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 196806011995121 009
Dr. Arief Kurniawan, S.T., M.T.
NIP 197409072002121 001

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

Dr. Arief Kurniawan, S.T., M.T.

NIP 19740907200212 1 001

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

Dr. Arief Kurniawan, S.T., M.T.

NIP 19740907200212 1 001

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) .....
Dr. Arief Kurniawan, S.T., M.T. NIP: 19740907200212 1 001	(Pembimbing II) .....
Dr. Diah Puspito Wulandari, S.T., M.Sc.. NIP: 19801219200501 2 001	(Penguji I) .....
Arta Kusuma Hernanda, S.T., M.T.. NIP: 1996202311024	(Penguji II) .....
. NIP:	(Penguji III) .....

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T..
NIP. 19700313199512 1 001

SURABAYA
Juli, 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. NIP: 19680601199512 1 009	(Advisor I) .....
Dr. Arief Kurniawan, S.T., M.T. NIP: 19740907200212 1 001	(Advisor II) .....
Dr. Diah Puspito Wulandari, S.T., M.Sc.. NIP: 19801219200501 2 001	(Examiner I) .....
Arta Kusuma Hernanda, S.T., M.T.. NIP: 1996202311024	(Examiner II) .....
. NIP:	(Examiner III) .....

Acknowledged,
Head of Computer Engineering Department ELECTICS - ITS

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T..
NIP. 19700313199512 1 001

SURABAYA
July, 2024

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa / NRP	:	I Gst Ngr Agung Hari Vijaya Kusuma / 07211940000073
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, July 2024

Mengetahui
Dosen Pembimbing

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

Mahasiswa

I Gst Ngr Agung Hari Vijaya Kusuma
NRP. 07211940000073

[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, July 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. Dr. Arief Kurniawan, S.T., M.T.

Pengembangan kursi roda otonom telah menjadi semakin penting dalam meningkatkan mobilitas 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 Intel NUC dan Kamera. Obstacle yang terdeteksi membuat NUC mengirim perintah ke ESP32 untuk menjalankan motor untuk melakukan manuver penghindaran. Pengujian performa keberhasilan penghindaran dilakukan dengan 30 kali percobaan pada objek manusia yang diam. Hasil pengujian menunjukkan bahwa kursi roda berhasil menghindar sebanyak 30 kali tanpa gagal, memberikan tingkat keberhasilan sebesar 100%. Hal ini menunjukkan bahwa sistem kursi roda otonom yang dirancang mampu melakukan penghindaran rintangan dengan sangat baik.

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.
coadvisor		2. Dr. Arief Kurniawan, S.T., M.T.

The development of autonomous wheelchairs has become increasingly important in improving mobility for individuals with limited mobility. This study proposes the development of a YOLOv8-based autonomous wheelchair system for obstacle avoidance, specifically focusing on human obstacle detection. By 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 and a camera. When an obstacle is detected, the NUC sends a command to the ESP32 to operate the motor to perform avoidance maneuvers. Performance testing of the avoidance success was conducted with 30 trials on stationary human objects. The test results showed that the wheelchair successfully avoided obstacles 30 times, providing a success rate of 100%. This indicates that the designed autonomous wheelchair system is capable of performing obstacle avoidance without mistake.

Keywords : ***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. dan Bapak Dr. Arief Kurniawan, 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. Orang tua saya tercinta yang selalu mendukung dan senantiasa menyayangi saya sedari kecil hingga dewasa.
6. Teman - teman lab B300 dan B201 serta teman - teman Departemen Teknik Komputer lainnya

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

Surabaya, Juli 2024

I Gst Ngr Agung Hari Vijaya Kusuma

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR PROGRAM

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Mobilitas merupakan aspek penting dalam kehidupan sehari-hari yang memungkinkan individu untuk melakukan berbagai aktivitas dengan mandiri. Namun, bagi individu yang mengalami kelumpuhan atau gangguan mobilitas lainnya, ketergantungan pada alat bantu seperti kursi roda menjadi hal yang tidak terhindarkan. Menurut Kamus Besar Bahasa Indonesia (KBBI), lumpuh adalah kondisi kehilangan fungsi gerak pada bagian tubuh tertentu yang dapat disebabkan oleh berbagai faktor, seperti cedera atau penyakit saraf [1]. Dalam konteks medis, kelumpuhan dapat terjadi akibat kerusakan pada sistem saraf, baik itu saraf pusat maupun saraf tepi, yang mengakibatkan hilangnya kontrol motorik pada anggota tubuh [2].

Perkembangan teknologi dalam bidang ini telah memberikan kontribusi signifikan dalam meningkatkan kualitas hidup penyandang disabilitas. Salah satu inovasi yang menonjol adalah pengembangan kursi roda elektrik yang dikendalikan melalui joystick. Penelitian oleh Choi, Chung, dan Oh menunjukkan bahwa kontrol gerak kursi roda elektrik yang diintegrasikan dengan joystick dapat meningkatkan keselamatan dan kenyamanan pengguna [3]. Meskipun demikian, tantangan tetap ada dalam hal navigasi dan penghindaran rintangan, terutama di lingkungan yang dinamis dan kompleks.

Seiring dengan kemajuan teknologi kecerdasan buatan, deep learning telah menjadi alat yang sangat efektif dalam mendeteksi, mengidentifikasi, dan melacak objek secara real-time. Lecrosnier dalam penelitiannya mengungkapkan bahwa penggunaan deep learning dalam deteksi dan pelokalisasian objek pada kursi roda pintar dapat meningkatkan mobilitas dan keselamatan pengguna di lingkungan kesehatan [4]. Teknologi ini memungkinkan kursi roda untuk secara otomatis menghindari rintangan, termasuk manusia, sehingga pengguna dapat bergerak dengan lebih aman dan efisien.

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.

Dengan latar belakang ini, proyek ini bertujuan untuk meningkatkan navigasi kursi roda otonom melalui integrasi YOLOV8 dalam deteksi dan penghindaran rintangan, serta memastikan keselamatan dan kenyamanan pengguna dalam berbagai situasi.

1.2 Permasalahan

Berdasarkan latar belakang yang telah dipaparan, 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. Obstacle yang akan dijadikan subjek penghindaran adalah Manusia.
2. Penelitian ini terbatas pada pengembangan system deteksi dan kendali untuk kursi roda berbasis computer vision dengan focus pada mendeteksi keberadaan manusia di sekitarnya.
3. System yang dikembangkan akan menggunakan deteksi objek YOLOv8 (You Only Look Once) untuk mendeteksi manusia dalam gambar.
4. Pengontrol kursi roda akan diimplementasikan untuk memberikan respon berbelok secara otomatis Ketika mendeteksi keberadaan manusia yang diam di dekatnya.
5. Penelitian ini tidak mencakup pengembangan perangkat keras kendali kursi roda, melainkan hanya focus pada pengembangan algoritma perangkat lunak untuk pengendalian otomatis.
6. Mediapipe tidak digunakan untuk *training* namun, digunakan untuk pengambilan landmark pada tubuh manusia.

1.5 Manfaat

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

BAB II
TINJAUAN PUSTAKA

2.1 Hasil penelitian terdahulu

Pada subbab berikut akan dijabarkan penelitian terdahulu.

2.1.1 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. [4]

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, lokalisisasi, dan pelacakan objek di dalam ruangan untuk kursi roda, khususnya pintu dan pegangan pintu.

Tujuan utama dari penelitian ini adalah meningkatkan kemampuan kursi roda otonom, 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. [4]

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. [5]

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 pendeteksian tersebut kemudian di-

tampilkan 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 pendeteksian objek di sekitar kursi roda. Dengan demikian, sistem ini memberikan kontribusi dalam mendukung sistem keamanan kursi roda elektrik, khususnya dalam menghadapi situasi saat 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.[6]

Penelitian ini bertujuan untuk Mengembangankan 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 mendeteksi 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.[6]

2.2 Object Detection

Deteksi objek secara real-time telah menjadi komponen penting dalam berbagai aplikasi, termasuk kendaraan otonom, robotika, dan video monitoring. Metode deteksi objek telah berkembang pesat dengan penggunaan jaringan saraf konvolusional (CNN) dan algoritme khusus seperti YOLO (*You Only Look Once*) dan SSD (Single Shot Multibox Detector).

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, membuat pencarian selektif menghasilkan usulan wilayah, dan jaringan saraf konvolusional 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 terpisah. Selain itu, metode seperti Mask R-NN telah memperluas kemampuan ke segmentasi instans, memungkinkan lokalizasi 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.[7]

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah jenis *deep learning* yang dirancang khusus untuk memproses data dalam bentuk grid seperti gambar. YOLO (You Only Look Once) adalah jenis arsitektur CNN yang digunakan untuk deteksi objek dalam gambar secara real-time. Dimana dalam penggunaannya CNN memiliki beberapa lapisan (*layers*) yang secara hierarkis mengekstrak fitur dari data masukan. Berikut adalah penjelasan mengenai komponen utama dari CNN:

2.3.1 Lapisan Konvolusi (*Convolutional Layer*)

Lapisan konvolusi adalah inti dari CNN. Pada lapisan ini, filter (*kernels*) diterapkan pada input untuk menghasilkan peta fitur (*feature map*). Filter ini bergerak melintasi input dengan operasi konvolusi, menangkap pola lokal seperti tepi, tekstur, dan bentuk. Setiap filter mempelajari fitur yang berbeda dari data. berikut merupakan rumus untuk operasi konvolusi dalam konteks matematika dan pemrosesan sinyal.

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

2.3.2 Lapisan Aktivasi (*Activation Layer*)

Lapisan aktivasi memperkenalkan non-linearitas ke dalam model, memungkinkan jaringan untuk mempelajari hubungan yang kompleks. Fungsi aktivasi yang umum digunakan adalah ReLU (*Rectified Linear Unit*), yang mendefinisikan $f(x) = \max(0, x)$. Fungsi ini menggantikan nilai negatif dalam peta fitur dengan nol.

2.3.3 Lapisan Pooling (*Pooling Layer*)

Lapisan pooling mengurangi dimensi spasial dari peta fitur, mengurangi jumlah parameter dan komputasi dalam jaringan. Tipe pooling yang sering digunakan adalah *max pooling*, yang memilih nilai maksimum dalam setiap jendela pooling. Pooling membantu dalam membuat deteksi fitur lebih robust terhadap perubahan posisi dan skala.

2.3.4 Lapisan Fully Connected (*Fully Connected Layer*)

Pada lapisan fully connected, setiap neuron terhubung dengan semua neuron di lapisan sebelumnya. Lapisan ini biasanya terletak di akhir jaringan dan digunakan untuk menggabungkan fitur-fitur yang telah diekstraksi menjadi prediksi akhir.

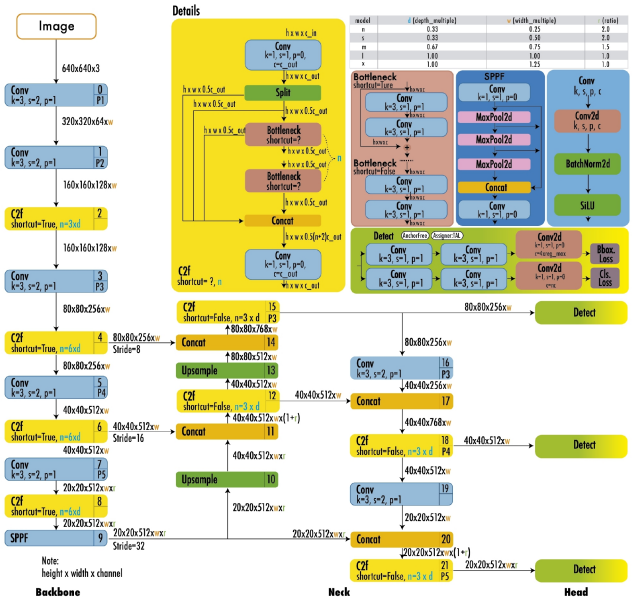
2.4 YOLO (*You Only Look Once*)

YOLO oleh [8]. Joseph Redmoon 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 pen-

dekatan sebelumnya yang menggunakan *sliding window* diikuti oleh pengklasifikasi yang harus dijalankan ratusan atau ribuan kali per gambar atau metode yang lebih canggih yang membagi tugas menjadi dua langkah. Pada metode tersebut, langkah pertama adalah mendeteksi daerah yang kemungkinan mengandung objek (*region proposals*), dan langkah kedua adalah menjalankan pengklasifikasi pada daerah-daerah yang terdeteksi tersebut. Selain itu, YOLO menggunakan *output* yang lebih sederhana berdasarkan hanya regresi untuk memprediksi *output* deteksi sebagai lawan dari Fast R-CNN yang menggunakan dua *output* terpisah, sebuah klasifikasi probabilitas dan regresi untuk sebuah *box* koordinat[8]

2.4.1 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.[7]



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.

2.5 Pose Estimation

Estimasi pose adalah metode yang menggunakan model pembelajaran mesin (*ML*) untuk memperkirakan pose manusia 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 peneliti dapat menentukan, misalnya, di mana siku manusia muncul dalam gambar. Penting untuk menyadari bahwa estimasi pose hanya memperkirakan tempat sendi tubuh kunci berada dan tidak mengenali siapa yang ada dalam gambar atau video.[9]

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.[9]

2.6 MediaPipe

MediaPipe adalah sebuah kerangka kerja yang dirancang oleh Google untuk membangun pipeline persepsi secara real-time. MediaPipe memungkinkan pengembang untuk mengintegrasikan berbagai jenis data sensor seperti video, audio, dan data lainnya dalam satu platform yang efisien dan dapat dijalankan di berbagai perangkat, mulai dari mobile hingga desktop [10]

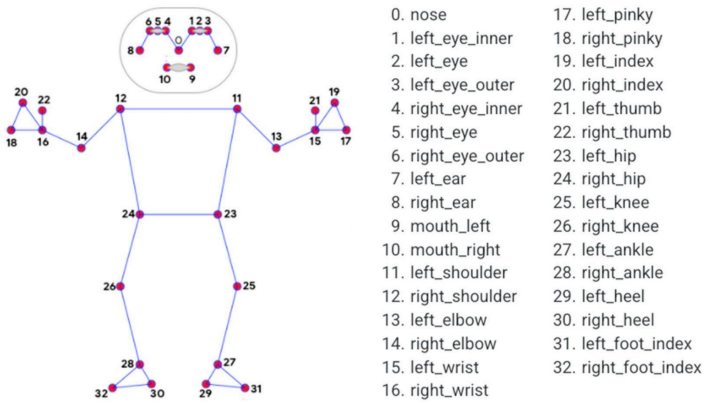
Kerangka kerja ini menggunakan konsep ”graph” sehingga setiap node di dalam graph merupakan sebuah ”calculator” yang melakukan tugas spesifik, seperti deteksi objek, pelacakan pose, atau segmentasi gambar. Setiap node ini dapat dikonfigurasi melalui GraphConfig, yang mendeskripsikan topologi serta fungsionalitas dari node-node tersebut.

Salah satu penerapan MediaPipe yang paling dikenal adalah pada estimasi pose manusia menggunakan MediaPipe Pose. Metode ini menggabungkan estimasi pose 2D dengan model humanoid yang lebih kompleks serta metode optimasi untuk mengestimasi sudut sendi pada pose 3D. Metode ini terbukti efektif dalam mengatasi masalah ambiguitas kedalaman pada es-

timasi pose 3D dan dapat berjalan secara real-time bahkan pada perangkat tanpa GPU. [10]

2.6.1 MediaPipe Pose

MediaPipe Pose (MPP), sebuah *framework open source* yang disediakan oleh Google, digunakan untuk mendapatkan perkiraan koordinat sendi manusia 2D dalam setiap bingkai gambar. MediaPipe Pose membangun garis - garis 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. [11]

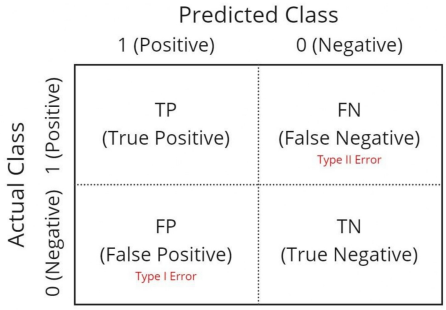


Gambar 2.2: Pose MediaPipe

2.7 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.



Gambar 2.3: Confusion Matrix

2.8 Evaluation Metrics

Penggunaan metrik evaluasi dalam konteks Deteksi Objek adalah hal yang sangat penting, karena ini berperan sebagai dasar pemahaman 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 mendeteksi 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 19 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, nilai 1 menunjukkan bahwa semua prediksi positif model adalah benar, sementara nilai 0 menunjukkan bahwa tidak ada prediksi positif yang benar.

$$\frac{TP}{TP + FP} \tag{2.2}$$

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 menunjukan bahwa kelas yang dikenali dengan benar banyak, atau false negative yang didapatkan sedikit. Rumus dari recall dapat dilihat pada persamaan dibawah

$$\frac{TP}{TP + FN} \tag{2.3}$$

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.

$$AP = \sum((Recall_{n+1} - Recall_n) \times Precision_{interp} \times Recall_{n+1}) \tag{2.4}$$

$$mAP = \frac{1}{n} \sum_{n}^{i=1} AP_i \tag{2.5}$$

2.9 Intersection over Union (IoU)

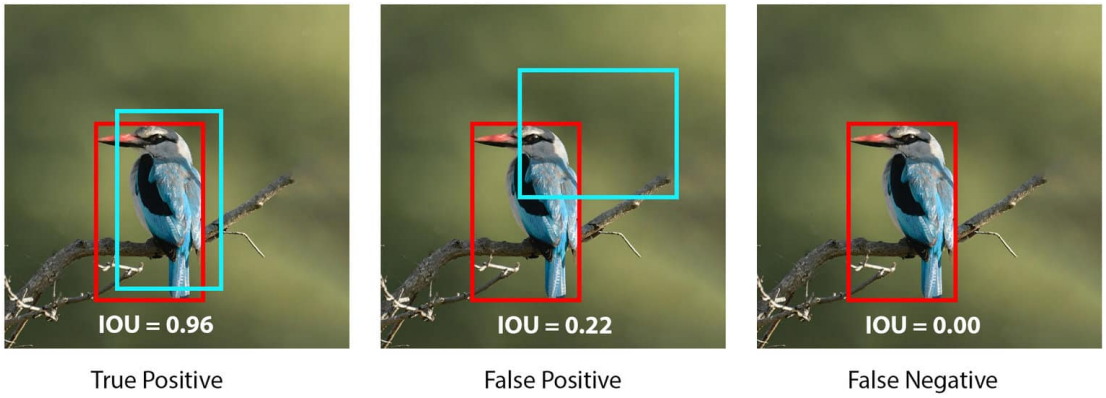
Intersection over Union, atau IoU, adalah metrik yang digunakan untuk mengevaluasi keakuratan posisi objek yang dideteksi oleh model dalam pemrosesan gambar. Prinsipnya adalah dengan menghitung area persinggungan antara kotak deteksi yang dihasilkan oleh model dengan kotak referensi yang merupakan standar 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 :

$$Intersectionover Union(IoU) = \frac{|A \bigcap B|}{|A \bigcup B|}. \tag{2.6}$$

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, 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.

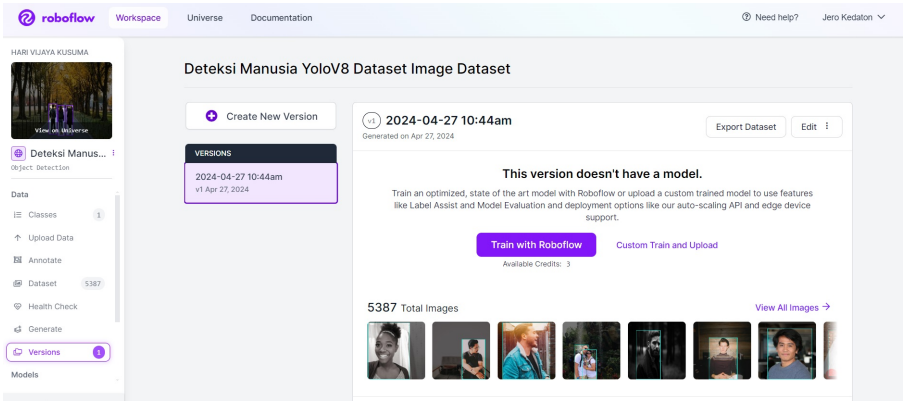
Penggunaan nilai kebenaran dasar dalam IoU merupakan perbandingan standar antara prediksi dan kondisi aktual objek yang diidentifikasi. Penandaan kotak kebenaran dasar, dilakukan secara manual oleh pakar, menentukan batas pasti objek dalam gambar. Skor IoU yang dihasilkan dari perbandingan antara prediksi 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 diandalkan dan akurat dalam berbagai kondisi dan skenario pengujian. [12]

2.10 Roboflow

RoboFlow adalah platform yang mendukung pengembangan dan penyebaran aplikasi visi komputer dengan menyediakan fitur - fitur 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.

