari bounding box yang terdeteksi oleh YOLO, dikombinasikan dengan tinggi nyata objek yang diketahui dan fokus panjang dalam piksel. penerapannya dapat dilihat sebagai berikut :

$$jarak = \left(\frac{focal_length_pixel \times tinggi_objek_nyata}{tinggi_bounding_box} \right) \quad (3.2)$$

Di sini, tinggi_objek_nyata mungkin merupakan tinggi rata-rata orang atau objek lain yang diidentifikasi sebagai obstacle. Dengan menggunakan tinggi bounding box yang dideteksi oleh YOLOv8, kursi roda dapat menghitung jarak ke objek tersebut, yang sangat penting untuk menghindari benturan dan pengambilan keputusan.

2. Perhitungan Jarak Obstacle berdasarkan Pose

Salah satu pendekatan yang baik dalam menentukan jarak dengan pose adalah menggunakan metode Jarak Euclidean. Jarak Euclidean dalam piksel adalah metode untuk mengukur jarak lurus antara dua titik dalam ruang gambar, yang biasanya diukur dalam piksel. Dalam konteks tugas akhir ini yang melibatkan kursi roda otomotif dengan integrasi MediaPipe, pengukuran ini sangat penting untuk berbagai fungsi, terutama dalam analisis pose dan penilaian proporsi objek dalam citra yang dihasilkan oleh kamera.

$$jarak_euclidean = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \times scale_factor \quad (3.3)$$

Rumus ini menghasilkan jarak antara dua titik dalam satuan yang sama dengan satuan koordinat $x1,y1$ dan $x2,y2$. Biasanya, jika koordinat-kordinat ini diukur dalam piksel, maka jarak yang dihasilkan juga akan dalam piksel.

Penambahan faktor skala ke dalam perhitungan jarak Euclidean berguna ketika perlu mengkonversi jarak dari satu unit ke unit lain, atau ketika koordinat titik disesuaikan ke skala tertentu yang tidak merefleksikan dimensi sebenarnya dalam piksel. Normalisasi Koordinat Dalam banyak aplikasi visi komputer, koordinat titik mungkin dinormalisasi ke skala [0, 1]. Di sini $x1,y1$ dan $x2,y2$ adalah proporsi dari lebar dan tinggi gambar. Untuk menghitung jarak nyata dalam piksel, perlu mendapatkan hasil kali dari rumus Euclidean dengan dimensi gambar.

Dari perhitungan diatas masih didapatkan jarak yang bernilai pixel. Dalam konteks perhitungan jarak perlu menggunakan nilai standar dalam Meter agar perhitungan tersebut sesuai dengan kaidah yang berlaku pada umumnya. Agar nilai pixel tersebut dapat berubah menjadi meter maka perlu dilakukan kalibrasi menggunakan nilai K. nilai K merupakan nilai kalibrasi berdasarkan pengukuran eksperimental. rumusnya dapat dilihat sebagai berikut.

$$jarak_meter = \left(\frac{k}{jarak_pixel} \right) \quad (3.4)$$

Nilai k menentukan seberapa besar pengaruh jarak dalam piksel terhadap jarak dalam meter. Nilai yang lebih besar atau lebih kecil akan secara langsung mempengaruhi hasil perhitungan jarak. Misalnya, nilai k yang lebih besar akan menghasilkan jarak yang lebih kecil untuk jumlah piksel yang sama. Nilai ini digunakan dalam konteks yang sangat spesifik di mana parameter tersebut menggambarkan hubungan langsung antara ukuran piksel dan jarak atau dimensi nyata, berdasarkan asumsi spesifik tentang geometri scene dan karakteristik kamera.

Nilai ini harus dikalibrasi secara akurat agar sesuai dengan karakteristik spesifik kamera dan setup yang digunakan. Kalibrasi yang tidak tepat akan menghasilkan pengukuran jarak yang tidak akurat, yang bisa berdampak pada keputusan navigasi kursi roda otonom. berikut rumus untuk pengkalibrasian nilai K:

$$k = \text{Jarak_nyata_objek} \times \text{Ukuran_objek_dalam_piksel} \quad (3.5)$$

Nilai k mungkin perlu disesuaikan jika kondisi lingkungan berubah, seperti perubahan pencahayaan yang mempengaruhi visibilitas atau ketepatan deteksi landmark oleh MediaPipe.

3. Perhitungan Lebar Obstacle Berdasarkan Pose

Dalam perhitungan lebar juga menggunakan perhitungan jarak euclidean namun yang membedakan adalah konteks spesifikasi aplikasi. Dimana kedua pendekatan ini berbeda dalam aplikasi dan konteksnya.

$$\text{width_pixel} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \times \text{scale_factor} \quad (3.6)$$

Dimana sesuai dengan dijelaskan sebelumnya nilai output yang didapatkan adalah pixel dari lebar manusia. untuk memetakan lebarnya pixel ke meter maka perlulah membuat sebuah rumus konversi yang menentukan ukuran dalam pixel (yang merupakan ukuran digital dan relatif dikonversikan ke ukuran dunia nyata (meter)). sehingga rumusnya menjadi :

$$\text{width_meters} = \text{width_pixel} \times \text{scale_factor} \quad (3.7)$$

Dengan mengalikan lebar objek dalam piksel dengan faktor skala, hasilnya adalah lebar objek dalam meter. Rumus ini berguna, misalnya, dalam aplikasi dimana perlu untuk mengetahui dimensi fisik objek dalam dunia nyata untuk membuat keputusan atau pengukuran yang tepat.

Faktor skala adalah nilai yang mengonversi ukuran dari unit piksel ke unit meter. Nilai ini diperoleh melalui proses kalibrasi. Faktor skala menentukan berapa meter yang diwakili oleh setiap piksel dalam gambar, berdasarkan jarak kamera ke objek dan pengaturan kamera lainnya seperti fokus panjang. Selain itu perlu diketahui bahwa faktor skala yang digunakan pada rumus ini berbeda dengan rumus jarak yang sebelumnya dijabarkan. berikut rumus untuk mengkalibrasi faktor skala dalam konteks lebar objek:

$$\text{scale_factor} = \frac{\text{dimensi_nyata_rata} - \text{rata}}{\text{ukuran_dalam_piksel_rata} - \text{rata}} \quad (3.8)$$

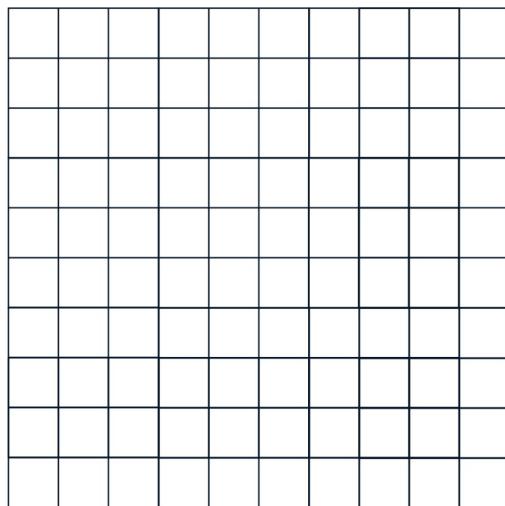
Nilai-nilai ini dapat digunakan dalam kode untuk mengonversi ukuran dalam piksel ke ukuran nyata berdasarkan pengukuran yang telah dikalibrasi. Dan sekali lagi pastikan kalibrasi dilakukan dengan baik dan benar agar mendapatkan hasil yang akurat.

3.2.6 Visualisasi hasil deteksi dalam Grid

Dalam tugas akhir ini, kursi roda otonom harus dapat mengetahui posisi obstacle yang akan dilaluinya. Oleh karena itu perlu dibuatnya sebuah peta yang dapat digunakan sebagai acuan kursi roda untuk mengambil tindakan berdasarkan jarak obstacle dan lebar obstacle yang

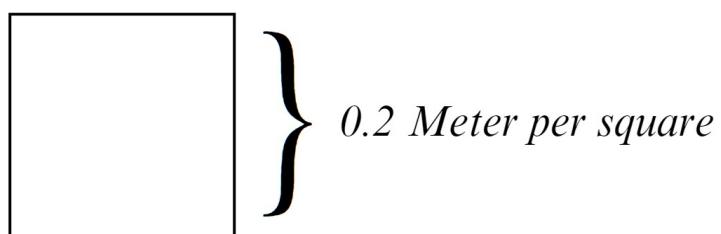
akan menjadi acuan untuk menjawab permasalan, yaitu di titik mana kursi roda harus menghindar dan berhasilkah obstacle dihindari. Grid menjadi salah satu pendekatan terbaik dalam memetakan hasil deteksi, dimana penggunaan grid tidak hanya mempermudah dalam pengambilan tindakan, namun grid juga memberikan visualisasi mudah untuk dimengerti. Dimana ukuran grid juga dapat dicustom sesuai dengan kebutuhan, baik dimensi maupun tampilannya.

Setelah perhitungan rumus diatas diimplementasikan dalam kode. Maka akan didapatkan beberapa variabel penting yang akan digunakan dalam memetakan posisi maupun ukuran obstacle dalam grid. sebelum itu berikut tampilan grid yang digunakan.



Gambar 3.16: Grid 10x10.

Penggunaan grid 10x10 didasari atas hasil citra dan performa model. Dimana baik bounding box maupun pose memiliki keterbatasan dimana hasil perhitungannya tidak melebihi parameter yang akan ditetapkan pada grid. Dimana parameter tersebut ialah sebagai berikut

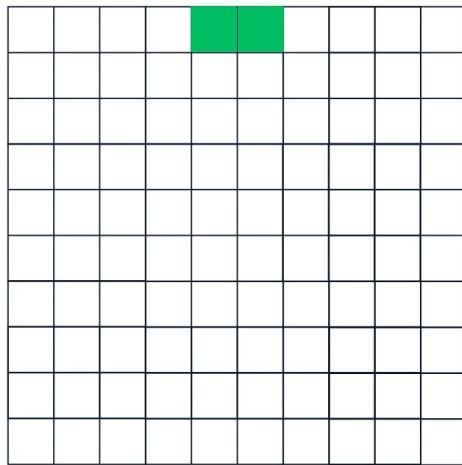


Gambar 3.17: Parameter untuk setiap kotak grid.