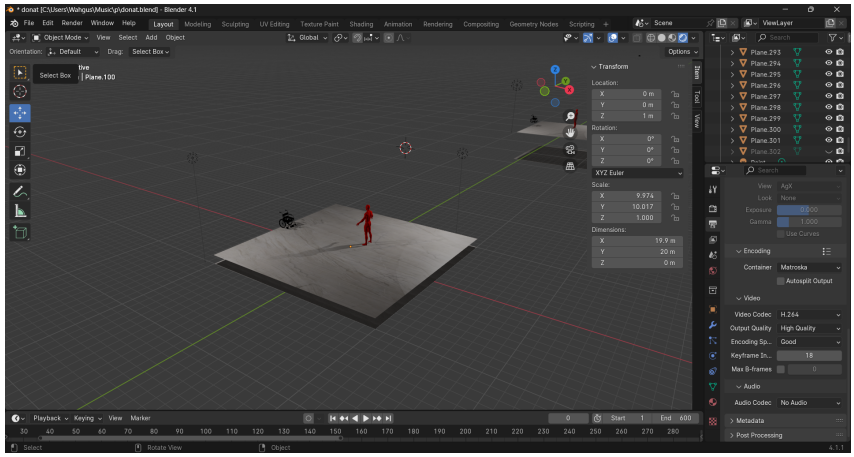
RoboFlow menawarkan integrasi yang mulus dengan banyak kerangka kerja pembelajaran mesin populer seperti TensorFlow, PyTorch, dan YOLO. Integrasi ini memungkinkan pengembang:

- 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 Blender Software

Blender adalah perangkat lunak *open-source* yang sangat populer untuk grafik 3D dan pemodelan yang digunakan secara luas dalam berbagai bidang, termasuk animasi, visualisasi ilmiah, pengembangan game, dan perencanaan bedah virtual. Blender menyediakan berbagai fitur utama, termasuk pemodelan, pemahatan, penataan UV, dan animasi. Fitur pemodelan dalam Blender mencakup dukungan penuh untuk N-Gon, penggeser tepi, dan berbagai alat pengisian grid dan jembatan. Blender juga mendukung pemahatan tingkat lanjut dengan alat-alat dan kuas khusus, subdivisi dinamis, dan multi-resolusi [13], [14]. Gambar 2.6 merupakan interface dari Blender Software.



Gambar 2.6: Interface Blender Software

Blender telah digunakan secara ekstensif untuk membuat ilustrasi ilmiah yang menarik, termasuk gambar untuk sampul jurnal dan grafik untuk makalah penelitian. Workshop yang diadakan oleh SMC media Centre, IISER Pune, telah membantu banyak ilmuwan dan mahasiswa Ph.D. dalam menguasai Blender untuk tujuan ini. Workshop ini memberikan tutorial langkah demi langkah dan dukungan chat yang sangat aktif untuk membantu peserta menyelesaikan masalah mereka [13].

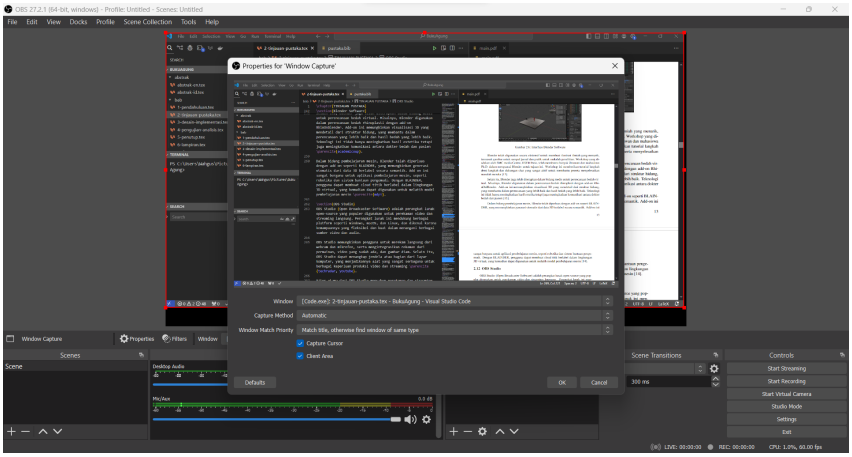
Selain itu, Blender juga telah diterapkan dalam bidang medis untuk perencanaan bedah virtual. Misalnya, Blender digunakan dalam perencanaan bedah rhinoplasti dengan add-on RhinOnBlender. Add-on ini memungkinkan visualisasi 3D yang mendetail dari struktur hidung, yang membantu dalam perencanaan yang lebih baik dan hasil bedah yang lebih baik. Teknologi ini tidak hanya meningkatkan hasil estetika tetapi juga meningkatkan komunikasi antara dokter bedah dan pasien [15].

Dalam bidang pembelajaran mesin, Blender telah diperluas dengan add-on seperti BLAINDER, yang memungkinkan generasi otomatis dari data 3D berlabel secara semantik. Add-on ini

sangat berguna untuk aplikasi pembelajaran mesin, seperti robotika dan sistem bantuan pengemudi. Dengan BLAINDER, pengguna dapat membuat cloud titik berlabel dalam lingkungan 3D virtual, yang kemudian dapat digunakan untuk melatih model pembelajaran mesin [14].

2.12 OBS Studio

OBS Studio (Open Broadcaster Software) adalah perangkat lunak open-source yang populer digunakan untuk perekaman video dan streaming langsung. Perangkat lunak ini mendukung berbagai platform seperti Windows, macOS, dan Linux, dan dikenal karena kemampuannya yang fleksibel dan kuat dalam menangani berbagai sumber video dan audio.



Gambar 2.7: Interface OBS Studio

OBS Studio memungkinkan pengguna untuk merekam langsung dari webcam dan mikrofon, serta mengintegrasikan rekaman dari permainan, video yang sudah ada, dan gambar diam. Selain itu, OBS Studio dapat menangkap jendela atau bagian dari layar komputer, yang menjadikannya alat yang sangat serbaguna untuk berbagai keperluan produksi video dan streaming [16], [17].

Fitur utama dari OBS Studio mencakup perekaman dan streaming langsung dengan kualitas tinggi menggunakan encoding H264/AAC. Pengguna dapat bekerja dengan berbagai sumber, mencampurnya, dan menciptakan satu siaran yang mulus. OBS Studio mendukung banyak layanan streaming utama seperti Twitch, YouTube, Facebook Live, dan lainnya [16], [18].

OBS Studio digunakan tidak hanya oleh gamer dan streamer, tetapi juga oleh pendidik dan profesional lainnya untuk berbagai keperluan produksi video. Dengan dukungan untuk berbagai sumber dan kemampuan untuk mencampur konten secara real-time, OBS Studio menjadi pilihan utama bagi banyak pengguna yang membutuhkan solusi perekaman dan streaming yang fleksibel dan kuat [17].

2.13 Intel NUC

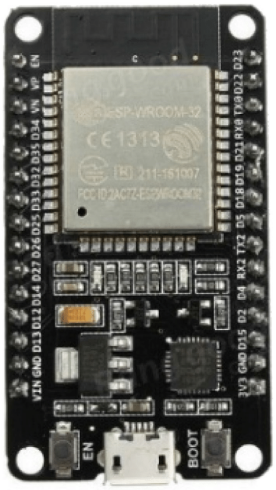
Intel NUC (Next Unit of Computing) adalah sebuah mini PC *Powerfull* yang dirancang oleh Intel untuk memenuhi berbagai kebutuhan komputasi, mulai dari penggunaan kasual hingga gaming kompetitif dan tugas profesional. Intel NUC dengan fitur prosesor Intel Core generasi dalam form factor kompak 4x4 inci. Dirancang untuk menawarkan kombinasi ukuran, kinerja,

keberlanjutan, dan keandalan yang dibutuhkan oleh bisnis modern. Intel NCU 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 singage.

Gambar 2.8: Gambar Intel NUC

2.14 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 2.9: Gambar ESP32 Devkit V1

2.15 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. [19]

Gambar 2.10: Gambar Motor Driver H-Bridge

2.16 Kursi Roda Elektrik KY-123

Kursi roda Elektrik KY-123 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.

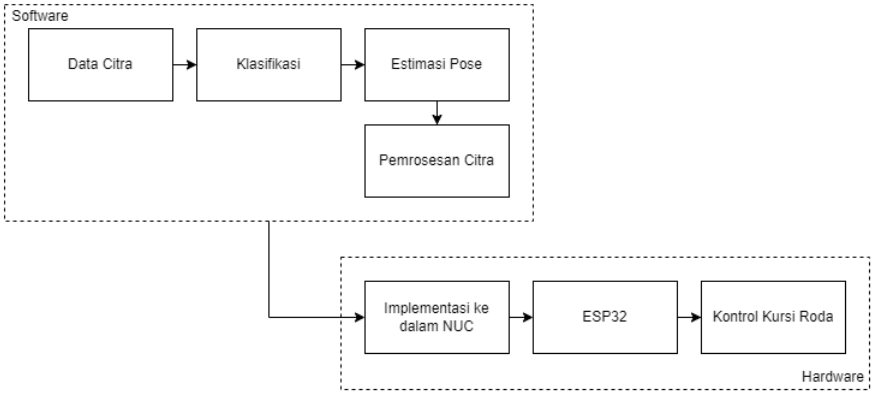
Gambar 2.11: Gambar Kursi Roda Elektrik KY-123

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

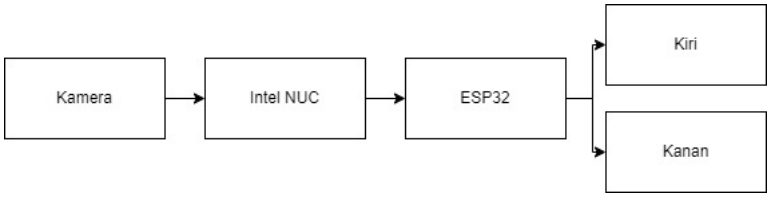
Penelitian dan pembuatan sistem ini diterapkan sesuai dengan desain dan implementasi pada bab ini. Desain sistem ini mencakup konsep pembuatan, perancangan, alur, dan implementasi infrastruktur yang dibuat dalam Blok Diagram. Desain dan penerapan diilustrasikan melalui penggunaan Gambar dan akan dijelaskan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



Gambar 3.1: Blok Diagram Sistem

3.2 Hardware

Perancangan hardware dilakukan sesuai dengan alur yang akan dideskripsikan pada subbab ini. Perangangan ini akan dipresentasikan dengan blok diagram alur yang telah merepresentasikan alur perancangan hardware ini. Gambar 3.2 merupakan blok diagram hardware yang akan ditampilkan sebagai berikut



Gambar 3.2: Blok Diagram Hardware

3.2.1 Intel NUC

Pada tahap awal, dilakukan proses akuisisi data dengan menggunakan kamera yang telah dipasangkan pada bracket khusus. Bracket ini dirancang sedemikian rupa sehingga memungkinkan kamera berada sejajar dengan manusia yang akan dideteksi. Hal ini bertujuan untuk memastikan bahwa sudut pandang kamera optimal untuk menangkap citra manusia dengan akurasi tinggi. Proses ini melibatkan pengambilan gambar dan video secara real-time, yang merupakan langkah krusial untuk memastikan bahwa data yang dihasilkan relevan dan akurat untuk analisis lebih lanjut.

Setelah data citra manusia berhasil diperoleh, langkah berikutnya adalah melakukan serangkaian pengolahan data citra tersebut menggunakan model deteksi yang telah dilatih sebelumnya. Model ini dilatih menggunakan YOLOV8. Model yang digunakan diberi nama Best.pt, yang merupakan file hasil pelatihan yang telah di *training* untuk mendeteksi kelas manusia. Selain menggunakan YOLO, proses ini juga melibatkan penggunaan MediaPipe, sebuah *framework* yang dirancang untuk menangkap dan memproses *landmark* tubuh manusia. Dengan menggabungkan kedua teknik ini, sistem dapat memperoleh informasi detail mengenai posisi dan pose tubuh manusia yang terdeteksi.

Untuk mendukung proses ini, beberapa *library* atau pustaka penting perlu diinstal terlebih dahulu. *Library* tersebut mencakup OpenCV untuk pengolahan citra, Ultralytics untuk Yolo, dan MediaPipe untuk menangkap landmark tubuh manusia listnya sebagai berikut :

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

library tersebut memastikan bahwa semua alat dan dependensi yang diperlukan untuk pengolahan citra dan deteksi objek sudah tersedia pada sistem.

Proses pengolahan data citra ini seluruhnya dilakukan pada perangkat Intel NUC, sebuah komputer mini yang memiliki kapasitas komputasi yang cukup untuk menjalankan model deteksi dan pengolahan citra secara real-time. Setelah data citra manusia diproses dan diklasifikasikan, hasil klasifikasi ini kemudian dikirimkan dari Intel NUC ke modul ESP32 melalui koneksi WiFi. Proses pengiriman data ini sangat penting karena hasil klasifikasi akan digunakan untuk mengendalikan motor pada kursi roda, memungkinkan kursi roda untuk menghindari hambatan secara otomatis.

Untuk memastikan bahwa proses pengiriman data ini berjalan lancar, Intel NUC harus terlebih dahulu terhubung dengan access point yang disediakan oleh ESP32. Proses ini melibatkan konfigurasi koneksi WiFi pada Intel NUC agar dapat berkomunikasi dengan ESP32.

Setelah koneksi berhasil terjalin, data hasil klasifikasi mulai dikirimkan. Pengiriman data ini menggunakan bahasa pemrograman Python, yang dipilih karena fleksibilitas dan kemampuannya dalam menangani berbagai macam operasi jaringan dan pengolahan data. Data dikirim dalam bentuk string, format yang sederhana namun efektif untuk transmisi data melalui jaringan.

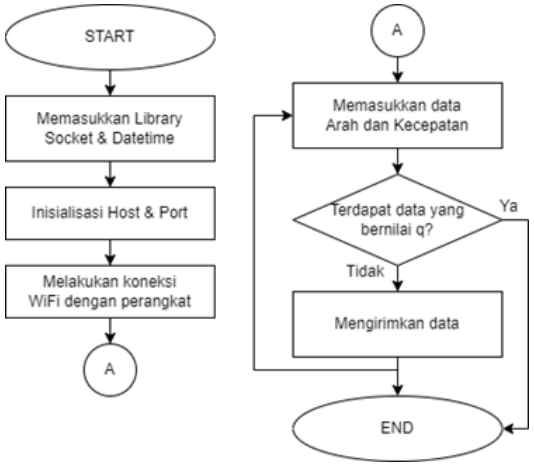
Untuk membangun koneksi jaringan yang memungkinkan pengiriman dan penerimaan data, digunakan beberapa library Python, yaitu socket, time, dan datetime. Library socket digunakan untuk menangani komunikasi jaringan, memungkinkan program untuk mengirim dan menerima data melalui koneksi TCP/IP. Library time digunakan untuk mengatur interval waktu tertentu selama proses pengiriman data, memastikan bahwa data dikirim pada waktu yang tepat. Sementara itu, library datetime digunakan untuk mencatat waktu pengiriman data, yang berguna

untuk logging dan debugging.

IP Address dari access point ESP32 dijadikan sebagai variabel host dalam program, sedangkan port 80 digunakan sebagai variabel port. Port 80 adalah port standar untuk komunikasi HTTP, yang memungkinkan program untuk menggunakan protokol HTTP untuk mengirim data.

Setelah semua konfigurasi selesai, program kemudian melakukan koneksi ke server yang telah ditentukan sesuai dengan IP Address dan port yang telah diinputkan. Koneksi ini memastikan bahwa data dapat dikirimkan dari Intel NUC ke ESP32 dengan lancar.

Program ini dirancang untuk berjalan secara berulang-ulang tanpa batas, terus-menerus menerima input dari pengguna dan mengirimkan data ke ESP32. Proses ini digambarkan dalam flowchart pada Gambar 3.3, yang menjelaskan langkah-langkah detail dari awal hingga akhir. [20]



Gambar 3.3: Flowchart mengirim data string melalui Wifi

Berikut adalah langkah-langkah detail berdasarkan flowchart. Program dimulai dengan mengimpor library yang diperlukan, yaitu socket dan datetime. Langkah pertama ini penting untuk memastikan bahwa semua alat dan fungsi yang dibutuhkan untuk komunikasi jaringan dan pengolahan waktu tersedia. Setelah itu, dilakukan inisialisasi variabel host dan port dengan menetapkan IP Address ESP32 sebagai host dan port 80 sebagai port. Inisialisasi ini penting untuk menentukan alamat jaringan dan port komunikasi yang akan digunakan oleh program.

Setelah inisialisasi, program kemudian melakukan koneksi WiFi dengan perangkat ESP32. Langkah ini memastikan bahwa koneksi jaringan antara Intel NUC dan ESP32 terjalin dengan baik, memungkinkan transfer data yang lancar. Setelah terhubung, program mulai memasukkan data arah dan kecepatan yang akan dikirimkan dari Intel NUC ke ESP32. Data ini penting untuk mengendalikan motor kursi roda berdasarkan hasil klasifikasi deteksi manusia yang telah dilakukan sebelumnya.

Selanjutnya, program memeriksa apakah ada data yang memiliki nilai ’q’ yang menunjukkan perintah tertentu. Jika ada, program akan menangani perintah tersebut sesuai dengan logika yang telah ditentukan. Ini memungkinkan program untuk mengeksekusi perintah khusus yang mungkin diberikan oleh pengguna. Jika tidak ada data dengan nilai ’q’, maka data akan dikirimkan ke ESP32 untuk digunakan dalam pengendalian motor kursi roda. Langkah ini

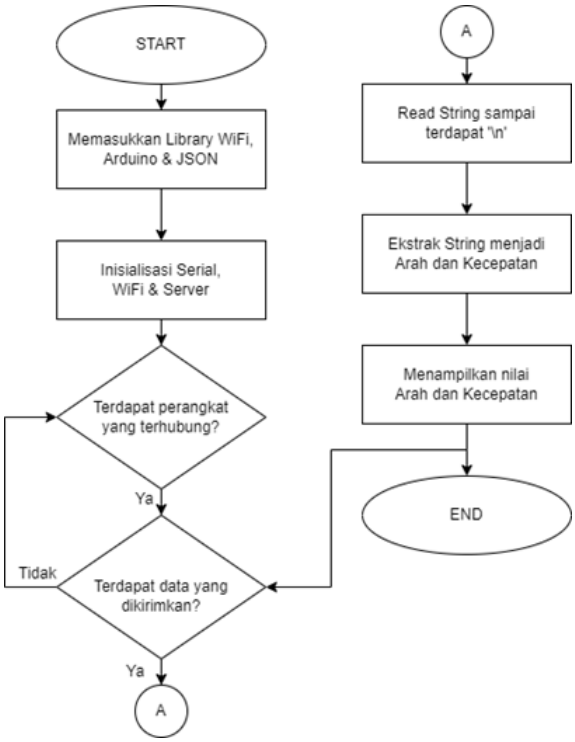
memastikan bahwa data yang dikirimkan selalu diperbarui dan digunakan untuk mengendalikan kursi roda secara real-time.

Proses ini berulang tanpa batas, terus-menerus menerima input dari pengguna, memproses data, dan mengirimkannya ke ESP32. Siklus berkelanjutan ini memastikan bahwa sistem selalu responsif terhadap perubahan lingkungan dan perintah pengguna, memungkinkan kursi roda untuk bergerak secara otonom dan aman. Dengan demikian, program ini dirancang untuk beroperasi secara efisien dan efektif dalam mengendalikan motor kursi roda berdasarkan data klasifikasi deteksi manusia yang diterima.

Dengan demikian, program ini memastikan bahwa data hasil klasifikasi dapat diterima oleh ESP32 dan digunakan untuk mengendalikan kursi roda secara efektif. Setiap langkah dalam proses ini dirancang untuk memastikan bahwa sistem dapat beroperasi secara real-time, responsif terhadap perubahan lingkungan, dan dapat diandalkan untuk tugas-tugas navigasi otonom.

3.2.2 ESP32

Selanjutnya, ESP32 akan menerima data arah hasil klasifikasi deteksi manusia dalam bentuk karakter huruf seperti A, B, C, D, atau E. Berdasarkan penelitian yang telah dilakukan sebelumnya, proses ESP32 dalam menerima data string dari NUC telah divisualisasikan pada flowchart yang ditampilkan pada Gambar 3.4. Program ini dirancang untuk berfungsi sebagai server WiFi yang bertugas menerima data dari perangkat yang terhubung, yang dalam pengujian kali ini adalah NUC. [20]



Gambar 3.4: Flowchart menerima data string melalui Wifi

Program dimulai dengan mengimpor beberapa library yang diperlukan untuk fungsi-fungsi yang akan dijalankan. Library yang diimpor termasuk WiFi, Arduino, dan JSON. Library WiFi digunakan untuk menangani koneksi jaringan, library Arduino untuk kontrol perangkat keras,

dan library JSON untuk pengolahan data dalam format JSON. Setelah itu, dilakukan inisialisasi untuk serial komunikasi, WiFi, dan server. Inisialisasi ini penting untuk memastikan bahwa ESP32 siap untuk menerima dan mengirim data melalui jaringan.

Selanjutnya, program memeriksa apakah ada perangkat yang terhubung dengan server ESP32. Ini dilakukan dengan menggunakan fungsi yang memeriksa koneksi jaringan. Jika tidak ada perangkat yang terhubung, program akan terus memeriksa hingga ada perangkat yang berhasil terhubung. Setelah perangkat terhubung, langkah berikutnya adalah memeriksa apakah ada data yang dikirimkan oleh perangkat tersebut. Jika data ditemukan, program akan memasuki loop untuk membaca data yang diterima secara terus-menerus. Data yang diterima akan disimpan dalam variabel *received Data*. Data string yang diterima akan dibaca sampai ditemukan karakter newline (/n), yang menandakan akhir dari string data tersebut.

Setelah data diterima dan disimpan, string data tersebut akan diekstraksi menjadi informasi arah dan kecepatan. Proses ekstraksi ini penting untuk mengubah string data menjadi format yang dapat digunakan oleh program untuk mengendalikan motor kursi roda. Data yang sudah diekstraksi akan ditampilkan untuk memastikan bahwa arah dan kecepatan yang diterima sesuai dengan yang diharapkan. Dengan data arah dan kecepatan yang sudah tersedia, ESP32 dapat mengirimkan instruksi ke motor kursi roda untuk bergerak sesuai dengan data yang diterima. Proses ini akan terus berulang selama perangkat terhubung dan data terus diterima.

Untuk memberikan pemahaman yang lebih jelas mengenai instruksi yang diterima dari hasil klasifikasi deteksi manusia, berikut pada tabel 3.1 merupakan kode instruksi dari hasil klasifikasi yang digunakan dalam program ini:

Tabel 3.1: Kode instruksi dari hasil klasifikasi

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

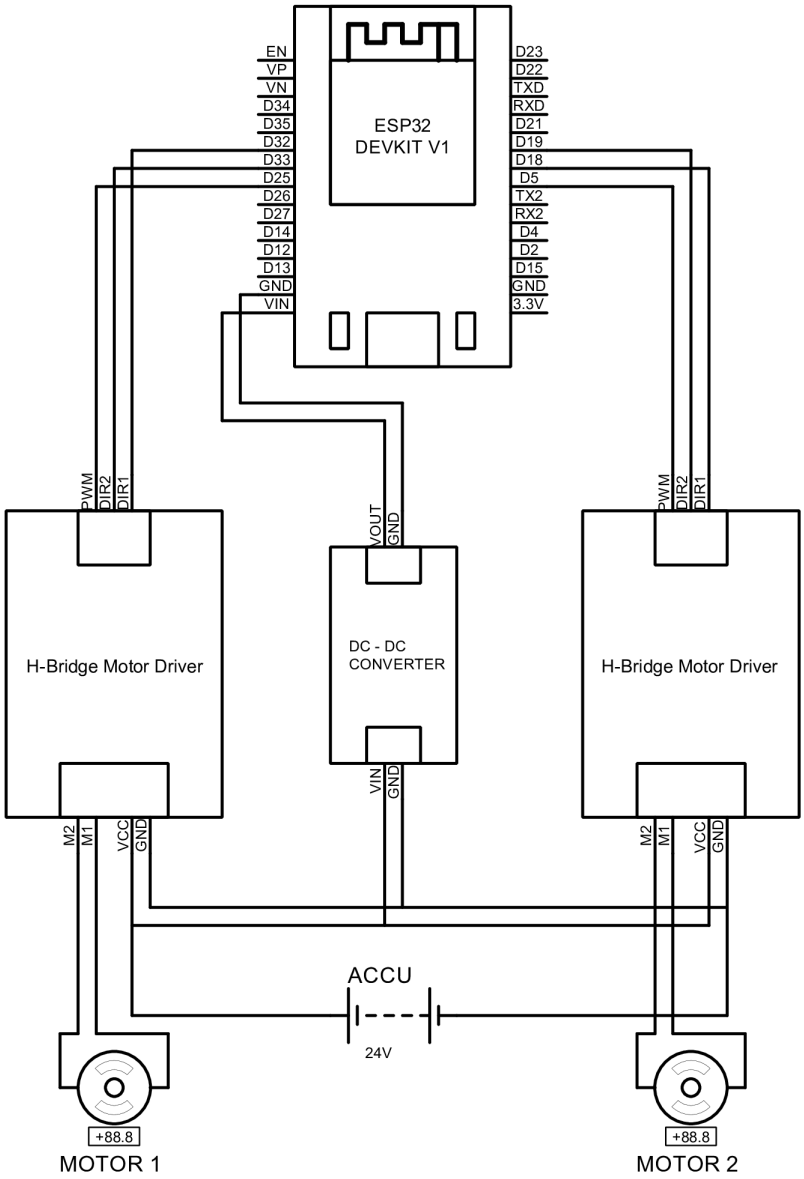
Kode instruksi tersebut digunakan untuk mengarahkan motor kursi roda sesuai dengan deteksi yang dilakukan oleh model klasifikasi YOLOv8. Misalnya, jika arah yang terdeteksi adalah 'Kiri', maka kode instruksi 'A' akan dikirimkan untuk menggerakkan kursi roda ke kiri. Begitu pula dengan arah 'Maju' yang akan mengirimkan kode instruksi 'B' untuk menggerakkan kursi roda maju, 'Stop' dengan kode 'C' untuk menghentikan kursi roda, 'Mundur' dengan kode 'D' untuk bergerak mundur, dan 'Kanan' dengan kode 'E' untuk bergerak ke kanan.

Program ini memastikan bahwa ESP32 dapat berfungsi secara efektif sebagai server yang menerima data dari NUC dan menggunakannya untuk mengendalikan motor kursi roda. Dengan langkah-langkah yang telah dijelaskan, program ini dirancang untuk berjalan secara kontinu tanpa batas, menunggu dan memproses data yang dikirimkan oleh perangkat yang terhubung. Flowchart pada Gambar 3.4 memberikan gambaran yang jelas mengenai alur kerja program ini, mulai dari inisialisasi, pengecekan koneksi, penerimaan data, hingga pemrosesan data dan pengendalian motor. Setiap langkah dalam flowchart ini dirancang untuk memastikan bahwa program dapat berjalan secara efisien dan responsif terhadap data yang diterima, sehingga sistem dapat beroperasi secara real-time dan memberikan respon yang cepat terhadap perubahan

yang terjadi di lingkungan sekitar kursi roda.

3.2.3 Skematik Alat

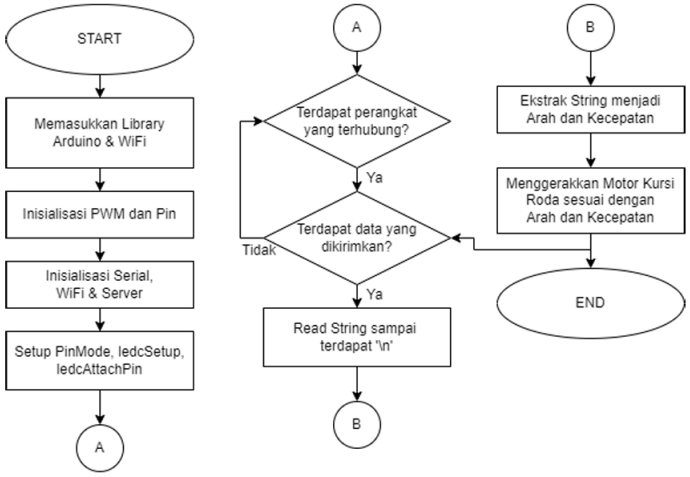
Dari penelitian yang telah dilakukan, telah dirancang sebuah sistem kontrol untuk motor kursi roda dengan skematik alat yang ditampilkan pada Gambar 3.5. ESP32 akan terhubung dengan dua buah H-Bridge Motor Driver dan sebuah DC-DC Converter. Setiap H-Bridge Motor Driver terhubung langsung ke motor roda kiri dan motor roda kanan untuk menggerakkan roda kursi roda secara efektif. Dalam skematik ini, ESP32 berfungsi sebagai otak dari sistem, mengirimkan sinyal kontrol ke motor driver berdasarkan data yang diterima melalui koneksi WiFi. [20]



Gambar 3.5: Skematik kontrol motor kursi roda

Proses kontrol motor kursi roda dijelaskan dalam flowchart pada Gambar 3.6, sesuai dengan penelitian yang telah dilakukan. Program ini dirancang untuk mengendalikan pergerakan

motor DC dengan cara memproses perintah yang diterima melalui jaringan WiFi. Langkah pertama dalam program adalah mengimpor library yang diperlukan, seperti Arduino dan WiFi, untuk memastikan semua fungsi yang dibutuhkan tersedia. Setelah itu, program melakukan inisialisasi PWM dan pin untuk mengatur sinyal kontrol yang akan dikirimkan ke motor driver.[20]



Gambar 3.6: Flowchart kontrol kursi roda Melalui Wifi

Setelah inisialisasi, program memeriksa apakah ada perangkat yang terhubung dengan server ESP32. Jika tidak ada perangkat yang terhubung, program akan terus memeriksa hingga ada perangkat yang berhasil terhubung. Setelah perangkat terhubung, langkah berikutnya adalah memeriksa apakah ada data yang dikirimkan oleh perangkat tersebut. Jika data ditemukan, program akan membaca data tersebut hingga menemukan karakter newline (/n), yang menandakan akhir dari string data.

Data yang diterima kemudian diekstraksi menjadi informasi arah dan kecepatan. Proses ekstraksi ini penting untuk mengubah string data menjadi format yang dapat digunakan oleh program untuk mengendalikan motor kursi roda. Dengan data arah dan kecepatan yang sudah tersedia, ESP32 dapat mengirimkan instruksi ke motor driver untuk menggerakkan roda kursi roda sesuai dengan arah dan kecepatan yang diinginkan.

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

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

Flowchart dan skematik yang ditampilkan memberikan gambaran yang jelas mengenai alur kerja dan koneksi perangkat keras yang digunakan dalam sistem kontrol ini. Dengan penjelasan yang mendetail, setiap aspek dari sistem ini dapat dipahami dengan baik, memastikan implementasi yang tepat dan operasi yang efisien dari kursi roda yang dikendalikan secara otonom. [20]

3.3 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 3.7 merupakan blok diagram alur akan ditampilkan sebagai berikut :

Gambar 3.7: Diagram Blok Software

3.3.1 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 pendeteksian, nantinya proses ini terjadi secara real-time guna mendapatkan keseluruhan citra yang dibutuhkan untuk mengenali manusia atau tidak pada citra secara terus-menerus. Gambar 3.8 merupakan contoh data citra.

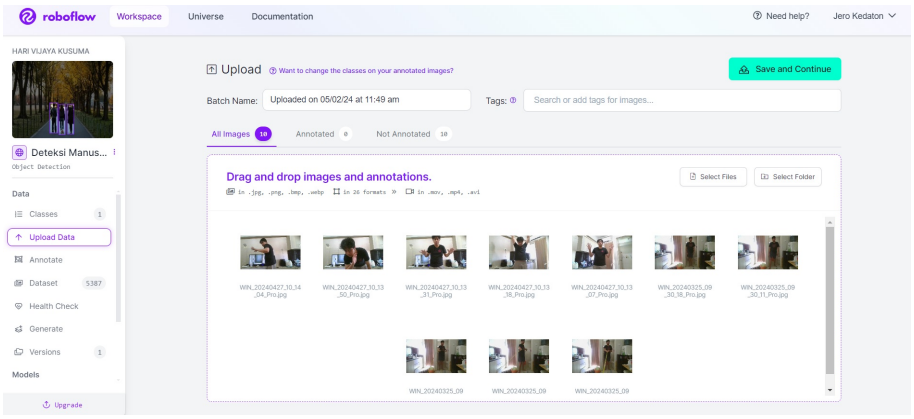
Gambar 3.8: Contoh Data Citra.

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 3.8 merupakan contoh hasil dari data citra yang didapatkan melalui perangkat kamera pada laptop pribadi. Dalam hal ini, karena

yang ingin dikenali adalah manusia, maka citra gambar citra yang didapatkan harus memiliki objek manusia didalamnya.

3.3.2 Labeling

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 3.9 merupakan contoh upload dataset ke roboflow.

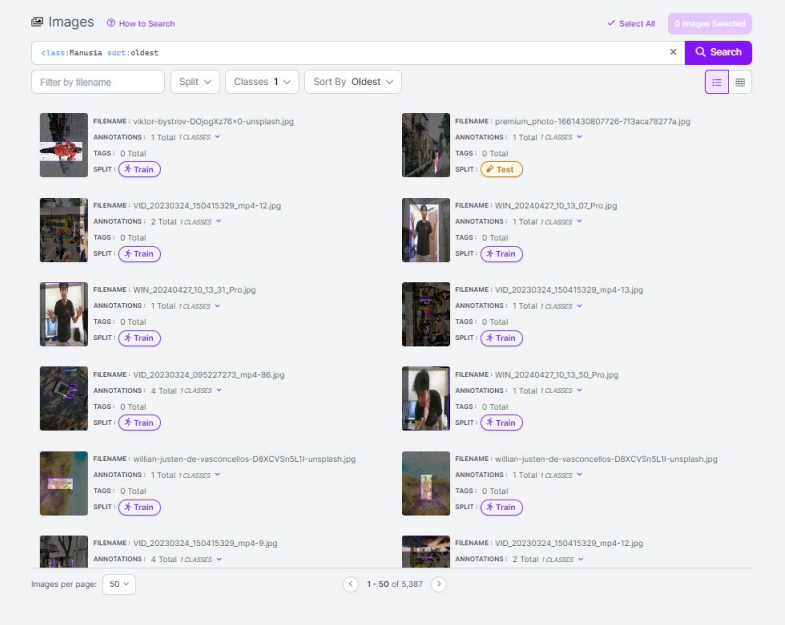


Gambar 3.9: Contoh upload dataset ke roboflow.

Dataset yang diupload haruslah mencakup beberapa skenario dan mencakup berbagai ukuran Setelah itu melakukan Praproses dataset dengan anotasi seperti pada Gambar 3.10.

Gambar 3.10: 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. Setelah selesai maka dataset akan terlihat seperti pada Gambar 3.11. Selain itu dalam proses anotasi harus diperhatikan posisi objek yang akan dianotasi harus jelas agar tidak ada kesalahan model dalam mendeteksi.



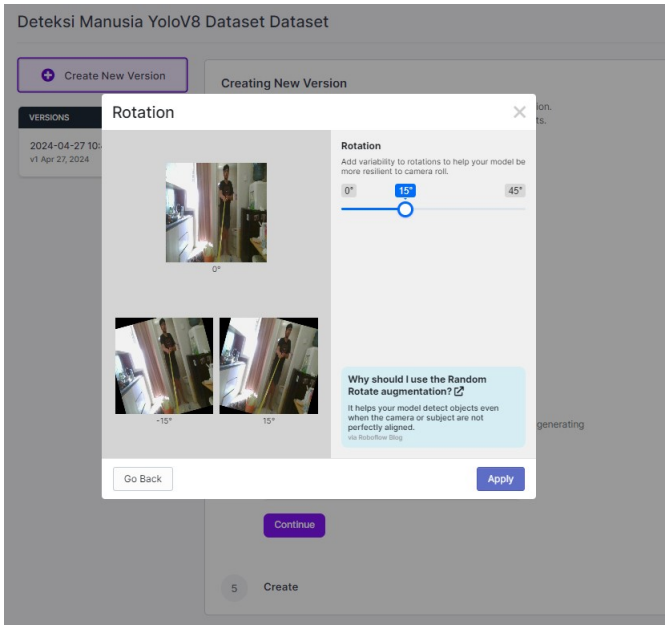
Gambar 3.11: Contoh Dataset.

Setelah seluruh Gambar yang dipilih telah dianotasi, terdapat opsi membuat datasets dan Pemilihan Gambar yang dikelompokkan dari grup batch dataset tesebut 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.

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 tesebut.

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

Adapula fitur Augmentasi data yaitu langkah ketika augmentasi diterapkan pada Gambar yang ada di kumpulan data yang dapat dilihat pada gambar 3.12. Fitur augmentasi dapat menambah keragaman dalam Dataset. Dalam tugas akhir ini perlu digunakan mengingat diperlukan banyak keragaman dalam data citra manusia, dimana dalam pengaplikasiannya dapat membantu meningkatkan performa model dalam mendeteksi Manusia.

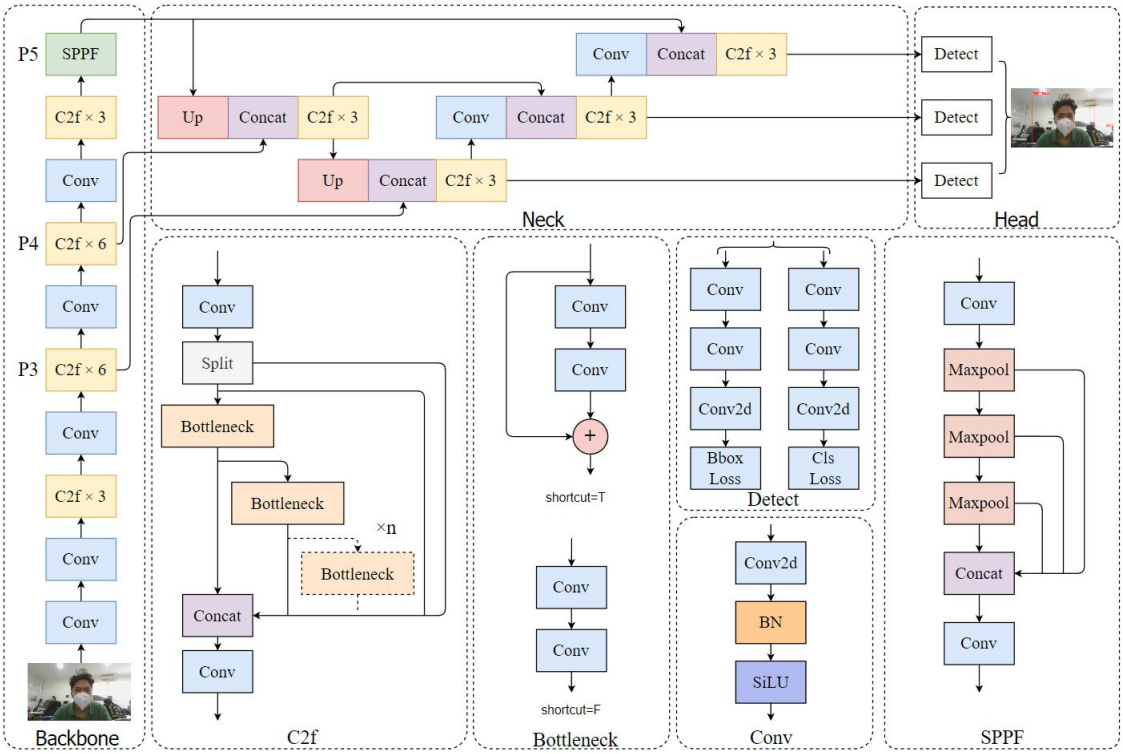


Gambar 3.12: Contoh Augmentasi Dataset.

3.3.3 Klasifikasi YoloV8

Dalam proses pengklasifikasian, setiap citra yang telah melalui proses Labeling akan dikenali dengan menggunakan YoloV8 yang telah ditraining untuk mengenali Manusia yang terdapat pada citra. Model akan mendapatkan memberikan output sesuai dengan kelas manusia, nantinya hasil dari klasifikasi akan dijadikan acuan posisi pada grid deteksi.

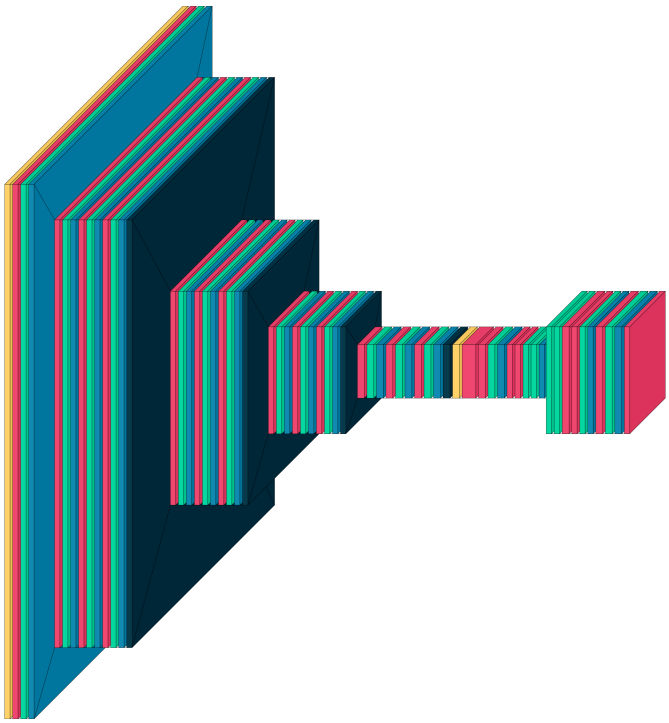
Model yang digunakan memiliki output kelas yaitu Manusia, adapun nilai yang diberikan oleh model yaitu berupa *Bounding Box* dan Nilai Konfiden (*Confidence Score*. Alur prosesnya dapat dilihat pada Gambar 3.13 yang merupakan blok diagram arsitektur.



Gambar 3.13: Visualisasi Arsitektur YoloV8

YoloV8 menggunakan Convolutional Neural Networks (CNN) sebagai dasar arsitekturnya. Dalam YOLO, jaringan CNN digunakan untuk ekstraksi fitur dari gambar input dan kemudian menerapkan jaringan lain untuk memprediksi *Bounding Box* dan kelas objek langsung dari gambar tersebut. Berdasarkan salah satu pelatihan yang telah dilakukan, Model ini memiliki 225 *Layer* atau lapisan, 3011043 parameter, dan 8.2 GFLOPs. Terdapat lapisan *Convolution 2D (Conv2D)*, Blok C2f, Blok SPPF, Lapisan Upsampling, dan Lapisan Concat. Gambar 3.14 merepresentasikan setiap jenis layer dengan warna yang berbeda.