Dapat dilihat dari gambar diatas, merupakan tetapan yang didasari dari performa diatas yang dimana setiap kotak dalam grid tersebut akan bernilai 0.2 meter baik dalam posisi vertikal maupun horizontal.

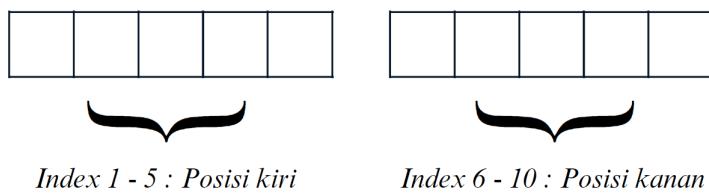
Agar dapat memetakan objek dengan baik dalam grid maka diperlukannya index untuk mengetahui posisi relatif kiri dan kanan objek terhadap kamera. Dimana untuk 10 kotak horizontal akan dibagi dimana Index 1 sampai 5 akan dikategorikan sebagai index kiri, dan index

6 hingga 10 akan dikategorikan dalam index kanan. pengambilan keputusan ini terkait dengan posisi kursi roda statis pada grid yang akan digambarkan sebagai berikut



Gambar 3.18: Posisi Kursi Pada grid.

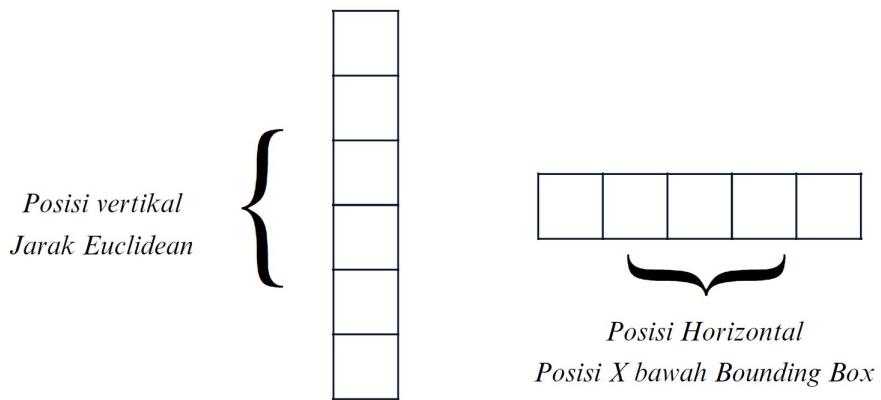
Dapat dilihat posisi kursi roda mengambil 2 kotak grid hijau pada bagian (5,1) dan (6,1) dimana keputusan ini didasarkan lebar kursi roda yang sesuai dengan ukuran grid ada. Pemilihan posisi atas menentukan index mana yang merupakan bagian kiri maupun kanan dalam grid. Dengan posisi atas ditetapkan sebagai posisi konstan kursi roda dan hasil input citra yang didapatkan bersebrangan dengan hasil deteksi (mirror) maka index dapat diposisikan sebagai berikut:



Gambar 3.19: Kategori Posisi berdasarkan Index.

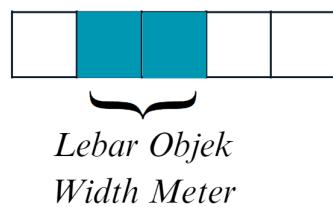
Pengkategorian ini berperan penting dalam pengambilan keputusan belok kursi roda dimana nantinya hasil deteksi yang berupa jarak , lebar serta posisi relatif akan ditampilkan pada grid.

Secara horizontal posisi objek hasil deteksi akan diambil berdasarkan Nilai perpindahan titik X bounding box bawah relatif terhadap pixel. Yang mana nilainya akan disesuaikan dengan Resolusi frame yang didapatkan oleh kamera. Dimana akan dibuat patokan untuk posisi tertentu yang melambangkan perpindahan sebesar 0.2 meter secara horizontal. Dengan asumsi bahwa kursi roda bergerak dan objek diam. maka hanya posisi objek yang terdeteksi didepanlah yang akan menjadi patokan penghindaran. Apabila objek terdeteksi namun saat berjalan maju objek menghilang maka kursi roda tidak perlu melakukan penghindaran. Oleh karena itu ditetapkan parameter jarak pada objek yang terdeteksi berdasarkan vertikal. Namun sebelum membahas lebih lanjut, akan dijelaskan secara rinci mengenai pemetaan hasil deteksi melalui variabel yang didapat.



Gambar 3.20: Variabel Dalam perpindahan Posisi terhadap Grid.

Berdasarkan Perhitungan rumus yang tadi didapatkan maka obstacle yang dideteksi dapat dipetakan posisinya relatif terhadap sumbu x dan y pada grid. Dimana keputusan mengambil variabel tersebut didasari oleh visibilitas hasil deteksi dan performa model dari hasil training. sehingga posisi yang dipetakan akan menjadi lebih akurat dan pengambilan keputusan dapat lebih konsisten. Namun dari posisi tersebut grid masih belum dapat merepresentasikan besar objek yang akan dilalui sehingga diperlukanlah sebuah variabel tambahan yang dapat membuat grid menggambarkan lebar objek relatif terhadap pixel yang nantinya akan dipetakan dalam grid. sehingga pada contoh gambar selanjutnya akan dijabarkan pemetaan lebar objek pada grid.



Gambar 3.21: Lebar Objek dalam grid.

Dapat dilihat pada gambar diatas pemetaan lebar berdasarkan nilai yang didapatkan pada variabel width meter. Dimana sesuai dengan parameter ukuran kotak yang telah ditentukan. Maka untuk setiap penambahan 0.2 meter ukuran lebar hasil deteksi yang didapatkan akan menambah jumlah kotak yang akan ditampilkan.

Adapun beberapa kondisi dimana perlu dilakukannya kalibrasi mengingat bounding box yang ditampilkan tidak sepenuhnya dapat dengan akurat merepresentasikan posisi objek dalam horizontal sehingga perlunya ditambahkan sebuah nilai kalibrasi yang akan berperan dalam mengatasi limitasi bounding box yang kadang memiliki nilai tengah yang tidak konsisten. Penambahan nilai pada rumus dibawah dapat menyesuaikan nilai posisi x pada grid agar lebih mendekati nilai kenyataan.

$$Grid_x = \left(\frac{posisi_x_bounding_box + konstanta}{scale_factor} \right) \quad (3.9)$$

Setelah pemetaan grid dilakukan selanjutnya akan ditampilkan parameter yang akan digunakan dalam pengambilan keputusan belok kursi roda.

3.2.7 Output Hasil deteksi

Dalam menentukan keputusan dalam pembelokan kursi roda akan ditambahkan beberapa parameter. Parameter ini berfungsi untuk menentukan Kapan kursi roda harus berbelok dan seberapa jauh kursi roda harus berbelok. Parameter tersebut akan dibagi menjadi beberapa status yang dimana kondisi-kondisi yang telah ditetapkan harus terpenuhi untuk dapat dikatakan memenuhi syarat.

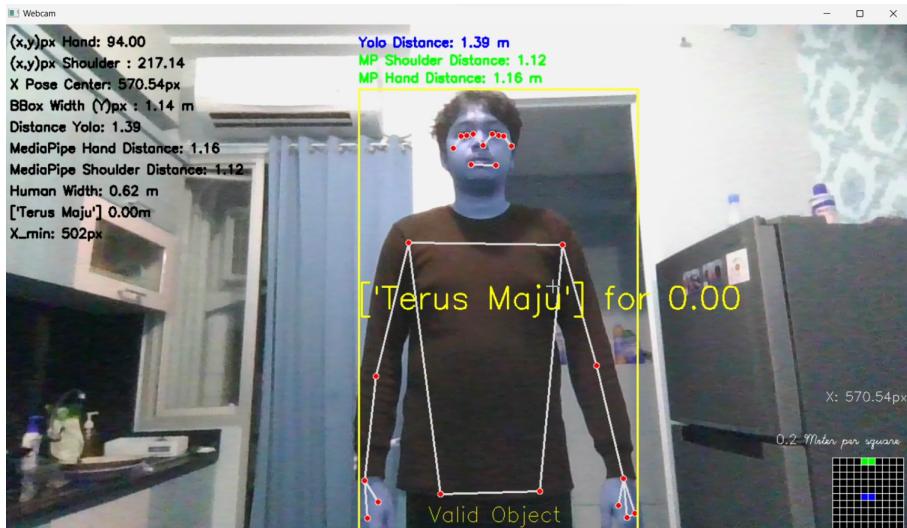
1. Valid Object Condition

Pada kondisi ini objek harus terdeteksi pada input citra untuk memenuhi kondisi ini. Tidak ada parameter khusus yang mengharuskan kursi roda untuk berbelok dalam kondisi ini, karena posisi kursi roda masih tergolong jauh dari posisi hasil deteksi. Dimana pengambilan keputusan yang akan dilakukan kursi roda adalah untuk tetap maju. Berikut merupakan contoh gambar untuk kondisi yang memenuhi.



Gambar 3.22: Contoh valid object diatas 0.8 Meter.

Dapat dilihat pada gambar diatas dimana objek hanya terdeteksi namun belum terdapat indikasi mendesak untuk kursi roda dalam mengambil keputusan karena posisi hasil deteksi yang masih tergolong jauh. adapun warna bounding box yang digambar berwarna biru sebagai penanda objek tergolong *safe distance* terhadap kursi roda. Sehingga instruksi yang akan dikirim adalah tetap maju.

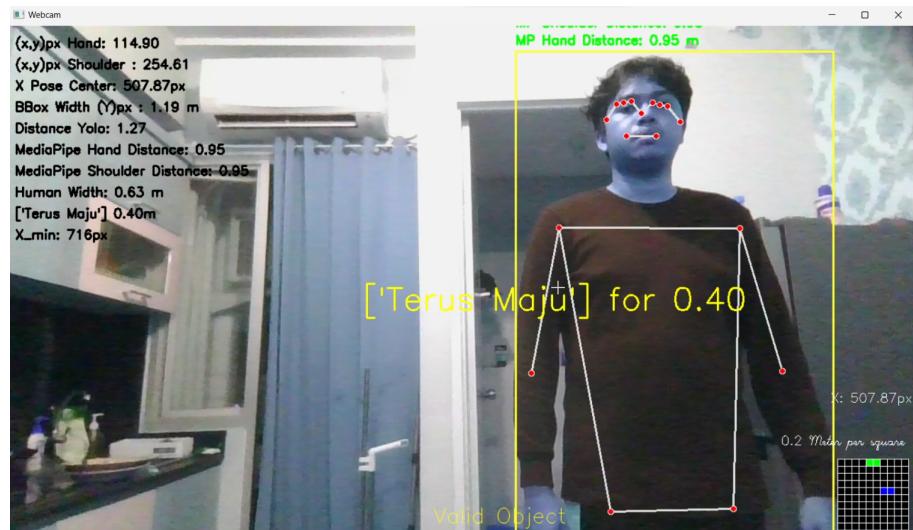


Gambar 3.23: Contoh valid object dibawah 0.8 Meter.

Saat hasil objek yang terdeteksi menjadi lebih dekat terhadap kursi roda maka bounding box akan berubah menjadi warna kuning yang menjadi indikasi bahwa objek yang terdeteksi sudah dibawah 0.8 Meter. sejauh ini pengambilan keputusan untuk belok belum dilakukan, namun akan menampilkan "valid object" pada bagian bawah layar kamera. Mengindikasikan objek benar-benar terdeteksi dan mulai mendekati. Dapat dilihat juga pada Grid di pojok kanan bawah bahwa perpindahan telah terjadi yang direpresentasikan berdasarkan perhitungan variabel yang ditampilkan pada pojok kanan atas layar kamera. Adapun beberapa contoh lainnya dapat dilihat digambar berikut.



Gambar 3.24: Contoh lain valid object dibawah 0.8 Meter condong kiri.



Gambar 3.25: Contoh lain valid object dibawah 0.8 Meter condong kanan.

2. Near Object Condition

Pada kondisi ini objek harus terdeteksi pada input citra untuk memenuhi kondisi ini dan objek harus berada dibawah 0.5 Meter jarak terdeteksi. Dalam kondisi ini akan dibagi menjadi 3 pengambilan keputusan. yaitu sebagai berikut:

Hasil deteksi objek pada grid menunjukan Index Kiri Lebih besar dari Index kanan

Pada kondisi ini posisi hasil deteksi menunjukkan nilai yang lebih besar pada Index kiri (Index 1-5) yang berarti bahwa objek sedang berada pada sebelah kiri posisi terhadap kursi roda. Dapat dilihat pada contoh gambar dibawah.



Gambar 3.26: Contoh kondisi Near Object Index Kiri >Kanan .

Dapat dilihat pada gambar, grid nilainya lebih mengarah index kiri dari pada kanan. Dimana dilihat dari pertimbangan posisi tersebut maka kursi roda akan menghindar ke Kanan yang

merupakan belokan yang lebih aman ketimbang belokan ke kiri.

Hasil deteksi objek pada grid menunjukan index Kanan lebih besar dari Index kiri

Pada kondisi ini posisi hasil deteksi menunjukkan nilai yang lebih besar pada index kanan (6-10) yang berarti bahwa objek sedang berada pada sebelah kanan posisi terhadap kursi roda. dapat dilihat pada contoh gambar dibawah

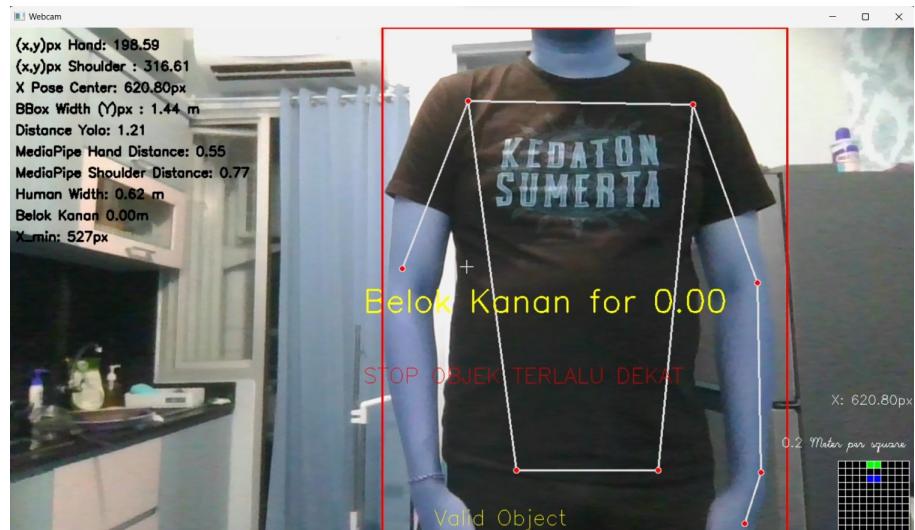


Gambar 3.27: Contoh kondisi Near Object Index Kanan >kiri

Dapat dilihat pada gambar, grid nilainya lebih mengarah index kanan dari pada kiri. Dimana dilihat dari pertimbangan posisi tersebut maka kursi roda akan menghindar ke kiri yang merupakan belokan yang lebih aman ketimbang belokan ke kanan.

Hasil deteksi objek pada grid menunjukan posisi Linear terhadap kursi roda

Pada kondisi ini perlu ditambahkan sebuah perintah untuk pengambilan keputusan dimana posisi belok pada kondisi ini baik melalui kanan maupun kiri tidak akan memiliki kelebihan/kekurangan karena dalam posisi ini nilai index sama besarnya baik kanan maupun kiri. sehingga perlu dilakukan pendekatan yang baik agar pengambilan keputusan ini tidak menimbulkan kesalahan. Pada tugas proyek ini pengambilan keputusan ini didasari pada nilai random. Sehingga keputusan yang diambil bisa ke kanan maupun ke kiri.



Gambar 3.28: Contoh kondisi Near Object Liniear.

Dapat dilihat pada gambar, gridnya sejajar dengan posisi kursi roda. Sehingga penggunaan random akan sangat berguna untuk mengambil keputusan apabila menghadapi kasus seperti ini.

[Halaman ini sengaja dikosongkan]

BAB IV

PENGUJIAN DAN ANALISIS

Pada penelitian ini dilakukan uji dan analisis dari model prediksi yang sudah dibuat. Pengujian ini dilakukan dengan beberapa skenario pengujian. Dalam pengujian ini akan diambil data yang akan merujuk pada performa sistem yang telah dibuat. Data - data yang diambil juga akan divisualisasikan untuk mendapatkan indikasi terkait hasil tugas akhir yang telah dilakukan.

4.1 Pengujian

Pada subbab ini dilakukan beberapa skenario pengujian terhadap model untuk mengetahui performa dan tingkat kesalahan yang dapat membuka kesempatan pengembangan penelitian di masa depan serta menarik kesimpulan secara keseluruhan. Berikut adalah skenario pengujian yang dilakukan.

4.1.1 Hasil Pengujian Performa menggunakan YOLOV8

Dalam pembuatan model yang digunakan, dilakukan pengambilan dan labeling data citra yang akan digunakan sebagai set data pelatihan atau set data *train*. Dimana data yang digunakan pada sistem yang akan dibuat menggunakan data citra manusia yang telah dijabarkan pada bab sebelumnya.

Data yang digunakan sebagai set data berjumlah 1735 data citra manusia. Dalam pengembangan model nantinya data citra ini akan dibagi menjadi 2 bagian dimana sebesar 1435 data citra sebagai set pelatihan dan sisanya sebesar 300 data citra akan digunakan sebagai set validasi. Setelah proporsi dataset ditentukan maka API key akan digenerate pada roboflow yang nantinya akan dipanggil untuk dilakukan proses training.

Setelah proses pemuatan set dilakukan, maka akan dilanjutkan pada proses training model. Proses training dilakukan dengan menggunakan beberapa parameter, yang dimana parameter ini akan dibandingkan performanya melalui nilai confusion matrix maupun nilai akurasi deteksi seperti presisi , recall, mapupun classification loss.

Pelatihan pertama menggunakan *100epoch* dan *batch size* sebesar 16 dengan tindakan praproses merubah ukuran gambar pada dataset menjadi 800x800 piksel dengan pemberian augmentasi berupa 90°rotasi dan noise. Penggunaan augmentasi ini menambah jumlah data citra yang semulanya 1435 menjadi 4305. Adapun tujuan dari pelatihan ini untuk mengetahui seberapa baik peningkatan model pre- trained yang digunakan dalam deteksi manusia dengan menggunakan dataset yang teraugmentasi.

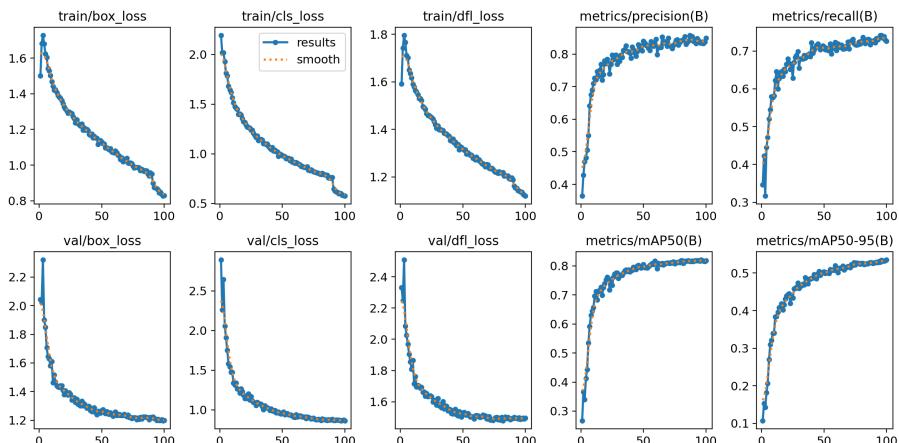
Didapatkan nilai box loss pada proses training menggunakan YOLOV8 ini adalah sebesar 0.819 setelah 100 epoch. Box loss mengukur seberapa baik model memprediksi bounding boxes seputar objek. Loss ini umumnya dihitung menggunakan metode seperti Intersection over Union (IoU) atau variasinya seperti CIoU atau DIoU, yang mengukur kesesuaian antara bounding box yang diprediksi dengan ground truth. Tujuan dari loss ini adalah untuk mengoptimalkan model agar bounding box yang diprediksi sesuai dengan posisi dan ukuran objek sebenarnya dalam gambar. Adapun penurunan yang nilai box loss yang didapatkan selama training

menandakan bahwa hasil training yang dilakukan telah berhasil membuat model menentukan koordinat bounding box pada object manusia dengan akurat.