Warna biru menunjukkan lapisan convolutional yang ada dalam backbone, neck, dan head. Warna kuning menunjukan C2f blok residual dengan convolution dan Add (shortcut connections). Warna Hijau Muda menunjukkan SPPF lapisan Spatial Pyramid Pooling Fast. Merah Muda Menunjukkan lapisan UpSampling. Warna Ungu menunjukkan lapisan Concatenate untuk menggabungkan peta fitur. Untuk detail input dan output dari visualisasi arsitektur diatas dapat dilihat pada gambar berikut

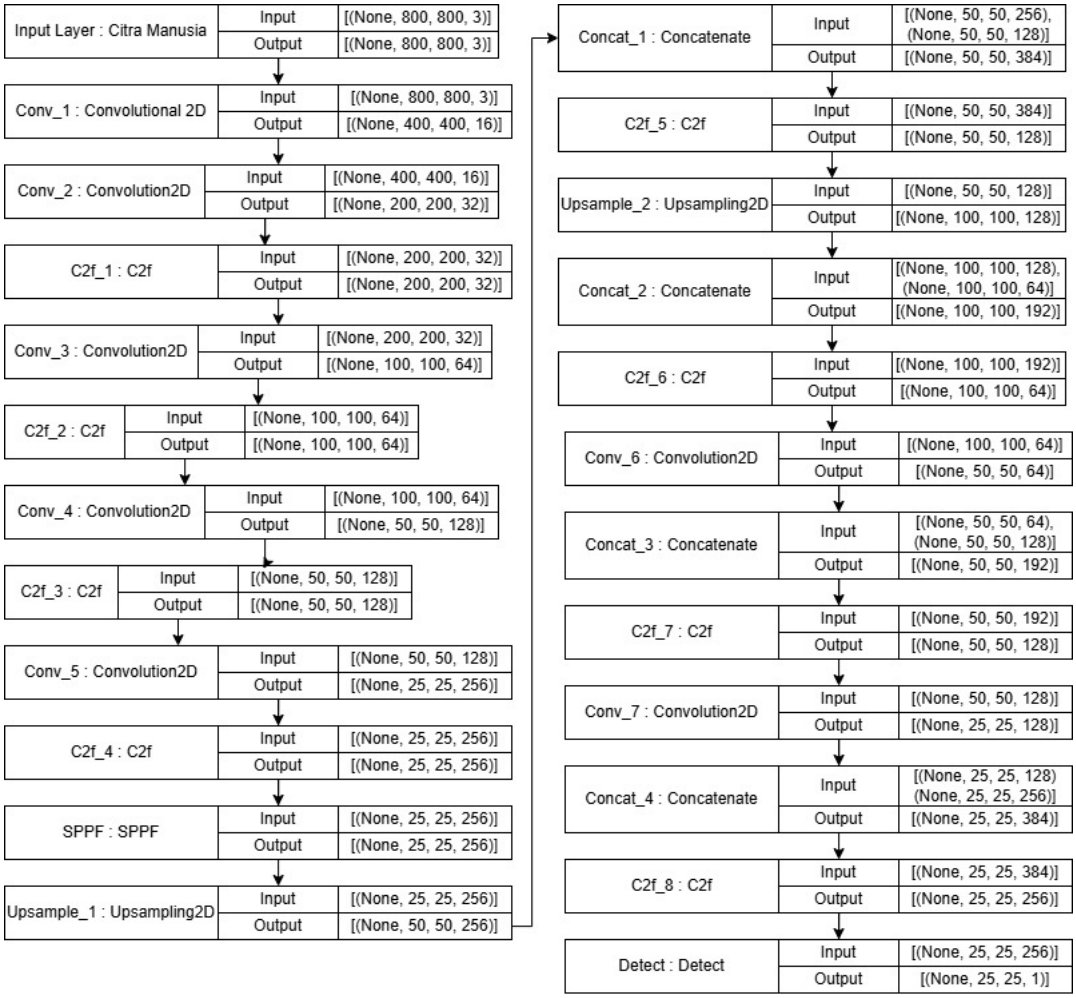


Gambar 3.14: Visualisasi Dengan VisualKeras

Model YOLOv8 menggunakan beberapa jenis lapisan berbeda untuk memproses gambar dan mendeteksi objek. Pertama, lapisan Convolution 2D (Conv2D) melakukan operasi konvolusi pada gambar input untuk mengekstraksi fitur. Operasi ini melibatkan filter pada gambar, mengalikan dan menjumlahkan nilai-nilai piksel untuk menghasilkan fitur map baru. Blok C2f, atau Cross Stage Partial Networks, menggabungkan fitur dari dua tahap yang berbeda menggunakan beberapa lapisan konvolusi dengan skip connections, yang meningkatkan pemahaman konteks spasial dan fitur kompleks.

Blok SPPF (Spatial Pyramid Pooling Fast) bertujuan untuk menangkap informasi dari berbagai skala dalam fitur map dengan menerapkan pooling pada beberapa tingkat, seperti 1x1, 3x3, dan 5x5, kemudian menggabungkannya. Ini membantu model mendeteksi objek dengan berbagai ukuran dan posisi. Lapisan Upsampling meningkatkan resolusi fitur map dengan memperbesar dimensinya menggunakan metode seperti nearest neighbor atau bilinear interpolation, memungkinkan deteksi objek dengan resolusi lebih tinggi.

Lapisan Concat (Concatenate) menggabungkan dua atau lebih fitur map dari jalur yang berbeda dalam model. Ini dilakukan dengan menggabungkan fitur map di sepanjang dimensi channel, sehingga mempertahankan informasi dari berbagai tahap pemrosesan dan memberikan konteks yang lebih kaya untuk deteksi objek. Dengan menggabungkan informasi dari berbagai lapisan, model dapat membuat representasi yang lebih kuat dan akurat, yang penting untuk tugas deteksi objek yang kompleks. Detail input output dapat dilihat pada Gambar 3.15



Gambar 3.15: Arsitektur YoloV8

Model YOLOv8 menggunakan berbagai jenis lapisan untuk memproses gambar dan mendeteksi objek, dengan setiap lapisan memiliki jumlah filter, resolusi output, dan jumlah channel yang spesifik. Adapun input dan output dilambangkan dengan nilai [(Batch, Resolusi, Resolusi, Channel)]. Batch di set None karena hanya akan di set pada proses training (nilai umum 8, 16, 32 , 64) tergantung konfigurasi. Resolusi di set 800x800 piksel. channel merupakan dimensi dari fitur yang diekstraksi dari gambar pada input awal channel bernilai 3 sesuai dengan fitur RGB pada citra. Lapisan konvolusi pertama (Conv_1) menggunakan 16 filter yang mengubah gambar berwarna dengan resolusi 800x800 piksel dan 3 channel (RGB) menjadi fitur map dengan resolusi 400x400 dan 16 channel. Lapisan konvolusi kedua (Conv_2) menggunakan 32 filter menggandakan jumlah channel menjadi 32 dan mengurangi resolusi menjadi 200x200.

Blok C2f pertama (C2f_1) mempertahankan resolusi 200x200 dengan 32 channel, menggunakan beberapa lapisan konvolusi dengan skip connections. Lapisan konvolusi ketiga (Conv_3) menggunakan 64 filter meningkatkan jumlah channel menjadi 64 dan mengurangi resolusi menjadi 100x100. Blok C2f kedua (C2f_2) mempertahankan resolusi 100x100 dengan 64 channel, kembali menggunakan beberapa lapisan konvolusi dengan skip connections.

Lapisan konvolusi keempat (Conv_4)menggunakan 128 filter mengubah resolusi menjadi 50x50 dan meningkatkan jumlah channel menjadi 128. Blok C2f ketiga (C2f_3) mempertahankan resolusi 50x50 dengan 128 channel. Lapisan konvolusi kelima (Conv_5), menggunakan

256 filter mengurangi resolusi lebih lanjut menjadi 25x25 dan meningkatkan jumlah channel menjadi 256. Blok C2f keempat (C2f_4) dan blok SPPF (Spatial Pyramid Pooling Fast) menggunakan 256 filter mempertahankan resolusi 25x25 dengan 256 channel, menangkap informasi multi-skala dalam fitur map.

Lapisan upsampling pertama (Upsample_1) meningkatkan resolusi fitur menjadi 50x50 dengan 256 channel. Lapisan concat pertama (Concat_1) menggabungkan fitur dari dua jalur berbeda, menghasilkan fitur map dengan resolusi 50x50 dan 384 channel. Blok C2f kelima (C2f_5) mengurangi jumlah channel menjadi 128 sambil mempertahankan resolusi 50x50. Lapisan upsampling kedua (Upsample_2) meningkatkan resolusi map menjadi 100x100 dengan 128 channel

Lapisan concat kedua (Concat_2) menggabungkan fitur menjadi 192 channel dengan resolusi 100x100. Blok C2f keenam (C2f_6) mempertahankan resolusi 100x100 dengan 64 channel. Lapisan konvolusi keenam (Conv_6) mengurangi resolusi menjadi 50x50 dengan 64 channel. Lapisan concat ketiga (Concat_3) menggabungkan fitur menjadi 192 channel. Blok C2f ketujuh (C2f_7) mempertahankan resolusi 50x50 dengan 128 channel.

Lapisan konvolusi ketujuh (Conv_7) mengurangi resolusi menjadi 25x25 dengan 128 channel. Lapisan concat keempat (Concat_4) menggabungkan fitur menjadi 384 channel. Blok C2f kedelapan (C2f_8) mempertahankan resolusi 25x25 dengan 256 channel. Akhirnya, lapisan deteksi (Detect) menghasilkan output dengan resolusi 25x25 dan satu channel, memberikan koordinat bounding box dan confidence score untuk satu kelas objek.

3.3.4 Estimasi Pose MediaPipe

Pada penelitian ini, deteksi pose dilakukan menggunakan Python dengan library OpenCV dan framework MediaPipe. Proses dimulai saat objek manusia terdeteksi dalam frame citra, dan kemudian MediaPipe digunakan untuk mendeteksi landmark pada tubuh peraga. Framework MediaPipe dipilih karena kemampuannya dalam mendeteksi titik-titik kunci (keypoints) pada tubuh manusia dengan presisi tinggi dan dalam berbagai kondisi pencahayaan dan posisi. Setelah mendapatkan landmark, titik-titik yang relevan akan digambarkan dengan garis yang membentuk kerangka yang sesuai dengan pose tubuh peraga.

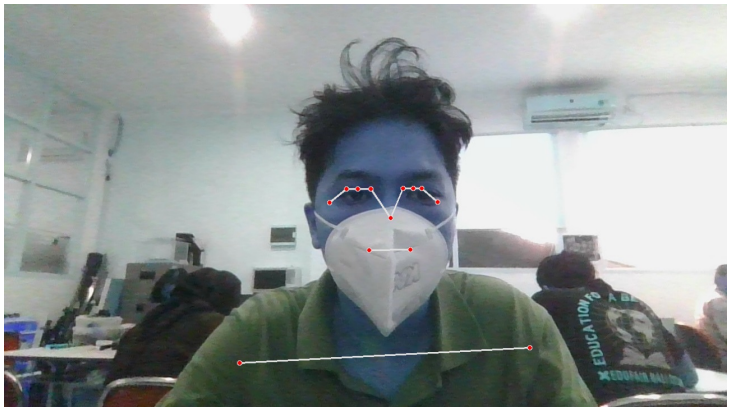
Proses deteksi dimulai dengan inisialisasi kamera untuk menangkap frame citra secara real-time. Setelah frame citra diterima, dilakukan pra-pengolahan seperti konversi gambar menjadi skala abu-abu untuk mengurangi kompleksitas dan meningkatkan kecepatan deteksi. Gambar yang telah dipra-pengolah tersebut kemudian diumpankan ke model MediaPipe untuk mendeteksi pose.

Dalam penelitian ini, beberapa landmark yang dianggap relevan adalah keypoint pada siku, lengan bawah, dan bahu kanan serta kiri. Titik-titik keypoint ini dipilih berdasarkan visibilitas dan konsistensi dalam deteksi. Tabel 3.2 menunjukkan nomor dan nama keypoint yang digunakan dalam estimasi pose:

Tabel 3.2: Tabel Keypoint yang digunakan

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

Setelah mendapatkan titik-titik landmark, jarak antar titik pada piksel diukur. Pengukuran ini dilakukan dengan menghitung jarak Euclidean antara dua titik kunci, yang merupakan metode yang efisien untuk menghitung jarak dalam ruang dua dimensi. Nilai piksel ini kemudian digunakan sebagai acuan dalam perhitungan untuk menghindari obstacle. Gambar 3.16 dan 3.17 adalah contoh visualisasi dari landmark pada data citra yang dihasilkan:



Gambar 3.16: Contoh hasil deteksi pose.

Gambar 3.17: Contoh hasil deteksi pose seluruh tubuh.

Penggunaan landmark pada lengan dan bahu sebagai acuan dalam perhitungan didasarkan pada visibilitas yang lebih baik dibandingkan dengan landmark lain seperti kaki dan kepala, terutama dalam konteks penghindaran manusia. Dalam jarak dekat, kedua landmark ini lebih mudah terlihat dan menghasilkan nilai yang lebih konsisten. Hal ini penting dalam aplikasi penghindaran obstacle, di mana kecepatan dan akurasi deteksi sangat krusial untuk menghindari tabrakan.

Selain itu, penggunaan landmark ini juga meminimalisir kemungkinan hilangnya titik kunci dalam berbagai kondisi pencahayaan dan sudut pandang. Pengambilan titik-titik kunci yang konsisten sangat penting untuk memastikan bahwa data yang diperoleh dapat diandalkan untuk pengambilan keputusan dalam sistem otonom. Dengan menggunakan landmark yang konsisten, sistem dapat mempertahankan kinerja yang tinggi meskipun terdapat variasi dalam kondisi lingkungan.

Sederhananya, Deteksi pose dengan MediaPipe bekerja dengan mengekstraksi fitur visual dari citra input. Framework ini mengidentifikasi dan menandai titik-titik kunci pada tubuh manusia yang kemudian dihubungkan untuk membentuk kerangka. Data landmark ini diambil dalam bentuk koordinat piksel, yang kemudian digunakan untuk berbagai aplikasi seperti penghindaran obstacle, pengenalan aktivitas, dan kontrol gerakan.

Selain itu, jarak antar titik kunci dihitung dengan menggunakan rumus Euclidean Distance, yang memungkinkan pengukuran jarak antara dua titik dalam ruang piksel. Informasi ini digunakan untuk memperkirakan posisi relatif dari bagian tubuh tertentu dan mengatur respons sistem sesuai dengan data yang diterima. Dengan pendekatan ini, sistem dapat mendeteksi dan merespons perubahan posisi manusia dalam lingkungan secara real-time, meningkatkan akurasi dan keandalan dalam situasi penghindaran obstacle.

3.3.5 Pemrosesan Citra

Pemrosesan citra dalam tugas akhir ini bertujuan untuk menentukan jarak dan posisi manusia yang menjadi obstacle bagi kursi roda otonom. Langkah pertama dalam pemrosesan ini adalah menggunakan model yang telah dilatih menggunakan YOLOV8 untuk mendeteksi kelas manusia. Setelah mendeteksi manusia, bounding box akan digambar di sekitar objek yang terdeteksi. Di dalam bounding box tersebut, akan digambar pula estimasi pose menggunakan MediaPipe. Kedua hasil deteksi ini, yaitu bounding box dan pose, akan digunakan untuk mendapatkan estimasi jarak berdasarkan rumus-rumus yang akan dijabarkan. Hasil dari estimasi jarak ini akan dipetakan ke dalam grid. Grid ini akan menjadi acuan bagi kursi roda untuk menentukan keputusan navigasi, seperti arah belok yang harus diambil.

Dalam tugas akhir ini, terdapat beberapa variabel utama yang akan diukur, yaitu jarak manusia, lebar manusia, dan posisi manusia. Perhitungan jarak manusia dibedakan menjadi dua metode, yaitu berdasarkan bounding box dan pose.

3.3.5.1 Perhitungan Jarak Obstacle berdasarkan Bounding Box

Salah satu cara untuk menghitung jarak dengan menggunakan bounding box adalah melalui konsep *focal length pixel* (f_p). Fokus panjang dalam piksel, atau *focal length pixel*, adalah konversi dari fokus panjang lensa kamera yang biasanya diukur dalam milimeter ke dalam satuan piksel. Ini merupakan konsep kunci dalam fotogrametri dan visi komputer yang digunakan untuk menghubungkan informasi visual dari kamera ke ukuran fisik di dunia nyata.

Dalam konteks tugas akhir ini, fokus panjang dalam piksel digunakan untuk mengkonversi ukuran obstacle dari unit piksel menjadi unit meter. Hal ini penting karena kursi roda otonom perlu memahami jarak nyata ke obstacle untuk mengambil keputusan navigasi yang tepat. Rumusannya dapat dilihat pada Persamaan 3.1 :

$$f_p = \frac{D \times h_b}{H_o} \tag{3.1}$$

Dimana:

- f_p adalah *focal length* dalam piksel,
- D adalah jarak sebenarnya dari kamera ke objek,
- h_b adalah tinggi bounding box dalam piksel,
- H_o adalah tinggi sebenarnya dari objek.

Fokus panjang dalam piksel adalah parameter penting dalam perhitungan jarak. Parameter 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, digunakan tinggi dari bounding box yang terdeteksi oleh YOLO, dikombinasikan dengan tinggi nyata objek yang diketahui dan fokus panjang dalam piksel. Sehingga membentuk Persamaan 3.2 sebagai berikut:

$$D = \frac{f_p \times H_o}{h_b} \tag{3.2}$$

Di sini:

- H_o merupakan tinggi rata-rata manusia atau objek lain yang diidentifikasi sebagai obstacle.

Dengan menggunakan tinggi bounding box yang dideteksi oleh YOLOv8, kursi roda dapat menghitung jarak ke objek tersebut. Hal ini sangat penting untuk menghindari benturan dan pengambilan keputusan navigasi.

3.3.5.2 Perhitungan Jarak Obstacle berdasarkan Pose

Pendekatan lain dalam menentukan jarak dengan menggunakan pose adalah melalui 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 otonom dengan integrasi MediaPipe, pengukuran ini sangat penting untuk berbagai fungsi, terutama dalam analisis pose dan penilaian proporsi objek dalam citra yang dihasilkan oleh kamera. Rumusannya dapat dilihat pada Persamaan 3.3

$$d_p = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \times s_f \tag{3.3}$$

Dimana:

- d_p adalah jarak Euclidean dalam piksel,
- (x_1, y_1) dan (x_2, y_2) adalah koordinat dua titik yang diukur dalam piksel,
- s_f adalah faktor skala.

Rumus ini menghasilkan jarak antara dua titik dalam satuan yang sama dengan satuan koordinat x_1, y_1 dan x_2, y_2. Biasanya, jika koordinat-koordinat ini diukur dalam piksel, maka jarak yang dihasilkan juga akan dalam piksel.

Penambahan faktor skala dalam perhitungan jarak Euclidean berguna ketika perlu mengonversi jarak dari satu unit ke unit lain, atau ketika koordinat titik disesuaikan ke skala tertentu yang tidak merefleksikan dimensi sebenarnya dalam piksel. Untuk menghitung jarak nyata dalam piksel, perlu mendapatkan hasil kali dari rumus Euclidean dengan dimensi gambar.

Dari perhitungan di atas, masih didapatkan jarak yang bernilai piksel. Dalam konteks perhitungan jarak, perlu menggunakan nilai standar dalam meter agar perhitungan tersebut sesuai dengan kaidah yang berlaku pada umumnya. Agar nilai piksel tersebut dapat diubah menjadi meter, perlu dilakukan kalibrasi menggunakan nilai K. Nilai K merupakan nilai kalibrasi berdasarkan pengukuran eksperimental. Sehingga didapat Persamaan 3.4 sebagai berikut:

$$D_m = \left(\frac{K}{d_p}\right) \tag{3.4}$$

Dimana:

- D_m adalah jarak dalam meter,
- K adalah konstanta kalibrasi,
- d_p adalah jarak dalam piksel.

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 dapat berdampak pada keputusan navigasi kursi roda otonom. Berikut rumus untuk pengkalibrasian nilai K menggunakan Persamaan 3.5:

$$K = J_o \times U_p \tag{3.5}$$

Dimana:

- J_o adalah jarak nyata objek,
- U_p adalah ukuran objek dalam piksel.

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

3.3.5.3 Perhitungan Lebar Obstacle Berdasarkan Pose

Dalam perhitungan lebar obstacle, digunakan perhitungan jarak Euclidean, namun dengan konteks aplikasi yang berbeda. Kedua pendekatan ini berbeda dalam aplikasi dan konteksnya. Perbedaan tersebut dapat dilihat pada Persamaan 3.6

$$l_p = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \times s_f \tag{3.6}$$

Nilai output yang didapatkan adalah lebar manusia dalam satuan piksel. Untuk memetakan lebar dari piksel ke meter, perlu dibuat rumus konversi yang menentukan ukuran dalam piksel (yang merupakan ukuran digital dan relatif) ke ukuran dunia nyata (meter). Rumusannya dapat dilihat pada Persamaan 3.7:

$$l_m = l_p \times s_f \tag{3.7}$$

Dengan mengalikan lebar objek dalam piksel dengan faktor skala, hasilnya adalah lebar objek dalam meter. Rumus ini berguna dalam aplikasi yang perlu 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 pada Persamaan 3.8:

$$s_f = \frac{D_n}{U_p} \tag{3.8}$$

Dimana:

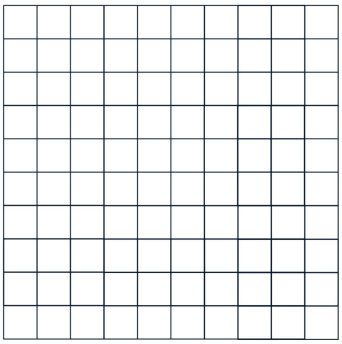
- D_n adalah dimensi nyata rata-rata,
- U_p adalah ukuran dalam piksel rata-rata.