Didapatkan nilai box loss pada proses validasi menggunakan YOLOV8 ini sebesar 1.2 , adapun box loss pada validasi ini mengindikasikan kemampuan mengenali objek pada data uji. Secara teori, penurunan object loss pada tahap validasi menandakan bahwa model mampu melakukan deteksi objek secara umum. bukan hanya pada data pelatihan.

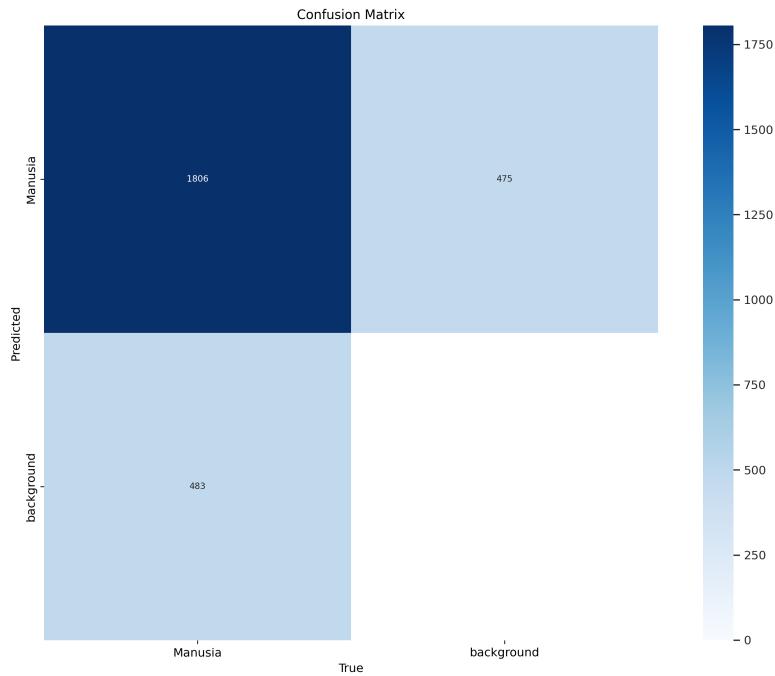
Skor mAP pada threshold 0.5 yang diperoleh pada proses ini adalah sebesar 80.9% dimana nilai ini mengindikasikan bahwa model memiliki kemampuan yang baik untuk mendeteksi objek dengan ketepatan yang tinggi, dengan syarat kriteria IoU minimal 0.5 terpenuhi. Ini mengindikasikan dalam banyak percobaan kasus, bounding box yang diprediksi oleh model memiliki overlap yang signifikan dengan bounding box ground truth.

Nilai nilai yang dicantumkan juga dapat dilihat lebih detail pada visualiasi berikut ini. Dimana nilai visualisasi cenderung turun pada Box loss dan cenderung naik pada skor mAP.



Gambar 4.1: Visualisasi Hasil training

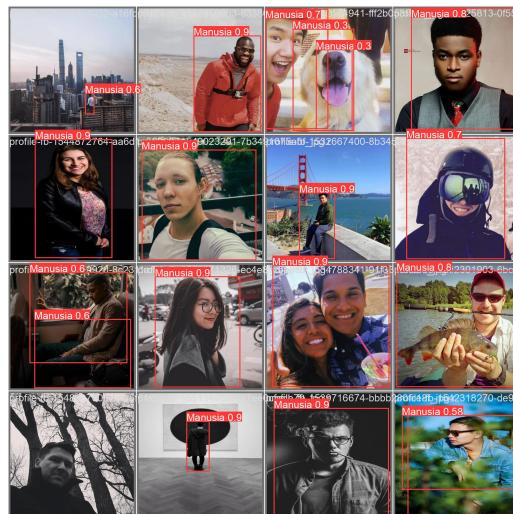
Adapun visualisasi melalui confusion matrix yang digunakan, untuk merepresentasikan hasil model lebih detail. Adapun indikator pada confusion matrix pada setiap kotaknya merepresentasikan nilai *true positive*, *false positive*, *false negative*, *true negative* sesuai dengan bahasan pada bab sebelumnya dapat dilihat pada gambar berikut ini.



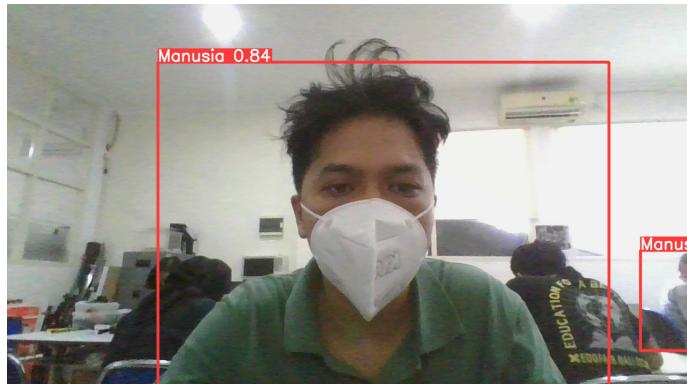
Gambar 4.2: Confusion Matrix Hasil Training

Dapat dilihat pada gambar dapat dilihat bahwa dari hasil klasifikasi model terdapat 1806 data citra yang termasuk dalam kategori true positive, 475 citra yang termasuk dalam kategori false positive dan 483 citra yang termasuk false negative.

Dilakukan pula tes inference model yang telah dilatih menggunakan set data test terhadap object manusia. Dapat dilihat pada gambar dibawah nilai confidence score yang cukup tinggi.



Gambar 4.3: Tes Inferensi Menggunakan Model Terlatih



Gambar 4.4: Tes Inferensi Menggunakan Model Terlatih

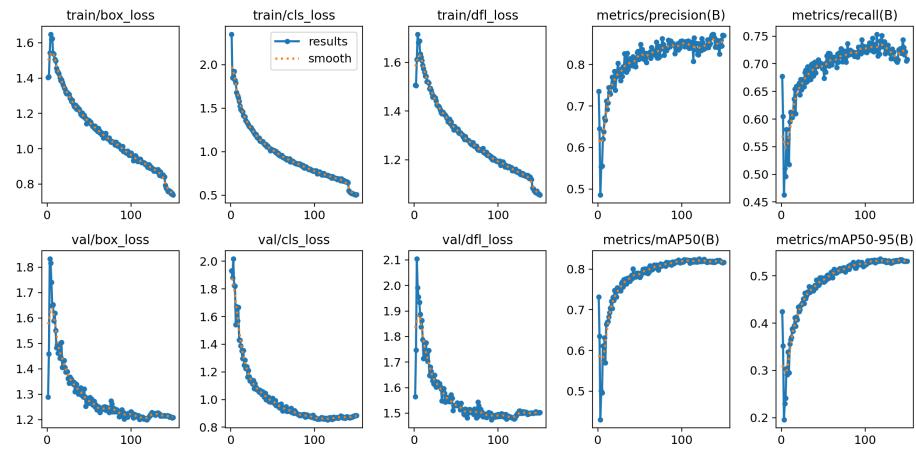
Pelatihan kedua menggunakan 150 *epoch* dan *batch size* sebesar 16 dengan tindakan pra proses merubah ukuran gambar pada dataset menjadi 800x800 piksel dengan pemberian augmentasi berupa 90°rotasi dan noise. Adapun tujuan dari pelatihan ini untuk mengetahui seberapa baik peningkatan model pre-trained yang digunakan dalam deteksi manusia dengan menggunakan dataset yang teraugmentasi.

Didapatkan nilai box loss pada proses training menggunakan YOLOV8 ini adalah sebesar 0.7383 setelah 150 epoch. Box loss mengukur seberapa baik model memprediksi bounding boxes seputar objek. Loss ini umumnya dihitung menggunakan metode seperti Intersection over Union (IoU) atau variasinya seperti CIoU atau DIoU, yang mengukur kesesuaian antara bounding box yang diprediksi dengan ground truth. Tujuan dari loss ini adalah untuk mengoptimalkan model agar bounding box yang diprediksi sesuai dengan posisi dan ukuran objek sebenarnya dalam gambar. Adapun penurunan yang nilai box loss yang didapatkan selama training menandakan bahwa hasil training yang dilakukan telah berhasil membuat model menentukan koordinat bounding box pada object manusia dengan akurat.

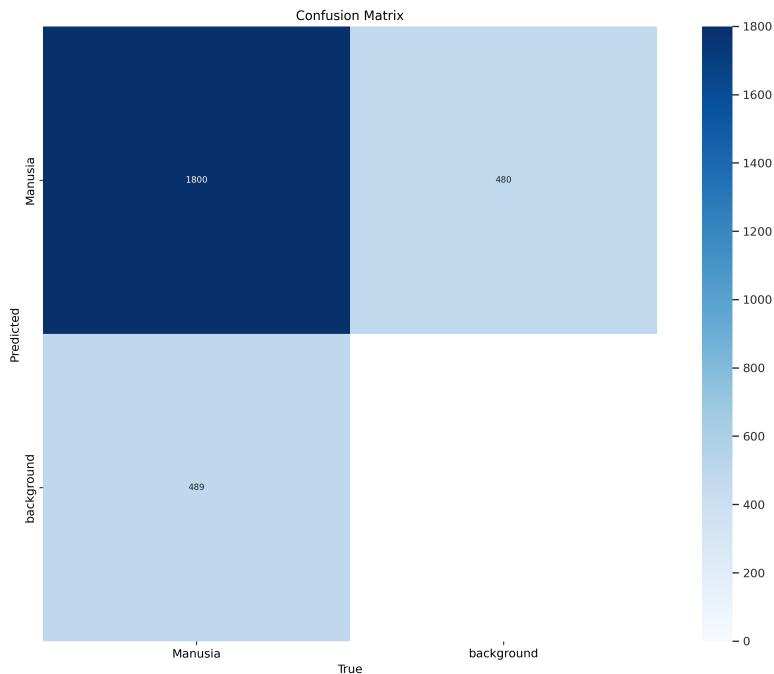
Didapatkan nilai box loss pada proses validasi menggunakan YOLOV8 ini sebesar 1.2143, adapun box loss pada validasi ini mengindikasikan kemampuan mengenali objek pada data uji. Secara teori, penurunan object loss pada tahap validasi menandakan bahwa model mampu melakukan deteksi objek secara umum. bukan hanya pada data pelatihan.

Skor mAP pada threshold 0.5 yang diperoleh pada proses ini adalah sebesar 81,7% dimana nilai ini mengindikasikan bahwa model memiliki kemampuan yang baik untuk mendeteksi objek dengan ketepatan yang tinggi, dengan syarat kriteria IoU minimal 0.5 terpenuhi. Ini mengindikasikan dalam banyak percobaan kasus, bounding box yang diprediksi oleh model memiliki overlap yang signifikan dengan bounding box ground truth.

Nilai nilai yang dicantumkan juga dapat dilihat lebih detail pada visualiasi berikut ini. Dimana nilai visualisasi cenderung turun pada Box loss dan cenderung naik pada skor mAP.



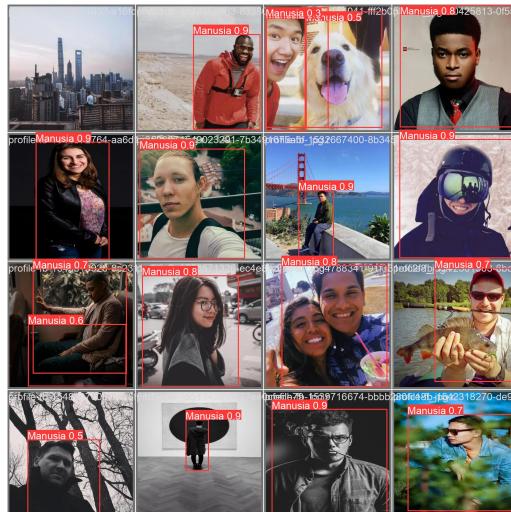
Gambar 4.5: Visualisasi Hasil training



Gambar 4.6: Confusion Matrix Hasil Training

Dapat dilihat pada gambar dapat dilihat bahwa dari hasil klasifikasi model terdapat 1800 data citra yang termasuk dalam kategori true positive, 480 citra yang termasuk dalam kategori false positive dan 489 citra yang termasuk false negative.

Dilakukan pula tes inference model yang telah dilatih menggunakan set data test terhadap object manusia. Dapat dilihat pada gambar dibawah nilai confidence score yang cukup tinggi.



Gambar 4.7: Tes Inferensi Menggunakan Model Terlatih



Gambar 4.8: Tes Inferensi Menggunakan Model Terlatih

Perbandingan Evaluasi

Dari kedua hasil training tersebut akan diambil yang terbaik berdasarkan pertimbangan akurasi. Dimana hasil dari confusion matrix kedua training menunjukkan hasil yang baik, untuk lebih memahami kemampuan dari model maka perlu dihitung menggunakan rumus evaluasi sebagai berikut :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1)$$

Sehingga dengan menginput masing nilai confusion matrix sebagai berikut :

```

1 # Asumsi FP dan TN 0 Karena tidak ada nilai prediksi lain kecuali kelas ←
    manusia
2 TP_100 = 1806
3 FN_100 = 483
4 TP_150 = 1800
5 FN_150 = 489
6 FP_150_assumed = 0
7 FP_100_assumed = 0

```

```
8  
9 accuracy_150_assumed = TP_150 / (TP_150 + FN_150 + FP_150_assumed)  
10 accuracy_100_assumed = TP_100 / (TP_100 + FN_100 + FP_100_assumed)  
11  
12 accuracy_150_assumed, accuracy_100_assumed
```

Dari hasil perhitungan diatas didapatkan nilai akurasi untuk masing training dengan epoch 100 dan 150 yaitu sebesar 78.90% dan 78.64%. Dari 2 nilai tersebut didapatkan indikasi bahwa mengurangi jumlah epoch tidak berpengaruh signifikan terhadap kemampuan model untuk men- genali manusia dengan benar, dan bahkan sedikit lebih efisien dalam hal ini. Ini bisa berguna un- tuk menghemat waktu dan sumber daya komputasi tanpa mengorbankan performa yang banyak. Sehingga dalam proses selanjutnya model dengan 100 epoch akan digunakan dalam melakukan pengujian pada NUC.

4.2 Deployment Pada Hardware

Proses Deployment yang dilakukan Pada bagian ini berupa evaluasi kinerja model yang diimplementasikan pada perangkat keras. Skenario untuk mengumpulkan data untuk pengujian dengan menggunakan kamera Logitech C920 yang dipasangkan pada kursi roda otomotif untuk mendeteksi manusia. Pengujian kinerja deployment dilakukan terhadap model yang dihasilkan pada pelatahan dengan 100 epoch. Dimana implementasi ini untuk mengukur:

1. Performa FPS sistem deteksi pada NUC.
2. Inference dan Response time pada sistem.
3. Tingkat keberhasilan kursi roda otomotif dalam menghindari manusia.

4.2.1 Performa Model Pada Intel NUC

Berikut hasil deployment yang dilakukan pada Intel NUC. Instalasi model pada NUC menggunakan venv yang tersedia pada Visual Studio code ide. Instalasi library perlu dilakukan untuk menjalankan model pada Intel NUC. Model output model berupa perintah yang dikir- imkan untuk menggerakan motor akan dicatat melalui perbandingan hasil pada terminal visual studio code dan pada Arduinoo ide. Berikut merupakan contoh data mentah yang dihasilkan dari output deteksi :

```
1 0: 480x800 1 Manusia, 102.8ms  
2 Waktu respons MediaPipe: 0.03261 detik  
3 FPS: 6.08  
4 Manusia di kiri  
5 Speed: 5.7ms preprocess, 102.8ms inference, 0.0ms postprocess per image ←  
       at shape (1, 3, 480, 800)  
6 Waktu respons YOLO: 0.000467 detik
```

Dapat dilihat nilai yang ditampilkan berupa Waktu response mediapipe, FPS dari sistem, posisi manusia , kecepatan preprocess, waktu inference pada sistem NUC, postprocess, dan waktu response YOLO.

Agar kita dapat dengan mudah membaca nilainya maka akan dilakukan proses olah data menggunakan python. Berikut merupakan teknik pengolahan data menjadi format CSV sebagai berikut:

```

1 import pandas as pd
2 import re
3
4 raw_data = """
5 0: 480x800 1 Manusia, 95.6ms
6 Waktu respons MediaPipe: 0.02218 detik
7 Jarak : 1.34
8 FPS: 6.37
9 Manusia di tengah
10 Speed: 5.2ms preprocess, 95.6ms inference, 0.0ms postprocess per image at↔
    shape (1, 3, 480, 800)
11 Waktu respons YOLO: 0.00047 detik
12 """
13
14 def parse_data_block(block):
15     lines = block.strip().split('\n')
16     parsed_data = {
17         'Inference': float(re.findall(r"(\d+\.\d+)ms", lines[0])[0]),
18         'MediaPipe Response Time': float(re.findall(r"(\d+\.\d+) detik", ←
19             lines[1])[0]),
20         'Jarak': float(re.findall(r"Jarak : (\d+\.\d+)", lines[2])[0]),
21         'FPS': float(re.findall(r"FPS: (\d+\.\d+)", lines[3])[0]),
22         'Position': lines[4],
23         'Preprocess Time': float(re.findall(r"(\d+\.\d+)ms preprocess", ←
24             lines[5])[0]),
25         'Postprocess Time': float(re.findall(r"(\d+\.\d+)ms postprocess", ←
26             lines[5])[0]),
27         'YOLO Response Time': float(re.findall(r"(\d+\.\d+) detik", lines←
28             [6])[0])
29     }
30     return parsed_data
31
32 blocks = raw_data.strip().split('\n\n')
33 data = [parse_data_block(block) for block in blocks]
34
35 df = pd.DataFrame(data)
36 csv_path = "/mnt/data/processed_data_v2.csv"
37 df.to_csv(csv_path, index=False)
38
39 csv_path, df.head()

```

Dengan menjalankan program tersebut akan didapatkan tabel yang berisikan Waktu response mediapipe, FPS dari sistem, posisi manusia , kecepatan preprocess, waktu inference pada sistem NUC, postprocess, dan waktu response YOLO. Nilai - nilai yang didapatkan dari program tersebut akan dilakukan pengujian pada bagian selanjutnya dan diambil kesimpulan.

4.2.2 Pengujian Berdasarkan FPS

Pengujian FPS akan dilakukan pada dua device yang pertama adalah pengujian pada device Laptop, dan pengujian pada device NUC. Dalam pengujian ini akan diambil nilai FPS sesuai dengan output yang didapatkan dan sudah diolah pada bagian sebelumnya.

Pengujian pada Laptop

Pada pengujian ini akan diambil nilai FPS pada laptop. Spesifikasi laptop yang digunakan pada pengujian ini merupakan sebagai berikut :

Tabel 4.1: Spesifikasi Laptop

Komponen	Spesifikasi
CPU	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz(8CPU)
RAM	16 GB DDR4-2666 SDRAM
GPU	NVIDIA® GeForce RTX™ 2060 with Max-Q design

Dapat dilihat pada tabel dibawah, didapatkan pada pengujian ini nilai rata - rata FPS sebesar 13.140. Dimana nilai FPS tertinggi adalah sebesar 13.23 dan FPS paling rendah ialah 13.05. Kestabilan nilai pada Pengujian ini menandakan bahwa performa sistem yang dijalankan pada laptop berjalan dengan sangat baik dan efisien.