Nilai-nilai ini dapat digunakan dalam kode untuk mengonversi ukuran dalam piksel ke ukuran nyata berdasarkan pengukuran yang telah dikalibrasi. Kalibrasi harus dilakukan dengan baik dan benar untuk mendapatkan hasil yang akurat.

3.3.5.4 Implementasi Grid Deteksi

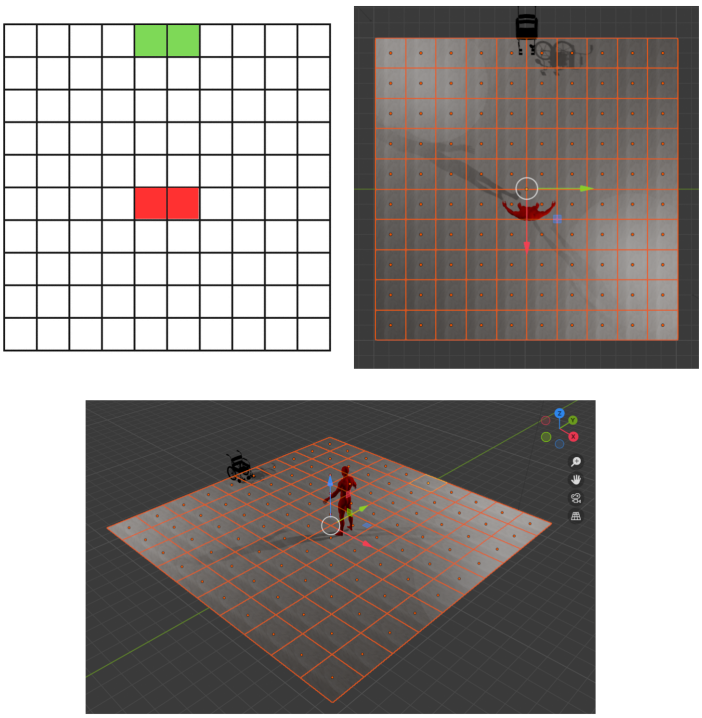
Dalam tugas akhir ini, kursi roda otonom harus dapat mengetahui posisi obstacle yang akan dilaluinya. Oleh karena itu, diperlukan sebuah peta yang dapat digunakan sebagai acuan kursi roda untuk mengambil tindakan berdasarkan jarak obstacle dan lebar obstacle yang akan menjadi acuan untuk menjawab permasalahan, yaitu di titik mana kursi roda harus menghindar dan apakah obstacle berhasil dihindari. Grid menjadi salah satu pendekatan terbaik dalam memetakan hasil deteksi. Penggunaan grid tidak hanya mempermudah pengambilan tindakan, tetapi juga memberikan visualisasi yang mudah dimengerti. Ukuran grid juga dapat disesuaikan sesuai dengan kebutuhan, baik dimensi maupun tampilannya.

Setelah perhitungan rumus di atas diimplementasikan dalam kode, akan didapatkan beberapa variabel penting yang akan digunakan dalam memetakan posisi maupun ukuran obstacle dalam grid. Sebelum itu, berikut adalah tampilan grid yang digunakan dapat dilihat pada Gambar 3.18 sebagai berikut:



Gambar 3.18: Grid 10x10.

Grid dibuat menggunakan library OpenCV. Library ini digunakan atas dasar implementasinya yang mudah dan ringan. Agar Lebih mudah dimengerti, penjelasan akan disertai dengan visualisasi 3d melalui blender pada Gambar 3.19.

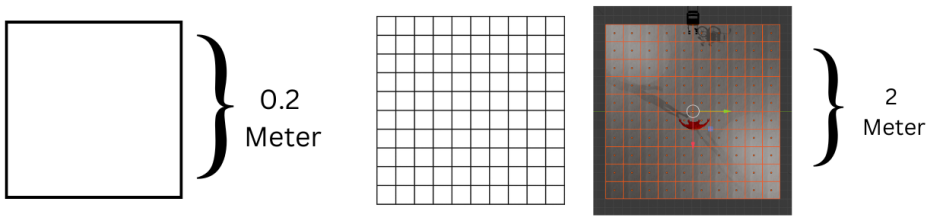


Gambar 3.19: Visualisasi Grid dengan blender.

Pada gambar 3.19 terdapat visualisasi 3d yang kondisinya sudah disesuaikan dengan grid 10x10 yang digunakan. Prespektif yang digunakan untuk menggambarkan kursi roda berada

diatas, dimana ini berdasarkan limitasi yang ada pada citra yang mana pengambilan titik koordinat pixel (0,0) berada pada pojok kiri atas. Sehingga agar grid dapat berfungsi sebagaimana mestinya maka prespektif kursi roda harus berada pada titik atas.

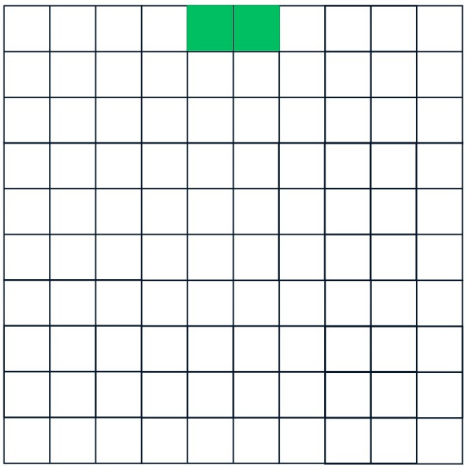
Penggunaan grid 10x10 didasari atas hasil citra dan performa model. Baik bounding box maupun pose memiliki keterbatasan di mana hasil perhitungannya tidak melebihi parameter yang akan ditetapkan pada grid. Parameter tersebut dapat dilihat pada Gambar 3.20 sebagai berikut:



Gambar 3.20: Parameter untuk setiap kotak grid.

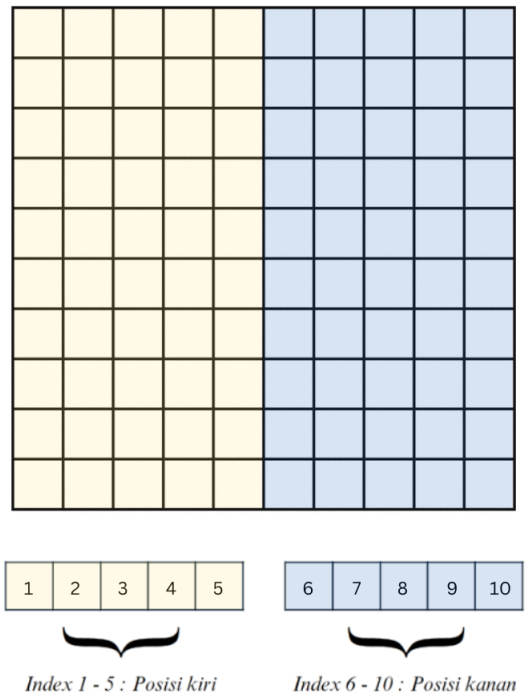
Dari gambar di atas, setiap kotak dalam grid bernilai 0.2 meter baik dalam posisi vertikal maupun horizontal. Penentuan nilai 0.2 meter per kotak ini didasarkan pada hasil kalibrasi dan pengukuran eksperimental yang memastikan ukuran ini optimal untuk deteksi obstacle yang akurat dan efisien.

Untuk memetakan objek dengan baik dalam grid, perlu dibuat indeks untuk mengetahui posisi relatif kiri dan kanan objek terhadap kamera. Untuk 10 kotak horizontal, akan dibagi menjadi indeks kiri dan kanan, di mana indeks 1 sampai 5 dikategorikan sebagai indeks kiri, dan indeks 6 hingga 10 dikategorikan sebagai indeks kanan. Pengambilan keputusan ini terkait dengan posisi kursi roda statis pada grid, yang digambarkan pada Gambar 3.21 sebagai berikut:



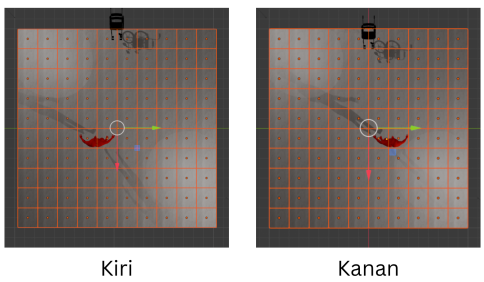
Gambar 3.21: Posisi Kursi Pada grid.

Dapat dilihat bahwa posisi kursi roda mengambil 2 kotak grid hijau pada bagian (5,1) dan (6,1). Keputusan ini didasarkan pada lebar kursi roda yang sesuai dengan ukuran grid. Posisi atas menentukan indeks mana yang merupakan bagian kiri maupun kanan dalam grid. Dengan posisi atas yang ditetapkan sebagai posisi konstan kursi roda dan hasil input citra yang didapatkan berlawanan dengan hasil deteksi (mirror), maka indeks dapat digambarkan sesuai Gambar 3.22 sebagai berikut:



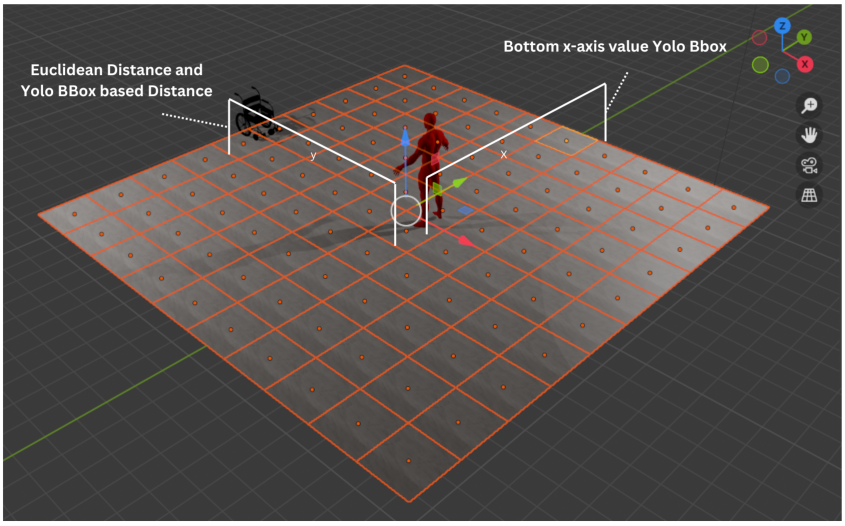
Gambar 3.22: Kategori Posisi berdasarkan Index.

Pengkategorian ini berperan penting dalam pengambilan keputusan belok kursi roda, di mana nantinya hasil deteksi yang berupa jarak, lebar, serta posisi relatif akan ditampilkan pada grid. Gambar 3.23 merupakan visualisasi 3d untuk kategori posisi berdasarkan index.



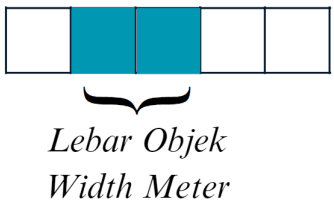
Gambar 3.23: Visualisasi Posisi berdasarkan Index dengan blender.

Secara horizontal, posisi objek hasil deteksi diambil berdasarkan nilai perpindahan titik X bounding box bawah relatif terhadap piksel. Nilainya disesuaikan dengan resolusi frame yang didapatkan oleh kamera. Akan dibuat patokan untuk posisi tertentu yang melambangkan perpindahan sebesar 0.2 meter secara horizontal. Dengan asumsi bahwa kursi roda bergerak dan objek diam, hanya posisi objek yang terdeteksi di depan yang akan menjadi patokan penghindaran. Jika objek terdeteksi namun saat berjalan maju objek menghilang, kursi roda tidak perlu menghindar. Oleh karena itu, ditetapkan parameter jarak pada objek yang terdeteksi berdasarkan vertikal. Sebelum membahas lebih lanjut, berikut dijelaskan secara rinci mengenai pemetaan hasil deteksi melalui variabel yang didapat melalui Gambar 3.24.



Gambar 3.24: Visualisasi Variabel Dalam perpindahan Posisi terhadap Grid dengan blender.

Berdasarkan perhitungan rumus yang telah dijelaskan, obstacle yang dideteksi dapat dipetakan posisinya relatif terhadap sumbu x dan y pada grid. 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 ukuran objek yang akan dilalui, sehingga diperlukan variabel tambahan yang dapat membuat grid menggambar lebar objek relatif terhadap piksel yang nantinya akan dipetakan dalam grid. Gambar 3.25 berikut merupakan pemetaan lebar objek pada grid:



Gambar 3.25: Lebar Objek dalam grid.

Dapat dilihat pada gambar di atas, pemetaan lebar berdasarkan nilai yang didapatkan pada variabel *width meter*. Sesuai dengan parameter ukuran kotak yang telah ditentukan, untuk se-

tiap penambahan 0.2 meter ukuran lebar hasil deteksi yang didapatkan akan menambah jumlah kotak yang akan ditampilkan.

Adapun beberapa kondisi di mana perlu dilakukan kalibrasi mengingat bounding box yang ditampilkan tidak sepenuhnya dapat dengan akurat merepresentasikan posisi objek secara horizontal, sehingga perlu ditambahkan nilai kalibrasi untuk mengatasi limitasi bounding box yang kadang memiliki nilai tengah yang tidak konsisten. Penambahan nilai pada rumus di bawah dapat menyesuaikan nilai posisi x pada grid agar lebih mendekati nilai kenyataan.

$$Grid\_x = (\frac{posisi\_x\_bounding\_box + konstanta}{scale\_factor}) \tag{3.9}$$

Setelah pemetaan grid dilakukan, berikut parameter yang digunakan dalam pengambilan keputusan belok kursi roda:

- **Jarak Vertikal (Euclidean Distance)**: Jarak vertikal dihitung menggunakan rumus Euclidean Distance yang telah dijelaskan sebelumnya.
- **Posisi Horizontal**: Posisi horizontal diambil berdasarkan nilai x dari titik bawah bounding box, yang kemudian dikonversi ke dalam unit grid menggunakan faktor skala.
- **Lebar Obstacle**: Lebar obstacle dihitung berdasarkan nilai bounding box yang dihasilkan oleh YOLO, dan dikonversi ke dalam unit grid sesuai dengan parameter 0.2 meter per kotak.

Dengan parameter-parameter ini, kursi roda otonom dapat mengambil keputusan yang tepat untuk menghindari obstacle berdasarkan posisi relatif dan ukuran obstacle yang dipetakan dalam grid.

3.3.5.5 Navigasi Penghindaran Kursi Roda

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.

Syarat utama agar dapat menginisiasi perintah ialah objek harus berada dibawah 1 Meter jarak terdeteksi. Dalam kondisi ini akan dibagi menjadi 3 pengambilan keputusan. yaitu sebagai berikut:

3.3.5.5.1 Hasil Tanpa Deteksi

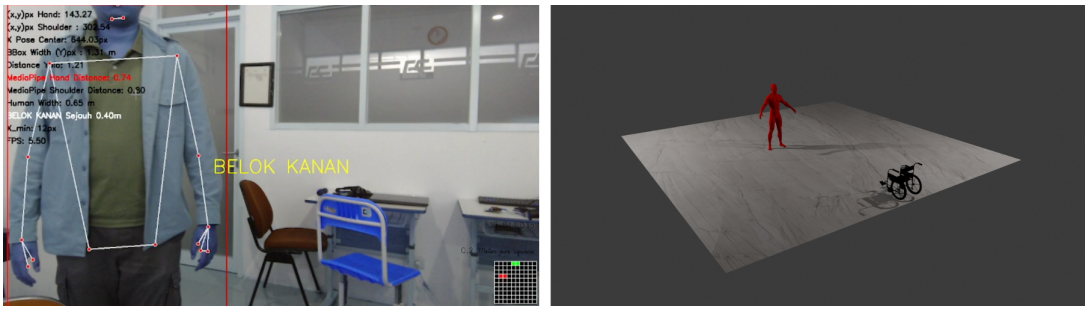
Pada kondisi ini tidak ada objek yang terdeteksi, yang berarti tidak ada obstacle yang menghalangi jalan kursi roda oleh karena itu kursi roda akan terus bergerak maju. Pada kondisi ini bounding box, pose, grid dan lainya tidak ditampilkan karena tidak ada manusia yang terdetksi. dapat dilihat pada contoh gambar 3.26

Gambar 3.26: Contoh kondisi tidak terdeteksi Manusia.

Dapat dilihat pada gambar 3.26, terdapat teks ”Manusia Tidak Terdeteksi” yang berarti tidak ada objek manusia yang terdeteksi.

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

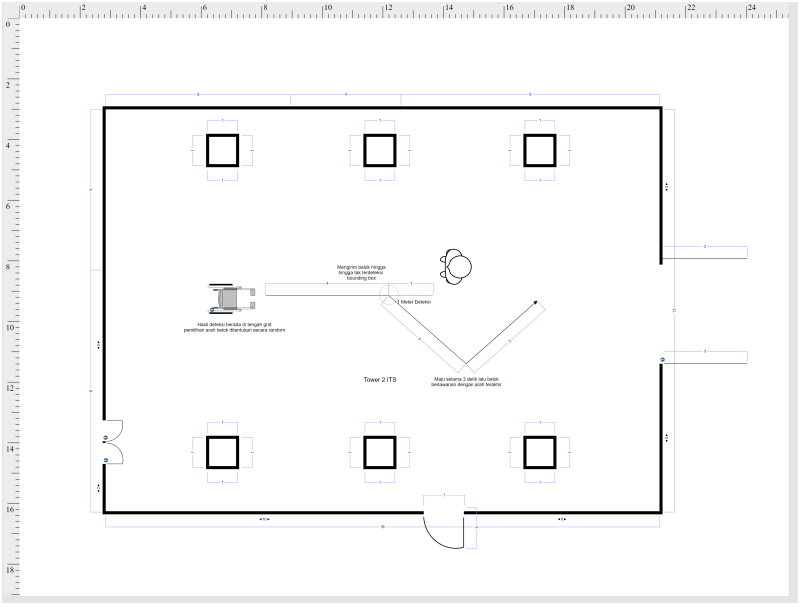
Pada kondisi ini posisi hasil deteksi menunjukan nilai yang lebih besar pada Index kiri (Index 1-5) yang berarti bahwa objek sedang berada pada sebelah kiri posisi terhadap kursi roda. Dapat dilihat pada contoh gambar dibawah.



Gambar 3.27: Contoh kondisi Index Kiri >Kanan.

Dapat dilihat pada gambar 3.27, 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.

Kursi roda baru dapat dikatakan menghindar jika dapat kembali ke arah asal, dengan demikian Gambar 3.28 adalah gambar skematik penghindaran yang akan dilalui dalam kondisi ini.

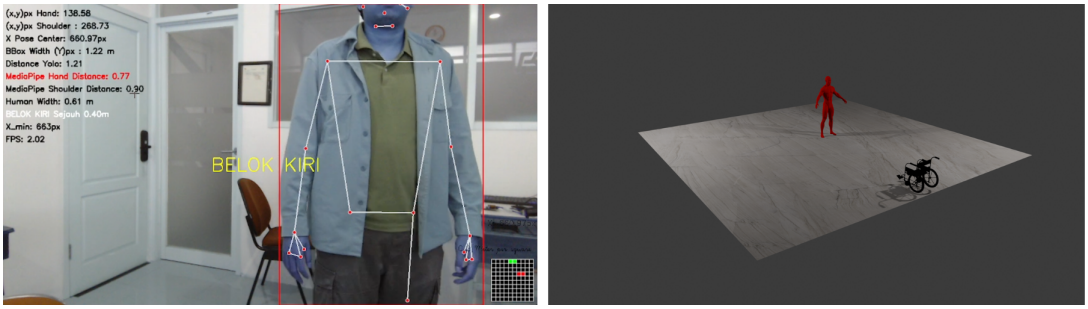


Gambar 3.28: Skematik penghindaran pada kondisi Index Kiri >Kanan.

Dapat dilihat pada Gambar 3.28 saat kursi roda sudah pada jarak 1 meter deteksi, maka akan berbelok sesuai dengan kondisi index dan menyimpan arah belokan ini. Apabila sudah tidak tergambar bounding box maka kursi roda akan maju selama 4 detik, lalu mengecek arah belokan terakhir. Setelah itu kursi roda akan Belok lagi sesuai dengan arah terakhir selama 5 detik, lalu mereset kondisi arah terakhir. Terakhir pada kondisi ini kursi roda akan masuk ke kondisi tanpa deteksi, sehingga ia akan terus bergerak maju hingga deteksi selanjutnya.

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

Pada kondisi ini posisi hasil deteksi menunjukan 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 3.29

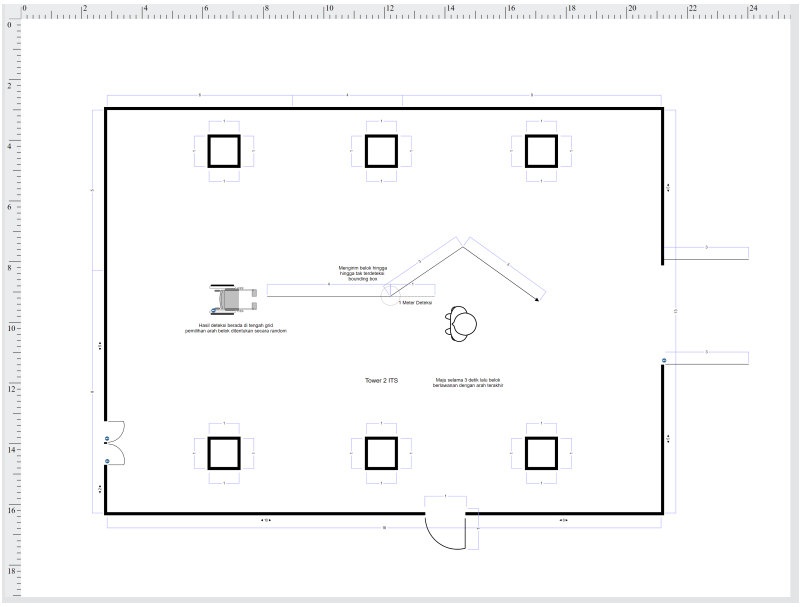


Gambar 3.29: Contoh kondisi Index Kanan >Kiri.