Tabel 4.2: Tabel FPS pada Laptop

No	FPS
1	13.23
2	13.23
3	13.23
4	13.23
5	13.23
6	13.23
7	13.23
8	13.23
9	13.23
10	13.23
11	13.23
12	13.23
13	13.23
14	13.23
15	13.05
16	13.05
17	13.05
18	13.05
19	13.05
20	13.05
21	13.05
22	13.05
23	13.05
24	13.05
25	13.05
26	13.05
27	13.05
28	13.05
29	13.05
30	13.05

2. Pengujian pada NUC

Pada pengujian ini akan diambil nilai FPS pada NUC. Spesifikasi NUC yang digunakan pada pengujian ini merupakan sebagai berikut :

Tabel 4.3: Spesifikasi NUC

Komponen	Spesifikasi
CPU	Intel® Core™ i7-1165G7
RAM	32 GBLPDDR4
OS	Windows 11 Home Single Language

Dapat dilihat pada tabel dibawah, didapatkan pada pengujian ini nilai rata - rata FPS sebesar 6.111. Dimana nilai FPS tertinggi yang didapatkan ialah 6.47 dan FPS paling rendah ialah

5.54. Performa yang dihasilkan pada NUC tergolong stabil namun hasilnya kurang mendekati performa laptop.

Tabel 4.4: Tabel FPS pada NUC

No	FPS
1	6.37
2	6.43
3	6.43
4	5.57
5	5.54
6	5.79
7	5.83
8	6.42
9	6.47
10	6.47
11	6.47
12	6.42
13	6.42
14	6.42
15	6.42
16	6.43
17	6.42
18	6.42
19	6.47
20	6.42
21	6.42
22	6.43
23	6.42
24	6.42
25	6.47
26	6.42
27	6.42
28	6.37
29	6.43
30	6.33

4.2.3 Pengujian Berdasarkan Hasil Response Time

Untuk dapat melakukan pengujian hasil response time, maka obstacle harus memenuhi syarat Near Object Condition yang telah dibahas pada bab sebelumnya. Output yang dihasilkan setiap kali penghindaran berisi Arah dan time stamp pada Arduuino IDE dan Visual Studio code. Berikut merupakan contoh data mentah untuk kedua hasil output:

-
- ¹ 22:18:32.273 -> Stop
 - ² 22:18:32.948 -> Arah : E
 - ³ 22:18:33.131 -> Kanan
 - ⁴ 22:18:33.759 -> Arah : C

```
5 22:18:33.936 -> Stop
6 22:18:34.748 -> Arah : B
7 22:18:35.121 -> Maju
8 22:18:35.439 -> Arah : C
9 22:18:35.815 -> Stop
10 22:18:36.410 -> Arah : E
11 22:18:36.616 -> Kanan
```

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

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

Berdasarkan output diatas dapat dihitung Response Time sistem yang akan jabarkan pada tabel selanjutnya Hasil Response Time akan diuji untuk mendapatkan waktu yang dibutuhkan untuk melakukan pendekripsi dengan model yang kemudian diklasifikasi dan dikirim ke ESP32 hingga motor kursi roda mulai bergerak. Pengujian ini dilakukan secara real time pada perangkat NUC, perhitungan delay didapatkan dari mulai dikirim hingga berhentinya motor pada kursi roda. sedangkan perhitungan inference time dimulai dari dimulainya proses prediksi hingga didapatkan hasil dari proses klasifikasi. Rata - rata waktu delay yang didapatkan adalah 0.248 dari data yang sudah didapatkan dari pengujian NUC, adapun hasil nya dapat dilihat pada tabel dibawah. Adapun nilai inference rata-rata yang didapatkan adalah

Tabel 4.5: Hasil Pengujian Response Time

Inference	Sent	Receive	Motor	Delay	Arah
125.0	19:53:02.763	19:53:02.800	19:53:03.178	0.378	Stop
131.5	19:53:06.543	19:53:06.572	19:53:06.760	0.188	Maju
139.5	19:53:08.800	19:53:08.826	19:53:09.014	0.188	Stop
154.1	19:53:14.058	19:53:14.118	19:53:14.450	0.262	Kanan
131.6	19:53:17.932	19:53:17.974	19:53:18.162	0.188	Stop
141.9	19:53:19.645	19:53:19.667	19:53:19.855	0.188	Kiri
136.7	19:53:29.266	19:53:29.287	19:53:29.477	0.190	Stop
132.7	19:53:35.816	19:53:35.854	19:53:36.230	0.376	Maju
152.5	19:53:42.816	19:53:42.868	19:53:43.059	0.191	Stop
136.5	19:53:46.210	19:53:46.249	19:53:46.437	0.188	Kiri
141.8	19:53:55.093	19:53:55.124	19:53:55.314	0.190	Stop
140.1	19:54:01.549	19:54:01.586	19:54:01.766	0.180	Kanan
127.5	19:54:03.265	19:54:03.319	19:54:03.509	0.190	Stop
135.5	19:54:04.964	19:54:05.010	19:54:05.373	0.363	Kiri
142.5	19:54:20.862	19:54:20.886	19:54:21.073	0.187	Stop
136.6	19:54:22.398	19:54:22.440	19:54:22.819	0.379	Maju
123.1	19:54:24.094	19:54:24.133	19:54:24.323	0.190	Stop
143.4	19:54:32.486	19:54:32.517	19:54:32.713	0.196	Kiri
141.6	19:54:34.426	19:54:34.492	19:54:34.871	0.379	Stop
129.6	19:54:36.177	19:54:36.231	19:54:36.376	0.145	Kanan
152.1	19:54:37.870	19:54:37.923	19:54:38.111	0.188	Stop
134.2	19:54:40.144	19:54:40.177	19:54:40.546	0.369	Kiri
136.4	19:54:42.220	19:54:42.234	19:54:42.422	0.188	Stop
133.5	19:54:43.909	19:54:43.926	19:54:44.289	0.363	Kanan
137.1	19:54:51.301	19:54:51.333	19:54:51.523	0.190	Stop
140.1	19:54:54.763	19:54:54.796	19:54:55.142	0.346	Kiri
135.5	19:54:56.502	19:54:56.551	19:54:56.742	0.191	Stop
181.1	19:54:58.584	19:54:58.619	19:54:58.981	0.362	Maju
141.5	19:55:00.444	19:55:00.487	19:55:00.678	0.191	Stop
149.5	19:55:05.502	19:55:05.576	19:55:05.936	0.360	Kanan

4.2.4 Performa Keberhasilan

Pengujian ini dilakukan pengetesan terhadap kursi roda dalam menghindari Manusia secara real time. Pengetesan ini akan dilakukan Pada Tower 2 ITS. Pada pengujian ini diambil 20 kali sampel dan dicatat keberhasilan penghindaran. Manusia yang dideteksi Diam dan tidak bergerak. Hasil yang didapatkan sebagai berikut

Percobaan Ke	Hasil
1	Kursi Roda Berhasil Menghindar
2	Kursi Roda Berhasil Menghindar
3	Kursi Roda Berhasil Menghindar
4	Kursi Roda Berhasil Menghindar
5	Kursi Roda Berhasil Menghindar
6	Kursi Roda Berhasil Menghindar
7	Kursi Roda Berhasil Menghindar
8	Kursi Roda Berhasil Menghindar
9	Kursi Roda Berhasil Menghindar
10	Kursi Roda Berhasil Menghindar
11	Objek tidak terdeteksi dan Gagal Menghindar
12	Kursi Roda Berhasil Menghindar
13	Kursi Roda Berhasil Menghindar
14	Kursi Roda Berhasil Menghindar
15	Kursi Roda Berhasil Menghindar
16	Kursi Roda Berhasil Menghindar
17	Kursi Roda Berhasil Menghindar
18	Kursi Roda Berhasil Menghindar
19	Kursi Roda Berhasil Menghindar
20	Kursi Roda Berhasil Menghindar

Berdasarkan Hasil yang didapatkan Penghindaran berhasil sebanyak 19 kali dan gagal 1 kali pada percobaan ke 11. Adapun presentasi yang didapatkan dari hasil uji ini sebesar 98%

BAB V

PENUTUP

5.1 Kesimpulan

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

1. Model dengan Metrics tertinggi yang telah di-training dengan berbagai konfigurasi dan arsitektur adalah model dengan skor mAP di IoU 0.5 tertinggi sebesar 94.10% yang menggunakan arsitektur SSD-MobileNet-V2 FPN-Lite 320, skor tersebut lebih tinggi dibandingkan penelitian sebelumnya yang digunakan sebagai referensi
2. Performa NUC dalam pengujian FPS menghasilkan Nilai yang lebih rendah ketimbang Laptop pribadi Penulis dengan selisih 7.029
3. Rata - rata delay yang didapatkan pada pengujian adalah sekitar 0.248 dan rata- rata nilai inference yang didapatkan
4. Hasil Performa Deteksi menunjukkan hasil yang memuaskan dalam 10 sampel pengujian. Dengan presentasi keberhasilan sebesar 100%

5.2 Saran

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

1. Variasi dataset yang lebih ditingkat untuk meningkatkan performa deteksi
2. Menggunakan SBC yang memiliki performa yang lebih baik untuk fps yang lebih tinggi
3. Meningkatkan performa grid dengan mengubah ukuran 10x10 menjadi 100x100 atau lebih untuk hasil lebih detail

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

- Choi, J. H., Chung, Y., & Oh, S. (2019). Motion control of joystick interfaced electric wheelchair for improvement of safety and riding comfort. *Mechatronics*, 59, 104–114.
- Daring, K. (2016). Kbvi daring. <https://kbvi.kemdikbud.go.id/entri/lumpuh>
- Ekatama, I. (2024). *Perancangan sistem kontrol motor kursi roda secara nirkabel berbasis esp32* [Doctoral dissertation, Institut Teknologi Sepuluh Nopember].
- Elliot, R., on the south coast of England, R., & Follow. (2023, October). Nvidia-powered freedom: Kb's innovative autonomous wheelchair. <https://www.electromaker.io/blog/article/rusts-role-in-advancing-esp32-embedded-solutions>
- Kim, J.-W., Choi, J.-Y., Ha, E.-J., & Choi, J.-H. (2023). Human pose estimation using mediapipe pose and optimization method based on a humanoid model. *Applied Sciences*, 13(4). <https://www.mdpi.com/2076-3417/13/4/2700>
- Lecrosnier, L., Khemmar, R., Ragot, N., Decoux, B., Rossi, R., Kefi, N., & Ertaud, J.-Y. (2021). Deep learning-based object detection, localisation and tracking for smart wheelchair healthcare mobility. *International journal of environmental research and public health*, 18(1), 91.
- Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Ubweja, E., Hays, M., Zhang, F., Chang, C., Yong, M. G., Lee, J., Chang, W., Hua, W., Georg, M., & Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines. *CoRR, abs/1906.08172*. <http://arxiv.org/abs/1906.08172>
- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems [Software available from tensorflow.org]. <https://www.tensorflow.org/>
- Muhammad. (2018). Analisis driver h-bridge menggunakan relay pada motor bldc konstruksi axial flux (celah ganda). <https://repository.unej.ac.id/bitstream/handle/123456789/89217/Muhammad%20-%2020161910201115.pdf?sequence=1&isAllowed=y>
- Pansawira, P. (2022, April). Kelumpuhan - gejala, penyebab, dan mengobati - alodokter. <https://www.alodokter.com/kelumpuhan#:~:text=Kondisi%20ini%20dapat%20disebabkan%20oleh,Penanganan%20kelumpuhan%20tergantung%20pada%20penyebabnya>.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. <https://doi.org/10.1109/CVPR.2016.91>
- Shah, D. (2023). Intersection over union (iou): Definition, calculation, code.
- Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4), 1680–1716. <https://doi.org/10.3390/make5040083>
- Wijaya, W. K., Somawirata, I. K., & Labib, R. P. M. D. (2022). Deteksi objek menggunakan yolo v3 untuk keamanan pada pergerakan kursi roda elektrik. *Nucleus Journal*, 1(2), 94–98.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Kode Program

Sistem Penghindaran Menggunakan YoloV8

```
1 from ultralytics import YOLO
2 import cv2
3 import math
4 import mediapipe as mp
5 import numpy as np
6 import socket
7 import time
8 import datetime
9 from mediapipe.python.solutions.pose import PoseLandmark
10
11 def delay(seconds):
12     start_time = time.time()
13     while time.time() - start_time < seconds:
14         pass
15
16 host = "192.168.4.1" # Set to ESP32 Access Point IP Address
17 port = 80
18
19 mp_pose = mp.solutions.pose
20 pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.6)
21 mp_drawing = mp.solutions.drawing_utils
22
23
24 cap = cv2.VideoCapture(0)
25 cap.set(3, 1920)
26 cap.set(4, 1080)
27
28 k = 10.311
29 k2 = 24.222
30
31 focal_length_pixel = 481
32 tinggi_objek_nyata = 181
33
34
35 frame_count = 0
36 start_time = time.time()
37 fps = 0
38
39 model = YOLO("best2.pt")
40
41 def hitung_jarak(tinggi_bounding_box, focal_length_pixel, tinggi_objek_nyata):
42     if tinggi_bounding_box == 0:
43         return float('inf')
44     jarak = (tinggi_objek_nyata * focal_length_pixel) / tinggi_bounding_box
```

```

45     return jarak / 100 # Konversi ke meter
46
47 def hitung_lebar_objek(lebar_bounding_box, jarak_objek, ←
48     focal_length_pixel):
49     if lebar_bounding_box == 0 or focal_length_pixel == 0:
50         return 0
51     lebar_objek = (lebar_bounding_box * jarak_objek) / focal_length_pixel
52     return lebar_objek
53
54 def convert_coordinates(outputs, img_width, img_height):
55     boxes = []
56     for detection in outputs:
57         x_center, y_center, width, height = detection['x_center'], ←
58             detection['y_center'], detection['width'], detection['height']
59
60         x_min = (x_center - width / 2) * img_width
61         y_min = (y_center - height / 2) * img_height
62         x_max = (x_center + width / 2) * img_width
63         y_max = (y_center + height / 2) * img_height
64
65         boxes.append((x_min, y_min, x_max, y_max))
66     return boxes
67
68 # Fungsi untuk menghitung jarak Euclidean dalam piksel
69 def hitung_jarak_euclidean(landmark1, landmark2, lebar_img):
70     jarak_pix = math.sqrt((landmark1.x - landmark2.x)**2 + (landmark1.y←
71     - landmark2.y)**2) * lebar_img
72     return jarak_pix
73
74 def hitung_lebar_mediapipe(pose_results, lebar_img):
75     if pose_results.pose_landmarks:
76         bahu_kiri = pose_results.pose_landmarks.landmark[mp_pose.←
77             PoseLandmark.LEFT_SHOULDER]
78         bahu_kanan = pose_results.pose_landmarks.landmark[mp_pose.←
79             PoseLandmark.RIGHT_SHOULDER]
80         jarak_pix = hitung_jarak_euclidean(bahu_kiri, bahu_kanan, ←
81             lebar_img)
82
83         faktor_konversi = 0.00087 # 0.087cm per piksel
84         lebar_m = jarak_pix * faktor_konversi
85
86         return lebar_m
87     return 0
88
89 def hitung_jarak_vertikal(pose_results, tinggi_img):
90     if pose_results.pose_landmarks:
91         pusar = pose_results.pose_landmarks.landmark[mp_pose.PoseLandmark←
92             .NOSE]
93         # Misal kita asumsikan pusar sebagai acuan vertikal (bisa diganti←
94         # sesuai dengan landmark yang sesuai)
95         jarak_vertikal = (1 - pusar.y) * tinggi_img # Hitung jarak dari ←
96             atas gambar ke pusar
97         return jarak_vertikal
98     return 0
99
100 def draw_grid(img, detection_status, camera_position):
101     grid_size = 100 # Ukuran grid di sudut kanan bawah

```

```

93     start_x = img.shape[1] - grid_size - 10 # Posisi X grid
94     start_y = img.shape[0] - grid_size - 10 # Posisi Y grid
95
96     cell_size = int(grid_size / 10) # Membagi grid menjadi 10x10
97
98     # Menggambar sel grid dengan latar belakang putih
99     for i in range(10):
100         for j in range(10):
101             cell_color = (255, 0, 0) if detection_status[i][j] else (0, ←
102                             0, 0)
103             cv2.rectangle(img, (start_x + j * cell_size, start_y + i * ←
104                           cell_size),
105                         (start_x + (j + 1) * cell_size, start_y + (i + ←
106                           1) * cell_size), cell_color, -1)
107
108     # Menandai posisi kamera dengan kotak hijau
109     for pos in camera_position:
110         x, y = pos
111         cv2.rectangle(img, (start_x + x * cell_size, start_y + y * ←
112                           cell_size),
113                         (start_x + (x + 1) * cell_size, start_y + (y + 1) *←
114                           cell_size), (0, 255, 0), -1)
115
116     # Menggambar garis grid
117     for i in range(11):
118         cv2.line(img, (start_x, start_y + i * cell_size), (start_x + ←
119                         grid_size, start_y + i * cell_size), (255, 255, 255), 1)
120         cv2.line(img, (start_x + i * cell_size, start_y), (start_x + i * ←
121                         cell_size, start_y + grid_size), (255, 255, 255), 1)
122
123         text_x2 = start_x - 10
124         text_x = start_x - 80
125         text_y = start_y - 20 # Menempatkan teks 20 piksel di atas grid
126         text_y2 = start_y - 80
127         cv2.putText(img, "0.2 Meter per square", (text_x, text_y), cv2.←
128                     FONT_HERSHEY_SCRIPT_SIMPLEX, 0.6, (255, 255, 255), 1)
129         cv2.putText(img, f"X: {posisi_horizontal_piksel:.2f}px", (text_x2, ←
130                         text_y2), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1)
131
132     return img
133
134 def determine_direction(detection_grid, grid_size=10):
135     mid_point = grid_size // 2
136     left_count = np.sum(detection_grid[:, :mid_point])
137     right_count = np.sum(detection_grid[:, mid_point:])
138     manusia = None
139
140     #if jarak_mediapipe <0.7:
141     if left_count > right_count:
142         direction = 'Kiri'
143         manusia = "Manusia di Kanan"
144
145     elif right_count > left_count:
146         direction = 'Kanan'
147         manusia = "Manusia di Kiri"
148
149     else:

```

```

141     direction = 'Kanan'
142     manusia = "Manusia di tengah"
143 #else:
144 #    direction = ("Terus Maju")
145
146
147     return direction, abs(right_count - left_count) * 0.2, manusia # ←
148         Misalkan setiap kotak mewakili 0.2 meter
149
150 # Inisialisasi grid deteksi
151 detection_grid = np.zeros((10, 10), dtype=bool)
152 camera_position = [(4, 0), (5, 0)] # Posisi kamera pada grid (5,10) dan ←
153 (6,10)
154 newtex = None
155 while True:
156     success, img = cap.read()
157     if not success:
158         break # Jika tidak berhasil membaca frame, keluar dari loop
159
160     frame_count += 1
161     current_time = time.time()
162
163     # Hitung dan tampilkan FPS setiap satu detik
164     if current_time - start_time >= 1:
165         fps = frame_count / (current_time - start_time)
166         frame_count = 0
167         start_time = current_time
168         print(f"FPS: {fps:.2f}")
169
170     detection_grid.fill(False)
171     detection_grid[0, 4] = True # Pos 5,10 dalam 0-index
172     detection_grid[0, 5] = True # Pos 6,10 dalam 0-index
173     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
174
175     start_time_yolo = time.time()
176
177     results = model.predict(img, stream=True)
178
179     end_time_yolo = time.time()
180
181     response_time_yolo = end_time_yolo - start_time_yolo
182     print(f"Waktu respons YOLO: {response_time_yolo:.5f} detik")
183
184 #print waktu
185     for r in results:
186         boxes = r.boxes
187
188         for box in boxes:
189             confidence = math.ceil((box.conf[0] * 100)) / 100 # ←
190                 Menghitung confidence
191             if confidence > 0.7: # Memeriksa apakah confidence di atas ←
192                 0.6
193                 cls = int(box.cls[0]) # Mengonversi kelas objek ←
194                     terdeteksi menjadi integer