Dapat dilihat pada gambar 3.29, 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.

Kursi roda baru dapat dikatakan menghindar jika dapat kembali ke arah asal, dengan

demikian Gambar 3.30 adalah gambar skematik penghindaran yang akan dilalui dalam kondisi ini.

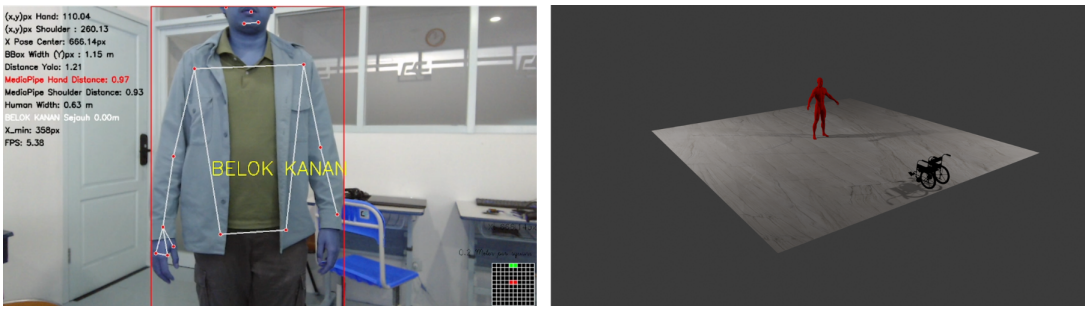


Gambar 3.30: Skematik penghindaran pada Contoh kondisi Index Kanan >Kiri.

Dapat dilihat pada Gambar 3.30 saat kursi roda sudah pada jarak 1 meter deteksi, maka akan berbelok sesuai dengan kondisi index dan menyimpan arah belokan ini. Apabila sudah tidak tergambar bounding box maka kursi roda akan maju selama 4 detik, lalu mengecek arah belokan terakhir. Setelah itu kursi roda akan Belok lagi sesuai dengan arah terakhir selama 5 detik, lalu mereset kondisi arah terakhir. Terakhir pada kondisi ini kursi roda akan masuk ke kondisi tanpa deteksi, sehingga ia akan terus bergerak maju hingga deteksi selanjutnya.

3.3.5.5.4 Hasil deteksi objek pada grid menunjukan posisi Linear terhadap kursi roda

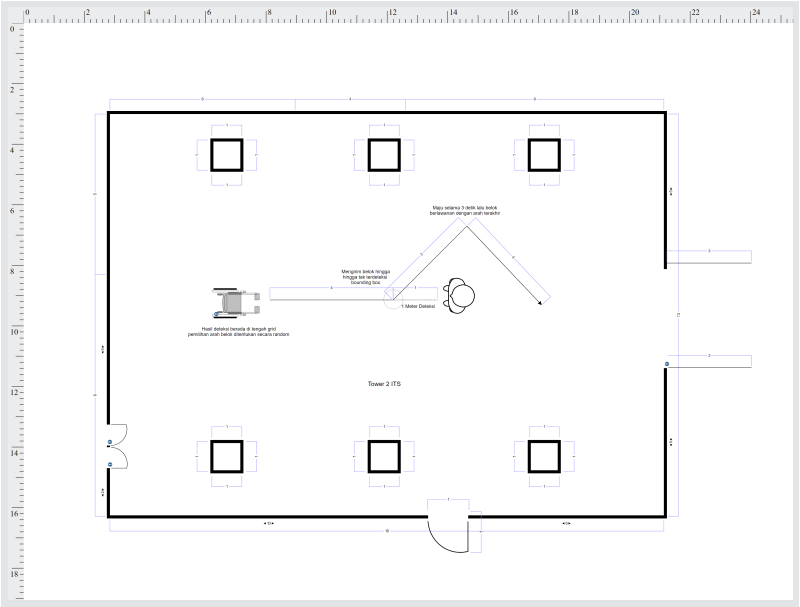
Pada kondisi ini perlu ditambahkan sebuah perintah untuk pengambilan keputusan dimana posisi belok pada kondisi ini baik melalui kanan maupun kiri tidak akan memiliki kelebihan/kekurangan karena dalam posisi ini nilai index sama besarnya baik kanan maupun kiri. sehingga perlu dilakukan pendekatan yang 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 3.31: Contoh kondisi Liniear.

Dapat dilihat pada gambar 3.31, gridnya sejajar dengan posisi kursi roda. Sehingga penggunaan random akan sangat berguna untuk mengabil keputusan apabila menghadapi kasus seperti ini.

Kursi roda baru dapat dikatakan menghindar jika dapat kembali ke arah asal, dengan demikian Gambar 3.32 adalah gambar skematik penghindaran yang akan dilalui dalam kondisi ini.



Gambar 3.32: Skematik penghindaran pada Contoh kondisi Linier.

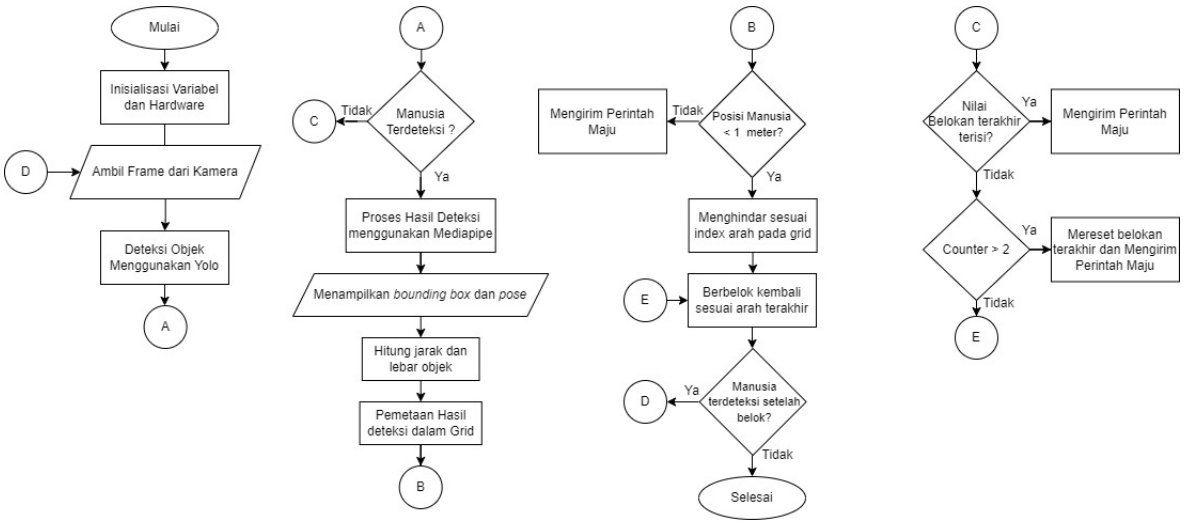
Dapat dilihat pada Gambar 3.32 saat kursi roda sudah pada jarak 1 meter deteksi, maka akan berbelok sesuai dengan kondisi index dan menyimpan arah belokan ini. Apabila sudah tidak tergambar bounding box maka kursi roda akan maju selama 4 detik, lalu mengecek arah belokan terakhir. Setelah itu kursi roda akan Belok lagi sesuai dengan arah terakhir selama 5 detik, lalu mereset kondisi arah terakhir.

3.3.6 Kode Program

Pada subbab ini akan dijabarkan Kode program yang telah dibuat dalam tugas akhir ini.

3.3.6.1 Program Untuk Penghindaran Manusia

Dalam Tugas Akhir ini agar kursi roda dapat menghindari obstacle maka kita harus dapat membaca citra manusia dalam frame sesuai dengan penjelasan pada subbab sebelumnya. Dimana berikut merupakan program utama yang menerima input citra hingga berhasil melakukan penghindaran. Gambar 3.33 merupakan *Flowchart* dari program 5.1 pada bab lampiran.



Gambar 3.33: Flowchart Program untuk penghindaran Manusia.

Flowchart pertama menggambarkan proses kerja program untuk penghindaran rintangan pada kursi roda otonom menggunakan YOLOv8 dan Mediapipe. Proses dimulai dengan inisialisasi variabel dan hardware yang diperlukan, yang digunakan dalam sistem seperti kamera, NUC dan Esp32. Setelah inisialisasi, sistem mengambil frame dari kamera yang terpasang pada kursi roda untuk deteksi objek. Frame yang diambil kemudian diproses menggunakan model YOLOv8 untuk mendeteksi objek. Hasil deteksi YOLO diperiksa apakah ada manusia yang terdeteksi. Jika tidak ada manusia yang terdeteksi, sistem akan mengirim perintah untuk maju dan kursi roda akan terus bergerak maju sesuai perintah yang diberikan.

Jika manusia terdeteksi, hasil deteksi YOLO akan diproses lebih lanjut menggunakan Mediapipe untuk mendapatkan pose dan landmark dari manusia yang terdeteksi. Sistem kemudian menampilkan bounding box dan pose dari manusia yang terdeteksi, membantu dalam visualisasi dan verifikasi deteksi yang dilakukan oleh YOLO dan Mediapipe. Berdasarkan bounding box dan pose yang dihasilkan, sistem menghitung jarak dan lebar objek (manusia), informasi penting untuk menentukan apakah kursi roda harus menghindar atau tidak. Hasil deteksi dan perhitungan jarak kemudian dipetakan ke dalam grid 10x10 untuk menentukan posisi relatif manusia terhadap kursi roda.

Setelah pemetaan, sistem memeriksa apakah posisi manusia kurang dari 1 meter dari kursi roda. Jika ya, kursi roda akan menghindar sesuai dengan index arah pada grid. Kursi roda bergerak menghindar berdasarkan arah yang ditentukan oleh grid, memastikan kursi roda dapat menghindari manusia dengan aman. Setelah menghindar, kursi roda akan mencoba berbelok kembali ke arah jalur awalnya untuk memastikan kursi roda tetap berada di jalur yang benar

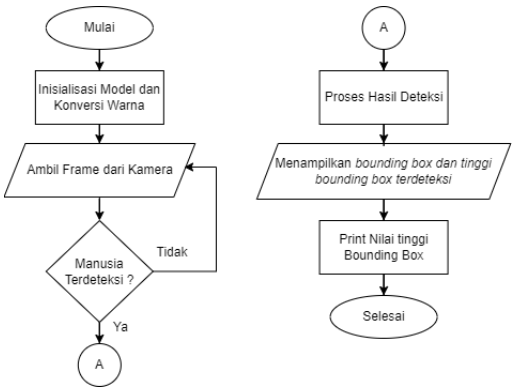
setelah menghindar. Sistem kembali memeriksa apakah masih ada manusia yang terdeteksi di jalur kursi roda setelah berbelok. Jika tidak ada, proses selesai dan kursi roda akan melanjutkan pergerakannya.

Apabila tidak terdapat deteksi pada manusia, maka akan diperiksa apakah terdapat nilai belokan terakhir, jika ya maka akan mengirim perintah maju. lalu diperiksa apakah nilai counter lebih dari atau sama dengan 2, jika ya maka akan mereset belokan terakhir dan mengirim perintah maju. Lalu apabila kedua syarat tersebut tidak terpenuhi yang berarti kursi roda telah menghindari seseorang maka akan maju lalu belok ke arah berlawanan yang berarti kursi roda telah kembali ke jalur utama

Flowchart ini memberikan gambaran jelas tentang bagaimana sistem penghindaran rintangan pada kursi roda otonom bekerja menggunakan YOLOv8 dan Mediapipe, mulai dari inisialisasi variabel dan hardware, pengambilan frame, deteksi objek, hingga penghindaran rintangan dan pengembalian ke jalur awal.

3.3.6.2 Program untuk Kalibrasi Focal Length

Dalam Tugas Akhir ini agar dapat mendeteksi jarak menggunakan Bounding Box maka diperlukan sebuah program kalibrasi dengan tujuan untuk menampilkan tinggi Bounding Box dalam pixel sesuai dengan penjelasan pada subbab sebelumnya. Gambar 3.34 merupakan *Flowchart* dari program 5.2 pada bab lampiran.



Gambar 3.34: Flowchart Program untuk Kalibrasi Focal Length.

Flowchart ini menggambarkan proses kerja Program untuk Kalibrasi yang menampilkan tinggi bounding Box. Proses dimulai dengan inisialisasi model dan konversi warna. Model YOLOv8 diinisialisasi menggunakan model yang telah dilatih (best100epoch.pt merupakan nama file model). Selanjutnya, frame diambil dari kamera yang telah diinisialisasi dengan resolusi tertentu. Frame yang diambil kemudian dikonversi ke format RGB untuk keperluan deteksi objek. Setelah itu, deteksi objek dilakukan menggunakan model YOLOv8 yang telah diinisialisasi sebelumnya.

Jika manusia terdeteksi dalam frame, proses hasil deteksi akan dilakukan. Hasil deteksi yang berupa bounding box akan diproses lebih lanjut untuk menampilkan bounding box dan tinggi bounding box dari manusia yang terdeteksi. Tinggi bounding box dihitung berdasarkan koordinat y dari bounding box yang terdeteksi. Jika bounding box memiliki tingkat kepercayaan (confidence) lebih dari 0.7 dan kelas objek adalah manusia, tinggi bounding box akan dihitung dan ditampilkan pada frame yang diproses. Sistem akan menggambar bounding box pada frame

dan menampilkan tinggi bounding box tersebut dalam satuan piksel. Proses ini terus berlangsung hingga pengguna menekan tombol ’q’ untuk keluar dari loop dan mengakhiri program. Video capture kemudian dilepaskan dan semua jendela OpenCV ditutup untuk mengakhiri program.

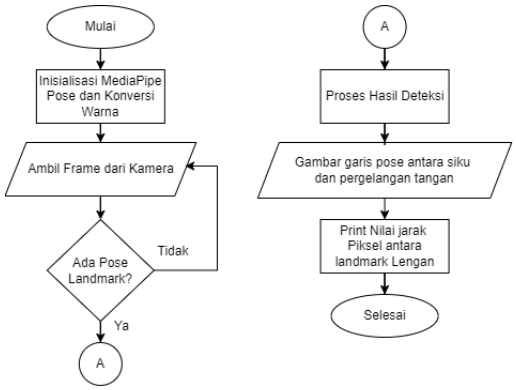
Kode Python yang sesuai menunjukkan langkah- langkah ini dengan menggunakan cv2.VideoCapture untuk menangkap frame dari kamera, konversi warna dengan cv2.cvtColor, deteksi objek menggunakan model.predict, dan menggambar bounding box serta tinggi bounding box pada frame dengan cv2.rectangle dan cv2.putText. Seluruh proses ini diatur dalam loop yang terus berjalan hingga pengguna menekan tombol ’q’ untuk keluar.

3.3.6.3 Program untuk Kalibrasi Euclidean Distance.

Pada Kalibrasi Euclidean Distance akan dibagi menjadi dua bagian yang pertama adalah pengambilan jarak pixel antar landmark lengan dan jarak pixel antar pixel landmark bahu.

3.3.6.3.1 Program Untuk mendapatkan Jarak pixel Landmar Lengan

Dalam Tugas Akhir ini agar dapat mendeteksi jarak menggunakan Pose maka diperlukan sebuah program kalibrasi dengan tujuan untuk menampilkan jarak antar Landmark lengan dalam pixel sesuai dengan penjelasan pada subbab sebelumnya. Gambar 3.35 merupakan *Flowchart* dari program 5.3 pada bab lampiran.



Gambar 3.35: Flowchart Program untuk Jarak Pixel Landmark Lengan.

Flowchart ini menggambarkan proses kerja sistem untuk mendeteksi pose manusia menggunakan MediaPipe dan menghitung jarak piksel antara landmark lengan. Proses dimulai dengan inisialisasi MediaPipe Pose dan konversi warna. MediaPipe Pose diinisialisasi untuk mendeteksi pose manusia dalam frame video. Selanjutnya, frame diambil dari kamera yang telah diinisialisasi dengan resolusi tertentu. Frame yang diambil kemudian dikonversi ke format RGB untuk keperluan deteksi pose.

Setelah itu, deteksi pose dilakukan menggunakan MediaPipe Pose yang telah diinisialisasi sebelumnya. Hasil deteksi pose diperiksa untuk mengetahui apakah ada landmark pose yang terdeteksi. Jika tidak ada landmark pose yang terdeteksi, sistem akan kembali mengambil frame baru dari kamera dan mengulangi proses deteksi. Jika ada landmark pose yang terdeteksi, proses hasil deteksi akan dilakukan.

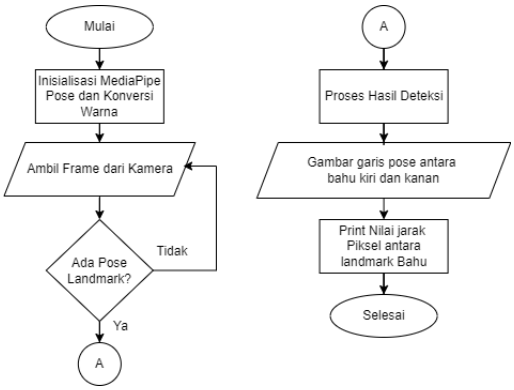
Proses hasil deteksi melibatkan penggambaran garis antara landmark siku dan pergelangan tangan pada frame yang diproses. Sistem akan menggambar garis ini untuk membantu visual-

isasi pose yang terdeteksi. Setelah menggambar garis, sistem akan menghitung dan mencetak jarak piksel antara landmark lengan (siku dan pergelangan tangan) pada frame yang diproses. Proses ini terus berlangsung hingga pengguna mengakhiri program.

Kode Python yang sesuai menunjukkan langkah-langkah ini dengan menggunakan cv2.VideoCapture untuk menangkap frame dari kamera, konversi warna dengan cv2.cvtColor, deteksi pose menggunakan pose.process, dan menggambar garis serta menghitung jarak piksel antara landmark lengan dengan cv2.line dan cv2.putText. Seluruh proses ini diatur dalam loop yang terus berjalan hingga pengguna menekan tombol 'q' untuk keluar. Video capture kemudian dilepaskan dan semua jendela OpenCV ditutup untuk mengakhiri program.

3.3.6.3.2 Program Untuk mendapatkan jarak pixel Landmark Bahu

Dalam Tugas Akhir ini agar dapat mendeteksi jarak menggunakan Pose maka diperlukan sebuah program kalibrasi dengan tujuan untuk menampilkan jarak antar Landmark Bahu dalam pixel sesuai dengan penjelasan pada subbab sebelumnya. Gambar 3.36 merupakan *Flowchart* dari program 5.3 pada bab lampiran.



Gambar 3.36: Flowchart Program untuk Jarak Pixel Landmark Bahu.

Flowchart ini menggambarkan proses kerja sistem untuk mendeteksi pose manusia menggunakan MediaPipe dan menghitung jarak piksel antara landmark bahu kiri dan bahu kanan. Proses dimulai dengan inisialisasi MediaPipe Pose dan konversi warna. MediaPipe Pose diinisialisasi untuk mendeteksi pose manusia dalam frame video. Selanjutnya, frame diambil dari kamera yang telah diinisialisasi dengan resolusi tertentu. Frame yang diambil kemudian dikonversi ke format RGB untuk keperluan deteksi pose.

Setelah itu, deteksi pose dilakukan menggunakan MediaPipe Pose yang telah diinisialisasi sebelumnya. Hasil deteksi pose diperiksa untuk mengetahui apakah ada landmark pose yang terdeteksi. Jika tidak ada landmark pose yang terdeteksi, sistem akan kembali mengambil frame baru dari kamera dan mengulangi proses deteksi. Jika ada landmark pose yang terdeteksi, proses hasil deteksi akan dilakukan.

Proses hasil deteksi melibatkan penggambaran garis antara landmark bahu kiri dan bahu kanan pada frame yang diproses. Sistem akan menggambar garis ini untuk membantu visualisasi pose yang terdeteksi. Setelah menggambar garis, sistem akan menghitung dan mencetak jarak piksel antara landmark bahu kiri dan bahu kanan pada frame yang diproses. Proses ini terus berlangsung hingga pengguna mengakhiri program.

Kode Python yang sesuai menunjukkan langkah-langkah ini dengan menggunakan cv2.Vid-

eoCapture untuk menangkap frame dari kamera, konversi warna dengan cv2.cvtColor, deteksi pose menggunakan pose.process, dan menggambar garis serta menghitung jarak piksel antara landmark bahu kiri dan bahu kanan dengan cv2.line dan cv2.putText. Seluruh proses ini diatur dalam loop yang terus berjalan hingga pengguna menekan tombol 'q' untuk keluar. Video capture kemudian dilepaskan dan semua jendela OpenCV ditutup untuk mengakhiri program.

BAB IV
PENGUJIAN DAN ANALISIS

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

4.1 Skenario Pengujian

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

1. Hasil Pengujian Performa Model
2. Pengujian Berdasarkan FPS
3. Pengujian Berdasarkan Hasil Response Time
4. Pengujian Kesesuaian Jarak Deteksi 150 cm
5. Pengujian Kesesuaian Jarak Deteksi 100 cm
6. Pengujian Kesesuaian Jarak Deteksi 50 cm
7. Performa Keberhasilan Penghindaran
8. Performa Akurasi Penghindaran
9. Performa Keberhasilan Penghindaran dengan Dua Obstacle
10. Performa keberhasilan Penghindaran dengan Tiga Obstacle
11. Performa Keberhasilan Penghindaran dengan Empat Obstacle

4.2 Hasil Pengujian Performa menggunakan Confusion Matrix

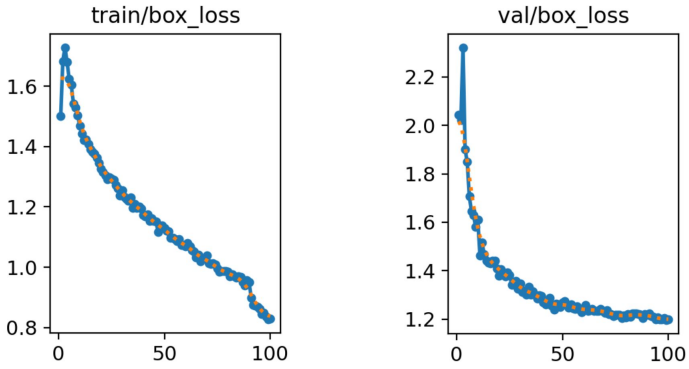
Data yang digunakan sebagai set data train berjumlah 1453 data citra manusia, sedangkat set data validasi berjumlah 341. Data-data tersebut akan dilakukan proses augmentasi sehingga menghasilkan 4359 set data train dan 1023 set data validasi. Setelah proporsi dataset ditentukan maka API key akan digenerate pada roboflow yang nantinya akan dipanggil untuk dilakukan proses training.

Setelah proses pemuatan set dilakukan, maka akan dilajutkan 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 skor mAP, precision, box loss. Gambar 4.1 merupakan Input layer yang akan digunakan.

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

Gambar 4.1: Input Layer Pelatihan Pertama

Pelatihan pertama menggunakan 100*epoch*. Input layer memiliki *batch size* sebesar 16 dengan tindakan praproses merubah ukuran gambar pada dataset menjadi 800x800 piksel. Adapun tujuan dari pelatihan ini untuk mengetahui seberapa baik peningkatan model pre- trained yang digunakan dalam deteksi manusia berdasarkan jumlah Epoch yang ditentukan. Berikut nilai box loss yang didapatkan dari hasil *training* pada Gambar 4.2

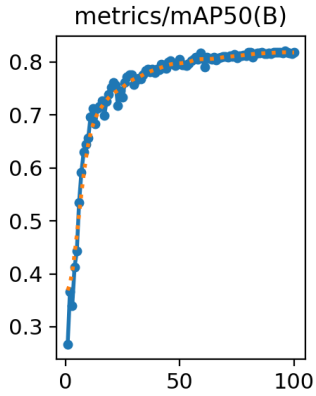


Gambar 4.2: Grafik Box Loss 100 Epoch

Didapatkan nilai box loss pada proses training menggunakan YOLOV8 ini adalah sebesar 0.82978 setelah 100 epoch. Box loss mengukur seberapa baik model memprediksi bounding boxes seputar objek. 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 1.199, adapun box loss pada validasi ini mengindikasikan kemampuan mengenali objek pada data

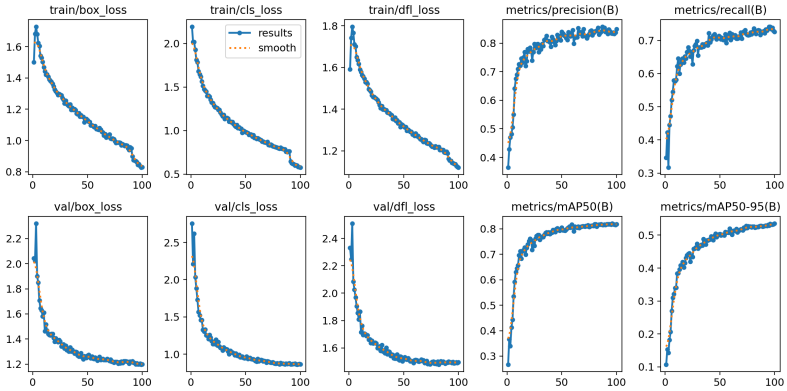
uji. Secara teori, penurunan object loss pada tahap validasi menandakan bahwa model mampu melakukan deteksi objek secara umum. bukan hanya pada data pelatihan. Selanjutnya Gambar 4.3 merupakan grafik nilai skor mAP.



Gambar 4.3: Grafik mAP 100 Epoch

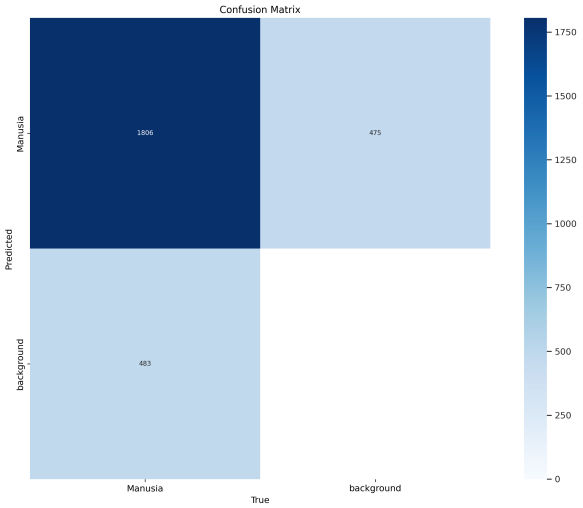
Skor mAP pada threshold 0.5 yang diperoleh dari proses ini adalah sebesar 81,85%, yang mengindikasikan bahwa model memiliki kemampuan yang sangat baik dalam mendeteksi objek dengan ketepatan tinggi, dengan syarat kriteria IoU minimal 0.5 terpenuhi. Hal ini menunjukkan bahwa dalam banyak percobaan kasus, bounding box yang diprediksi oleh model memiliki kesesuaian yang signifikan dengan bounding box ground truth.

Pada Gambar 4.4 merupakan grafik hasil yang didapatkan pada model secara keseluruhan. Didapatkan nilai- nilai pada masing masing grafik, pada train box_loss didapatkan nilai 0.82978, pada train cls_loss didapatkan nilai 0.57615, pada train dfl_loss didapatkan nilai 1.1197 , pada metrics precision didapatkan nilai 0.84974, pada metrics recall 0.72608, pada metrics mAP50 0.81855, pada metrics mAP50-95 0.53527, pada val box_loss didapatkan nilai 1.199, pada val cls_loss didapatkan nilai 0.86314, dan pada val dfl_loss didapatkan nilai 1.4956.



Gambar 4.4: Grafik Hasil *Training* 100 Epoch

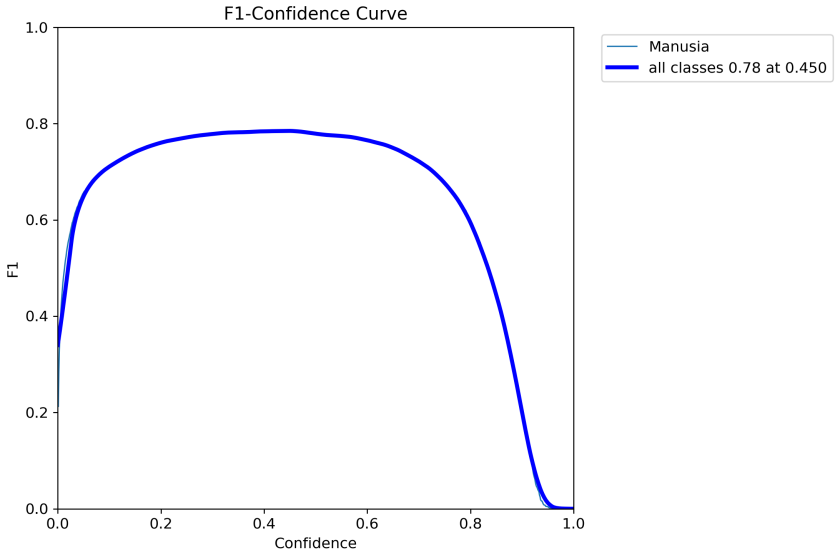
Adapun visualisasi melalui confusion matrix yang digunakan dapat dilihat pada Gambar 4.5, untuk merepresentasikan hasil model lebih detail. Adapun indikator pada confusion matrix pada setiap kotaknya merepresentasikan nilai *true positive*, *false positive*, *false negative*, *true negative*.



Gambar 4.5: Confusion Matrix Hasil Training 100 Epoch

Dapat dilihat pada dapat dilihat bahwa dari hasil klasifikasi model terdapat 1806 data citra yang termasuk dalam kategori true positive (Manusia yang benar terdeteksi), 483 citra yang termasuk dalam kategori false positive (Background yang salah terdeteksi sebagai Manusia), 475 citra yang termasuk false negative (Manusia yang tidak terdeteksi atau salah diklasifikasikan sebagai background) dan 0 citra yang termasuk true negative (karena tidak ada kelas lain selain Manusia).

Berikut adalah kurva F1-Confidence yang didapatkan dari hasil pelatihan model pada Gambar 4.6.



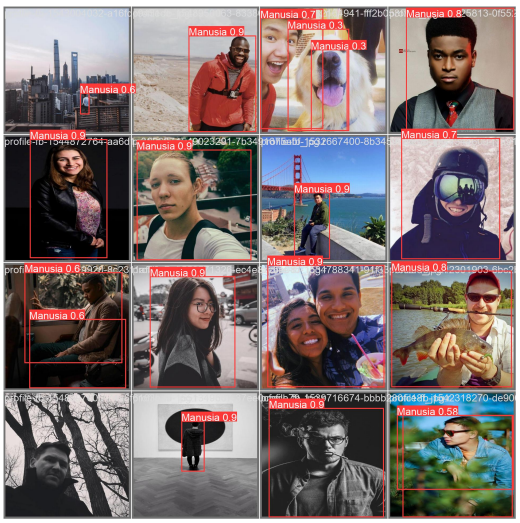
Gambar 4.6: Grafik F1-Confidence Hasil *Training* 100 Epoch

Kurva F1-Confidence menunjukkan hubungan antara nilai confidence dengan nilai F1-score. Pada kurva ini, terlihat bahwa nilai F1-score tertinggi yang dicapai oleh model adalah 0.78 pada nilai confidence 0.450. Nilai F1-score mengukur keseimbangan antara *precision* dan

recall dari model, dimana F1-score yang tinggi menunjukkan bahwa model memiliki keseimbangan yang baik antara *precision* dan *recall*.

Dari kurva tersebut, dapat disimpulkan bahwa model YOLOv8 mampu mendeteksi manusia dengan baik pada tingkat kepercayaan (*confidence*) tertentu. Kurva yang menurun tajam setelah nilai confidence sekitar 0.8 menunjukkan bahwa pada nilai confidence yang sangat tinggi, model cenderung mengabaikan banyak deteksi positif yang sebenarnya (*true positives*), sehingga menurunkan nilai F1-score secara keseluruhan.

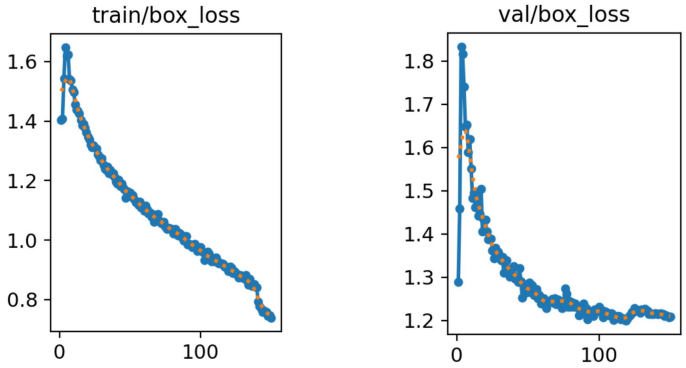
Dilakukan pula tes inference model yang telah dilatih menggunakan set data test terhadap object manusia. Dapat dilihat pada gambar 4.7 dan gambar 4.8 dibawah nilai confidence score yang cukup tinggi.



Gambar 4.7: Tes Inferensi Menggunakan Model Terlatih 100 Epoch dengan Dataset

Gambar 4.8: Tes Inferensi Menggunakan Model Terlatih 100 Epoch dengan Foto

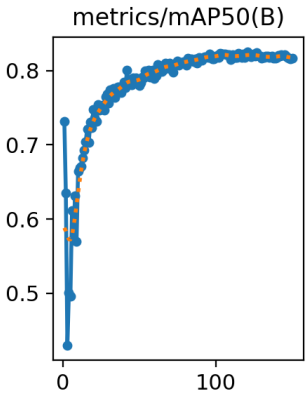
Pelatihan kedua menggunakan 150 *epoch* dan *batch size* sebesar 16 dengan tindakan pra proses merubah ukuran gambar pada dataset menjadi 800x800 piksel. Adapun tujuan dari pelatihan ini untuk mengetahui seberapa baik peningkatan model pre- trained yang digunakan dalam deteksi manusia berdasarkan jumlah Epoch yang ditentukan. Nilai box loss yang dihasilkan dapat dilihat pada gambar 4.9



Gambar 4.9: Grafik Box Loss 150 Epoch

Didapatkan nilai box loss pada proses training menggunakan YOLOV8 ini adalah sebesar 0.73832 setelah 150 epoch. Box loss mengukur seberapa baik model memprediksi bounding boxes seputar objek. 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.

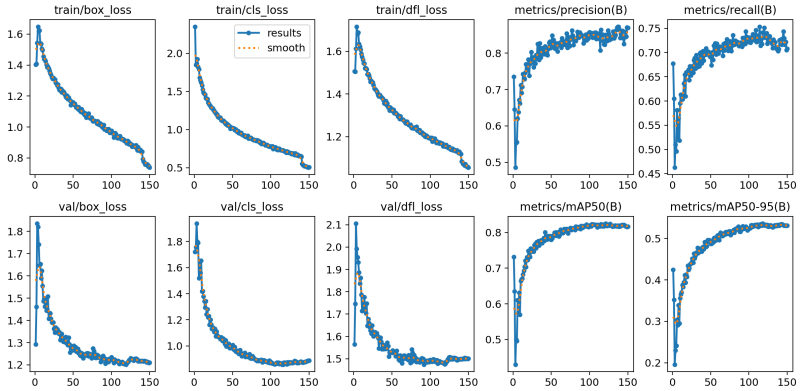
Didapatkan nilai box loss pada proses validasi menggunakan YOLOV8 ini sebesar 1.21, adapun box loss pada validasi ini mengindikasikan kemampuan mengenali objek pada data uji. Secara teori, penurunan object loss pada tahap validasi menandakan bahwa model mampu melakukan deteksi objek secara umum. bukan hanya pada data pelatihan. Selanjutnya skor mAP dapat dilihat pada Gambar 4.10



Gambar 4.10: Grafik mAP 150 Epoch

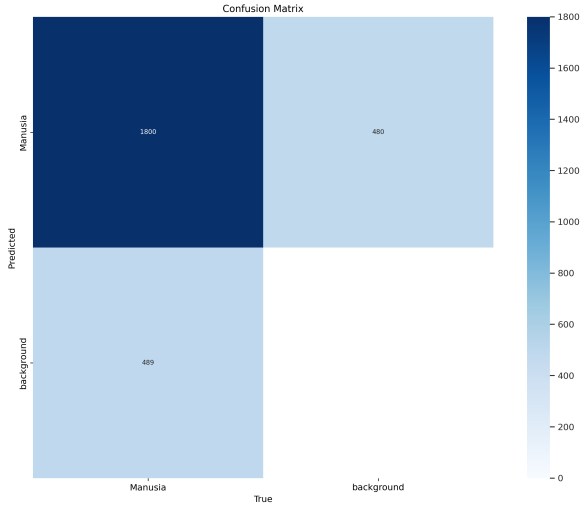
Skor mAP pada threshold 0.5 yang diperoleh dari proses ini adalah sebesar 81.68%, yang mengindikasikan bahwa model memiliki kemampuan yang sangat baik dalam mendeteksi objek dengan ketepatan tinggi, dengan syarat kriteria IoU minimal 0.5 terpenuhi. Hal ini menunjukkan bahwa dalam banyak percobaan kasus, bounding box yang diprediksi oleh model memiliki kesesuaian yang signifikan dengan bounding box ground truth.

Pada Gambar 4.11 merupakan grafik hasil yang didapatkan pada model secara keseluruhan. Didapatkan nilai- nilai pada masing masing grafik, pada train box _loss didapatkan nilai 0.73832, pada train cls _loss didapatkan nilai 0.50825, pada train dfl _loss didapatkan nilai 1.056 , pada metrics precision didapatkan nilai 0.86984, pada metrics recall 0.70773, pada metrics mAP50 0.81682, pada metrics mAP50-95 0.53086, pada val box _loss didapatkan nilai 1.21, pada val cls _loss didapatkan nilai 0.88665, dan pada val dfl _loss didapatkan nilai 1.5017



Gambar 4.11: Grafik Hasil *Training* 150 Epoch

Adapun visualisasi melalui confusion matrix yang digunakan dapat dilihat pada Gambar 4.12, untuk merepresentasikan hasil model lebih detail. Adapun indikator pada confusion matrix pada setiap kotaknya merepresentasikan nilai *true positive*, *false positive*, *false negative*, *true negative*.

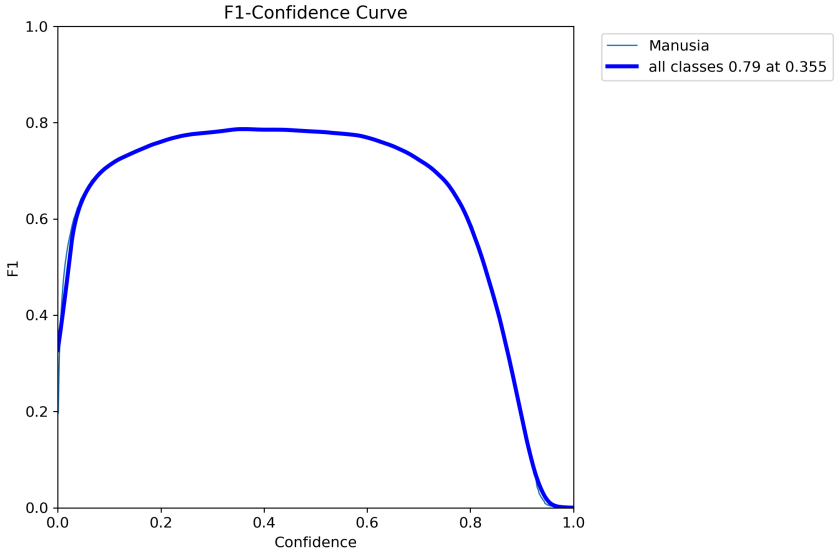


Gambar 4.12: Confusion Matrix Hasil Training 150 Epoch

Dapat dilihat pada dapat dilihat bahwa dari hasil klasifikasi model terdapat 1800 data citra yang termasuk dalam kategori true positive (Manusia yang benar terdeteksi), 489 citra yang termasuk dalam kategori false positive (Background yang salah terdeteksi sebagai Manusia), 480 citra yang termasuk false negative (Manusia yang tidak terdeteksi atau salah diklasifikasikan

sebagai background) dan 0 citra yang termasuk true negative (karena tidak ada kelas lain selain Manusia).

Berikut adalah kurva F1-Confidence yang didapatkan dari hasil pelatihan model pada Gambar 4.13.



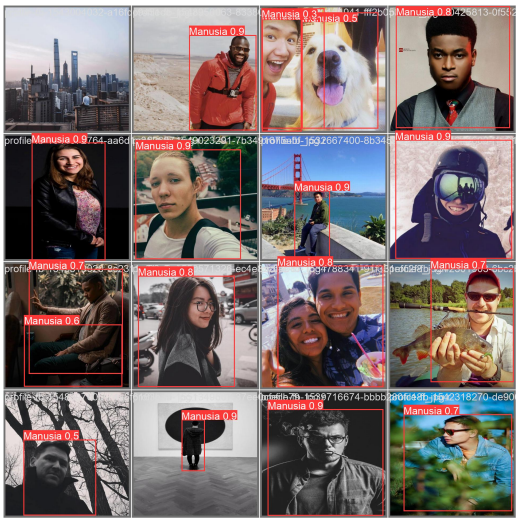
Gambar 4.13: Grafik F1-Confidence Hasil *Training* 150 Epoch

Kurva F1-Confidence menunjukkan hubungan antara nilai confidence dengan nilai F1-score. Pada kurva ini, terlihat bahwa nilai F1-score tertinggi yang dicapai oleh model adalah 0.79 pada nilai confidence 0.450. Nilai F1-score mengukur keseimbangan antara *precision* dan *recall* dari model, dimana F1-score yang tinggi menunjukkan bahwa model memiliki keseimbangan yang baik antara *precision* dan *recall*.

Dari kurva tersebut, dapat disimpulkan bahwa model YOLOv8 mampu mendeteksi manusia dengan baik pada tingkat kepercayaan (*confidence*) tertentu. Kurva yang menurun tajam setelah nilai confidence sekitar 0.8 menunjukkan bahwa pada nilai confidence yang sangat tinggi, model cenderung mengabaikan banyak deteksi positif yang sebenarnya (*true positives*), sehingga menurunkan nilai F1-score secara keseluruhan.