```

```

192     if cls == 0: # Memeriksa apakah objek terdeteksi adalah ←
193         'person'
194         x1, y1, x2, y2 = box.xyxy[0] # Mengambil koordinat ←
195             kotak pembatas objek
196         x1, y1, x2, y2 = map(int, [x1, y1, x2, y2])
197         color = (255, 0, 0) #inisialisasi warna
198
199         # Menggunakan YOLO untuk menghitung jarak
200         tinggi_bounding_box = y2 - y1
201         lebar_bounding_box = x2 - x1
202         jarak_yolo = hitung_jarak(tinggi_bounding_box, ←
203             focal_length_pixel, tinggi_objek_nyata)
204         # Menghitung lebar objek
205         lebar_objek = hitung_lebar_objek(lebar_bounding_box, ←
206             jarak_yolo, focal_length_pixel)
207         cv2.putText(img, f"X_min: {x1}px", (10, 300), cv2.←
208             FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
209         cv2.putText(img, f"BBox Width (Y)px : {lebar_objek:.2←
210             f} m", (10, 120), cv2.FONT_HERSHEY_SIMPLEX, 0.6, ←
211             (255, 255, 255), 2)
212         cv2.putText(img, f"Yolo Distance: {jarak_yolo:.2f} m"←
213             , (x1, y1 - 60), cv2.FONT_HERSHEY_SIMPLEX, 0.6, ←
214             (255, 0, 0), 2)
215         cv2.putText(img, f"Distance Yolo: {jarak_yolo:.2f}", ←
216             (10, 150), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, ←
217             255, 255), 2)
218
219         start_time_mediapipe = time.time()
220
221         # Crop gambar untuk analisis pose dengan MediaPipe
222         person_img = img[y1:y2, x1:x2]
223         person_img_rgb = cv2.cvtColor(person_img, cv2.←
224             COLOR_BGR2RGB)
225         pose_results = pose.process(person_img_rgb)
226
227         end_time_mediapipe = time.time()
228         response_time_mediapipe = end_time_mediapipe - ←
229             start_time_mediapipe
230         print(f"Waktu respons MediaPipe: {←
231             response_time_mediapipe:.5f} detik")
232         if pose_results.pose_landmarks:
233             # Menggambar landmarks pose
234             mp_drawing.draw_landmarks(img[y1:y2, x1:x2], ←
235                 pose_results.pose_landmarks, mp_pose.←
236                 POSE_CONNECTIONS)
237
238             # Menggunakan MediaPipe untuk estimasi jarak
239             lebar_img = person_img.shape[1]
240             siku_kanan = pose_results.pose_landmarks.landmark←
241                 [mp_pose.PoseLandmark.RIGHT_ELBOW]
242             pergelangan_kanan = pose_results.pose_landmarks.←
243                 landmark[mp_pose.PoseLandmark.RIGHT_WRIST]
244             bahu_kanan = pose_results.pose_landmarks.landmark←
245                 [mp_pose.PoseLandmark.RIGHT_SHOULDER]
246             bahu_kiri = pose_results.pose_landmarks.landmark[←
247                 mp_pose.PoseLandmark.LEFT_SHOULDER]

```

```

228     pusar = pose_results.pose_landmarks.landmark[←
229         mp_pose.PoseLandmark.NOSE]
230     jarak_pixbahu = hitung_jarak_euclidean(bahu_kanan←
231         , bahu_kiri, lebar_img)
232     jarak_pix = hitung_jarak_euclidean(siku_kanan, ←
233         pergelangan_kanan, lebar_img)
234     lebar_img = img.shape[1]
235     tinggi_img = img.shape[0]
236     lebar_m = hitung_lebar_mediapipe(pose_results, ←
237         lebar_img)
238     grid_size = 10 # Ukuran grid (10x10)
239     jarak_maksimum = 10 * 0.2
240     #max_jarak = grid_size * 0.2 # Jarak maksimal ←
241     # yang dapat diwakili grid (2 meter dalam kasus ←
242     # ini)
243
244     cv2.putText(img, f"Human Width: {lebar_m:.2f} m", ←
245         (10, 240), cv2.FONT_HERSHEY_SIMPLEX, 0.6, ←
246         (255, 255, 255), 2)
247
248     if jarak_pix > 0:
249         jarak_mediapipe = (k / jarak_pix) * 10 # ←
250             Menyesuaikan dengan faktor kalibrasi
251         jarak_mediapipebahu = (k2 / jarak_pixbahu) * ←
252             10
253
254         # Menghitung posisi grid berdasarkan jarak ←
255             MediaPipe
256         posisi_horizontal_piksel = pose_results.←
257             pose_landmarks.landmark[mp_pose.←
258                 PoseLandmark.NOSE].x * lebar_img
259         grid_x = int(((x1 + 175) / lebar_img) * 10)
260         pusar = pose_results.pose_landmarks.landmark[←
261             mp_pose.PoseLandmark.NOSE]
262         posisi_vertikal_piksel = pusar.y * tinggi_img
263         grid_y = int((jarak_maksimum - ←
264             jarak_mediapipe) / 0.2)
265         #grid_y = grid_size - 1 - int(jarak_mediapipe←
266             / 0.2)
267         grid_y = max(0, min(9 - grid_y, 9))
268         #grid_y = max(0, min(grid_y, grid_size - 1))
269         lebar_grid = max(1, int(lebar_m / 0.2))
270
271         grid_x = min(max(grid_x, 0), 9)
272         # Menghitung jumlah kotak berdasarkan lebar ←
273             dalam meter
274         # Setidaknya satu kot
275         for i in range(max(0, grid_x - lebar_grid // ←
276             2), min(10, grid_x + lebar_grid // 2)):
277             for j in range(max(0, grid_y), min(10, ←
278                 grid_y + lebar_grid // 2)):
279                 detection_grid[j][i] = True
280
281
282         cv2.putText(img, f"MP Hand Distance: {←
283             jarak_mediapipe:.2f} m", (x1, y1 - 10), ←
284             cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0)←
285             , 2)

```

```

263         cv2.putText(img, f"MediaPipe Hand Distance: {←
264             jarak_mediapipe:.2f}", (10, 180), cv2.←
265             FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255)←
266             , 2)
267         cv2.putText(img, f"MP Shoulder Distance: {←
268             jarak_mediapipebahu:.2f}", (x1,y1 - 35), ←
269             cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0), ←
270             2)
271         cv2.putText(img, f"MediaPipe Shoulder ←
272             Distance: {jarak_mediapipebahu:.2f}", (10 ←
273             , 210), cv2.FONT_HERSHEY_SIMPLEX, 0.6, ←
274             (255,255,255), 2)
275         cv2.putText(img, f"FPS: {fps:.2f}", (10, 330)←
276             , cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, ←
277             255, 255), 2)

278     # Menambahkan tampilan jarak piksel di sudut ←
279     # kiri atas
280     cv2.putText(img, f"(x,y)px Hand: {jarak_pix←
281         :.2f}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX←
282         , 0.6, (255, 255, 255), 2)
283     cv2.putText(img, f"(x,y)px Shoulder : {←
284         jarak_pixbahu:.2f}", (10, 60), cv2.←
285         FONT_HERSHEY_SIMPLEX, 0.6,(255,255,255),2 ←
286         )
287     cv2.putText(img, f"X Pose Center: {←
288         posisi_horizontal_piksel:.2f}px", (10, 90)←
289         , cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, ←
290         255, 255), 2)

291     # Mengubah warna bounding box menjadi merah ←
292     # jika kedua jarak terpenuhi
293     #if jarak_yolo < 1.3 or jarak_mediapipe < ←
294         1.3:
295         # color = (51, 255, 255)
296         #pesan = "Valid Object"
297         #cv2.putText(img,pesan,(600, 700), cv2.←
298             FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255)←
299             , 1)

300     direction, distance = determine_direction(←
301         detection_grid)
302     if jarak_mediapipe < 1.0 and direction == 'Kiri': #or←
303         lebar_bounding_box > 500:
304         pesan = "BELOK KIRI"
305         color = (0, 0, 255)
306         #pesanSend = 'BELOK KIRI'
307         # cv2.putText(img,pesan2,(500, 500), cv2.←
308             FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 1)
309         # pesan = determine_direction(detection_grid,←
310             grid_size=10)
311     elif jarak_mediapipe < 0.5 and direction == 'Kanan':
312         pesan = "BELOK KANAN"
313         color = (0,0,255)

314     else:
315         pesan = "MAJU"
316         color = (51,255,255)

```

```

292     #pesan = determine_direction(detection_grid, ←
293     #grid_size=10)
294     if pesan != newtex:
295         with socket.socket(socket.AF_INET, socket.←
296                             SOCK_STREAM) as s:
297             s.connect((host, port))
298
299             delay(0.5)
300
301             arah = 'C\n'
302             s.send(arah.encode('utf-8'))
303             time.sleep(1)
304
305             if pesan == "BELOK KIRI":
306                 arah = 'A\n'
307                 date = datetime.datetime.now()
308                 print(date)
309                 print("kiri")
310             elif pesan == "BELOK KANAN":
311                 arah = 'E\n'
312                 date = datetime.datetime.now()
313                 print(date)
314                 print("kanan")
315             else:
316                 arah = 'B\n'
317                 date = datetime.datetime.now()
318                 print(date)
319                 s.send(arah.encode('utf-8'))
320                 newtex = pesan
321
322             # Gambar bounding box YOLO
323             direction, distance, manusia = determine_direction(←
324                         detection_grid)
325             cv2.putText(img, f"{pesan} {distance:.2f}m", (10, ←
326                         270), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, ←
327                         255), 2)
328             cv2.putText(img, f"{pesan} for {distance:.2f}"←
329                         ,(500,400 ), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, ←
330                         255, 255), 2 )
331             cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)
332             print (manusia)
333             img = draw_grid(img, detection_grid, camera_position)
334
335             cv2.imshow('Webcam', img)
336             if cv2.waitKey(1) & 0xFF == ord('q'):
337                 break
338
339             cap.release()
340             cv2.destroyAllWindows()

```

Program Kalibrasi Lengan

BIOGRAFI PENULIS



I Gst Ngr Agung Hari Vijaya Kusuma, Atau yang biasa dikenal sebagai Agung Hari, lahir pada 8 Oktober 2000 di kota denpasar. Penulis merupakan anak pertama dari 3 bersaudara. Setelah lulus dari SMA Negeri 4 Denpasar. Penulis kemudian melanjutkan pendidikan ke jenjang strata satu di Departemen Teknik Komputer ITS, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember mulai tahun 2019.

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

Cloud Computing dan Orbit Future Academy.

[Halaman ini sengaja dikosongkan]