Adapun penurunan nilai F1-score setelah mencapai puncaknya menandakan bahwa meskipun model memiliki kepercayaan tinggi dalam prediksinya, model mungkin mengorbankan banyak deteksi yang benar untuk menghindari kesalahan positif (*false positives*). Oleh karena itu, penting untuk menentukan nilai threshold confidence yang optimal agar model dapat memberikan kinerja deteksi yang seimbang antara *precision* dan *recall*.

Dilakukan pula tes inference model yang telah dilatih menggunakan set data test terhadap object manusia. Dapat dilihat pada gambar 4.14 dan gambar 4.15 dibawah nilai confidence score yang cukup tinggi.



Gambar 4.14: Tes Inferensi Menggunakan Model Terlatih 150 Epoch dengan Dataset

Gambar 4.15: Tes Inferensi Menggunakan Model Terlatih 150 Epoch dengan Foto

Dari nilai nilai yang dijabarkan tersebut didapatkan indikasi bahwa penambahan jumlah epoch tidak berpengaruh signifikan terhadap kemampuan model untuk mengenali manusia dengan benar, hal tersebut didukung dari hasil yang didapatkan bahwa pada 150 Epoch model menunjukan sedikit indikasi overfitting. Sehingga dalam proses selanjutnya model dengan 100 epoch akan digunakan dalam pengujian selanjutnya.

4.3 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 sebanyak 30 kali yang sesuai dengan output sistem dan sudah diolah pada bagian sebelumnya.

Pada pengujian ini akan diambil nilai FPS pada laptop dan FPS pada NUC. Spesifikasi laptop yang digunakan untuk pengujian ini pada tabel 4.1 :

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

Spesifikasi NUC yang digunakan untuk pengujian ini pada tabel 4.2 :

Tabel 4.2: Spesifikasi NUC

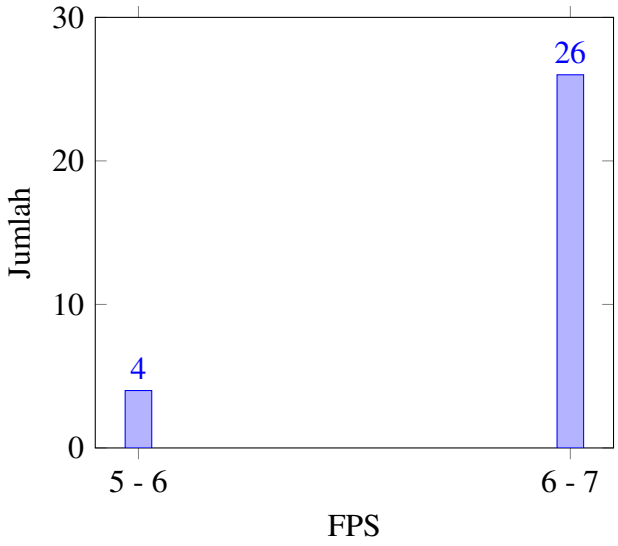
Komponen	Spesifikasi
CPU	Intel® Core™ i7-1165G7
RAM	32 GBLPDDR4
OS	Windows 11 Home Single Language

Dapat dilihat pada tabel 4.3 pada bagian kanan, didapatkan pada pengujian nilai fps laptop ini nilai rata - rata FPS sebesar 13.140. Dimana nilai FPS tertinggi adalah sebesar 13.23 dan FPS paling rendah ialah 13.05. Selain itu dapat dilihat pada tabel 4.3, didapatkan pada pengujian ini nilai rata - rata FPS pada intel NUC sebesar 6.111. Dimana nilai FPS tertinggi yang didapatkan ialah 6.47 dan FPS paling rendah ialah 5.54.

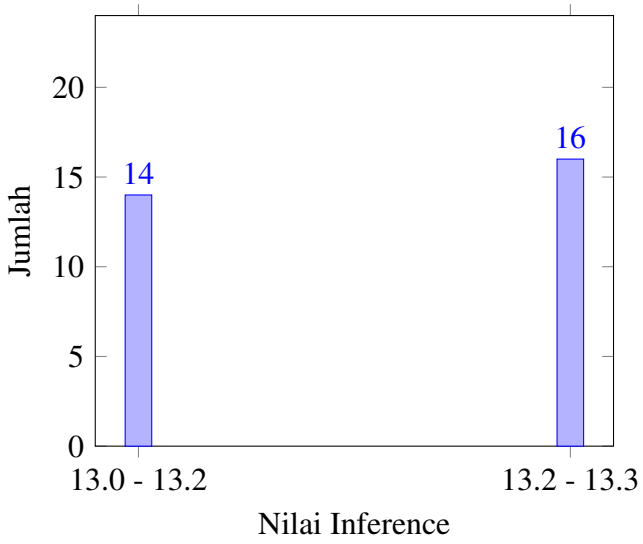
Gambar Histogram 4.16 dan 4.17 merupakan representasi dari nilai pengujian fps yang dikategorikan sesuai dengan jumlah fps yang dihasilkan. Tujuannya digunakan visualisasi ini untuk memudahkan representasi nilai jumlah dari hasil pengujian fps yang didapatkan.

Tabel 4.3: Nilai FPS pada NUC dan Laptop pribadi

No	FPS		No	FPS
1	6.37		1	13.23
2	6.43		2	13.23
3	6.43		3	13.23
4	5.57		4	13.23
5	5.54		5	13.23
...	...		...	...
30	6.33		30	13.05



Gambar 4.16: Histogram Hasil Pengujian *FPS Pada NUC*



Gambar 4.17: Histogram Hasil Pengujian *FPS Pada Laptop*

Kestabilan nilai pada laptop menandakan bahwa performa sistem yang dijalankan pada laptop berjalan dengan sangat baik dan efisien. SedangkanPerforma yang dihasilkan pada NUC tergolong stabil namun hasilnya kurang mendekati performa laptop

4.4 Pengujian Berdasarkan Hasil Response Time

Untuk dapat melakukan pengujian hasil response time, maka obstacle harus memenuhi syarat 1 meter yang telah dibahas pada bab sebelumnya. Output yang dihasilkan setiap kali penghindaran berisi Arah dan time stamp pada Ardruino IDE dan Visual Studio code. Berikut merupakan contoh data mentah untuk kedua hasil output:

```
22:18:32.273 -> Stop
22:18:32.948 -> Arah : E
22:18:33.131 -> Kanan
22:18:33.759 -> Arah : C
22:18:33.936 -> Stop
22:18:34.748 -> Arah : B
22:18:35.121 -> Maju
22:18:35.439 -> Arah : C
22:18:35.815 -> Stop
22:18:36.410 -> Arah : E
22:18:36.616 -> Kanan
```

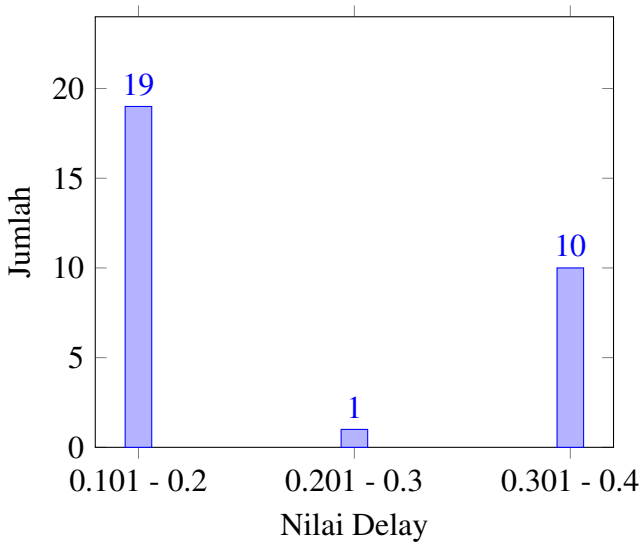
Output yang didapatkan pada ardruino berisikan Timestamp. Timestamp yang dihasilkan yaitu *Timestamp Receive ESP* dan *Timestamp Receive Motor*. Selain pada ardruino pada vscode juga didapatkan nilai time stamp *sent*, contoh outputnya dapat dilihat sebagai berikut.

```
BELOK KANAN,0.8,2024-05-08 22:18:14.263
MAJU,0.4,2024-05-08 22:18:18.762
BELOK KANAN,0.0,2024-05-08 22:18:28.614
MAJU,0.0,2024-05-08 22:18:30.940
BELOK KANAN,1.2,2024-05-08 22:18:32.899
MAJU,0.8,2024-05-08 22:18:34.728
BELOK KANAN,0.0,2024-05-08 22:18:36.392
```

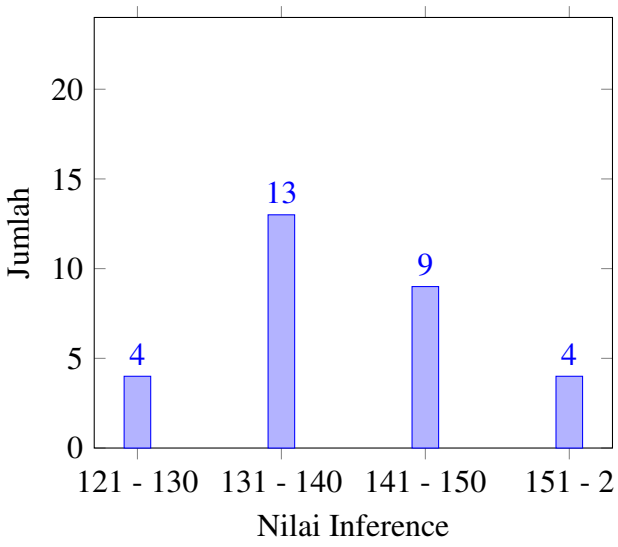
Berdasarkan output diatas dapat dihitung Response Time sistem sejumlah 30 kali yang akan jabarkan pada tabel 4.4 Hasil Response Time akan diuji untuk mendapatkan waktu yang dibutuhkan untuk melakukan pendeteksian dengan model yang kemudian diklasifikasi dan di teruskan ke ESP32 hingga motor kursi roda mulai bergerak. Tabel berisi 5 input pertama, agar data lebih mudah untuk dibaca, telah disertakan histogram yang akan digunakan untuk visualisasi dari data yang didapatkan. 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 0.2494 detik dari data yang sudah didapatkan dari pengujian NUC, adapun hasil nya dapat dilihat pada tabel dibawah. Adapun nilai inference rata-rata yang didapatkan adalah 139.4899 ms atau 0.1394899 detik

Tabel 4.4: 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
...	...	...	...	...	...



Gambar 4.18: Histogram Hasil Pengujian *Delay*



Gambar 4.19: Histogram Hasil Pengujian *Inference*

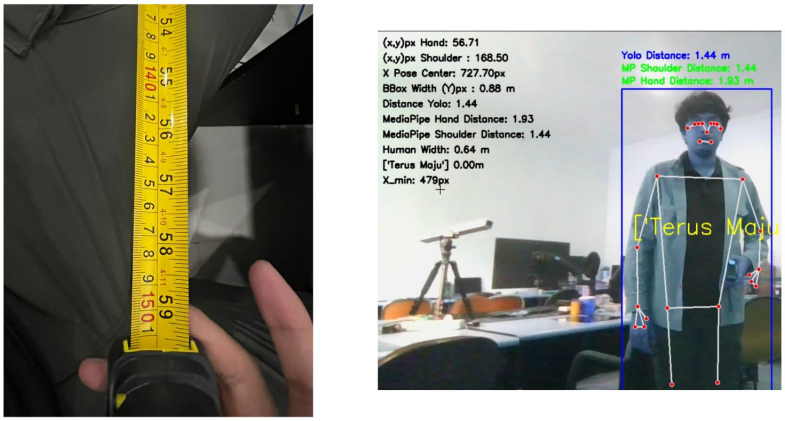
4.5 Pengujian Kesesuaian Jarak Deteksi

Pengujian ini dilakukan pengetasan terhadap model dalam menghasilkan jarak berdasarkan hasil perhitungan pada *Bounding Box* dan pose. Pengujian ini dilakukan dengan membandingkan jarak objek asli dengan jarak yang dihasilkan sistem pada Intel NUC terhadap manusia yang berdiri tegak. Kalibrasi telah dilakukan pada jarak 150cm, jarak ini diambil atas dasar visibilitas pose dan bounding box. Sehingga didapatkan nilai Focal Length sebesar 480, serta nilai K1 dan K2 sebesar 10.922 dan 24.222. Nilai-nilai tersebut akan digunakan dalam pengujian kesesuaian jarak pada 150cm, 100cm dan 50cm. Adapun tujuan dilakukan pengujian ini untuk menguji kemampuan sistem dalam mengukur jarak.

Pengujian ini dilakukan menggunakan alat ukur meteran yang ditancapkan pada kamera dan ditarik menuju peneliti untuk mendapatkan jarak. Nilai tersebut akan dihitung untuk diambil nilai rata rata *difference* atau perbedaan yang dihasilkan dari sistem terhadap pengukuran nyata.

4.5.1 Pengujian Kesesuaian Jarak Deteksi 150cm

Pada pengujian pertama jarak yang digunakan adalah 150cm atau 1.5m pada objek nyata yang dapat dilihat pada gambar 4.20 jarak telah diukur sehingga bisa dilakukan perbandingan dan didapatkan tabel 4.5 untuk jarak menggunakan Yolo *Bounding Box*.



Gambar 4.20: Hasil Pengukuran objek nyata 150cm

Tabel 4.5: Hasil Pengujian kesesuaian Jarak Yolo 150cm

Distance	*Yolo Bbox Distance*	*Difference*
150cm	146cm	4cm
150cm	145cm	5cm
150cm	147cm	3cm
150cm	144cm	6cm
150cm	145cm	5cm
150cm	146cm	4cm
150cm	150cm	0
150cm	150cm	0
150cm	148cm	2cm
150cm	152cm	2cm
150cm	150cm	0
150cm	148cm	2cm
150cm	145cm	5cm
150cm	144cm	6cm
150cm	146cm	4cm

Berdasarkan Tabel 4.5 nilai yolo cenderung stabil dan sesuai pada jarak 150cm. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 3,2 cm. Presentase Error yang dihasilkan yaitu sebesar 2,13 %. Hasil ini menunjukan performa yolo tergolong optimal dalam pendeteksian pada jarak 150cm.

Tabel 4.6: Hasil Pengujian kesesuaian Jarak dengan Landmark Bahu 150cm

Distance	*MediaPipe Shoulder*	*Difference*
150cm	144cm	6cm
150cm	145cm	5cm
150cm	146cm	4cm
150cm	145cm	5cm
150cm	146cm	4cm
150cm	145cm	5cm
150cm	143cm	7cm
150cm	144cm	6cm
150cm	144cm	6cm
150cm	143cm	7cm
150cm	145cm	5cm
150cm	146cm	4cm
150cm	147cm	3cm
150cm	145cm	5cm
150cm	146cm	4cm

Berdasarkan Tabel 4.6 nilai jarak menggunakan landmark Bahu cenderung stabil, namun untuk nilainya kurang sesuai pada jarak 150cm. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 5,06 cm. Presentase Error yang dihasilkan yaitu sebesar 3,37%. Hasil ini menunjukan performa landmark bahu tergolong optimal dalam pendeteksian pada jarak 150cm.

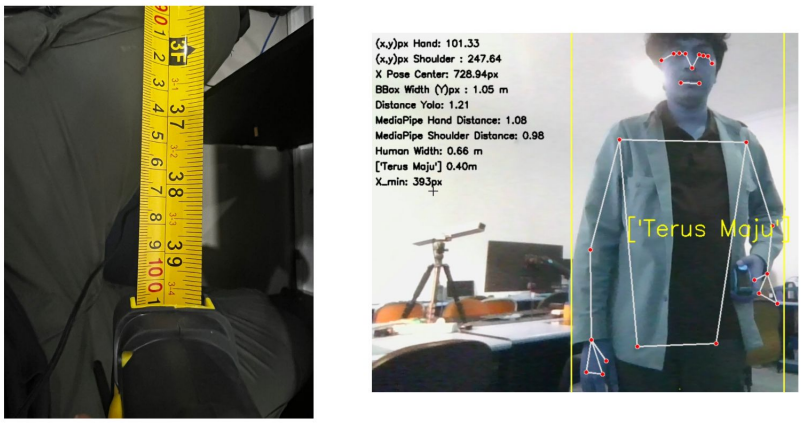
Tabel 4.7: Hasil Pengujian kesesuaian Jarak dengan Landmark Lengan 150cm

Distance	*MediaPipe Hand*	*Difference*
150cm	173cm	23cm
150cm	176cm	26cm
150cm	174cm	24cm
150cm	169cm	19cm
150cm	202cm	52cm
150cm	192cm	42cm
150cm	188cm	48cm
150cm	154cm	4cm
150cm	141cm	9cm
150cm	155cm	5cm
150cm	156cm	6cm
150cm	183cm	33cm
150cm	187cm	37cm
150cm	150cm	0
150cm	180cm	30cm

Berdasarkan Tabel 4.7 nilai jarak menggunakan landmark lengan cenderung tidak stabil dan cenderung fluktuatif terlihat pada nilainya yang sangat tidak sesuai pada jarak 150cm. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 23,8 cm. Presentase Error yang dihasilkan yaitu sebesar 15,86%. Hasil ini menunjukan performa landmark lengan tergolong tidak optimal dalam pendeteksian pada jarak 150cm.

4.5.2 Pengujian Kesesuaian Jarak Deteksi 100cm

Pada pengujian kedua jarak yang digunakan adalah 100cm atau 1m pada objek nyata yang dapat dilihat pada gambar 4.21 dimana jarak telah diukur dan siap dilakukan perbandingan sehingga menghasilkan tabel 4.8.



Gambar 4.21: Hasil Pengukuran objek nyata 100cm

Tabel 4.8: Hasil Pengujian kesesuaian Jarak Yolo 100cm

Distance	*Yolo Bbox Distance*	*Difference*
100cm	121cm	21cm
100cm	121cm	21cm
100cm	121cm	21cm
100cm	120cm	20cm
100cm	121cm	21cm
100cm	121cm	21cm
100cm	120cm	20cm
100cm	121cm	21cm
100cm	121cm	21cm
100cm	121cm	21cm
100cm	121cm	21cm
100cm	121cm	21cm
100cm	121cm	21cm
100cm	121cm	21cm
100cm	121cm	21cm

Berdasarkan Tabel 4.8 nilai yolo cenderung stabil namun memiliki nilai yang tidak sesuai. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 20,8 cm. Presentase Error yang dihasilkan yaitu sebesar 20,8% Ketidak sesuaian nilai ini disebabkan oleh visibilitas bounding box yang terbatas pada jarak 100cm sehingga menyebabkan hasil yang tidak sesuai.

Tabel 4.9: Hasil Pengujian kesesuaian Jarak dengan Landmark Bahu 100cm

Distance	*MediaPipe Shoulder*	*Difference*
100cm	100cm	0
100cm	101cm	1cm
100cm	99cm	1cm
100cm	101cm	1cm
100cm	100cm	0
100cm	97cm	3cm
100cm	98cm	2cm
100cm	96cm	4cm
100cm	98cm	2cm
100cm	99cm	1cm
100cm	96cm	4cm
100cm	95cm	5cm
100cm	96cm	4cm
100cm	98cm	2cm
100cm	102cm	2cm

Berdasarkan Tabel 4.9 nilai jarak menggunakan landmark Bahu cenderung stabil dan menunjukan nilai yang sesuai pada jarak 100cm. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 2,2 cm. Presentase Error yang dihasilkan yaitu sebesar 2,2%. Hasil ini menunjukan performa landmark bahu tergolong optimal dalam pendeteksian pada jarak 100cm.

Tabel 4.10: Hasil Pengujian kesesuaian Jarak dengan Landmark Lengan 100cm

Distance	*MediaPipe Hand*	*Difference*
100cm	103cm	3cm
100cm	104cm	4cm
100cm	105cm	5cm
100cm	104cm	4cm
100cm	101cm	1cm
100cm	103cm	3cm
100cm	104cm	4cm
100cm	105cm	5cm
100cm	106cm	6cm
100cm	104cm	4cm
100cm	106cm	6cm
100cm	103cm	3cm
100cm	102cm	2cm
100cm	101cm	1cm
100cm	102cm	2cm

Berdasarkan Tabel 4.10 nilai jarak menggunakan landmark lengan cenderung stabil dan memiliki nilai yang sesuai dengan jarak nyata. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 3,53 cm. Presentase Error yang dihasilkan yaitu sebesar 3,53%. Hasil ini menunjukan performa landmark lengan tergolong optimal dalam pendeteksian pada jarak 100cm.

4.5.3 Pengujian Kesesuaian Jarak Deteksi 50cm

Pada pengujian kedua jarak yang digunakan adalah 50cm atau 0,5m pada objek nyata yang dapat dilihat pada gambar 4.22 dimana jarak telah diukur dan siap dilakukan perbandingan sehingga menghasilkan tabel 4.11.

Gambar 4.22: Hasil Pengukuran objek nyata 50cm

Tabel 4.11: Hasil Pengujian kesesuaian Jarak Yolo 50cm

Distance	*Yolo Bbox Distance*	*Difference*
50cm	123cm	73cm
50cm	119cm	69cm
50cm	120cm	70cm
50cm	121cm	71cm
50cm	119cm	69cm
50cm	120cm	70cm
50cm	119cm	69cm
50cm	119cm	69cm
50cm	119cm	69cm
50cm	119cm	69cm
50cm	120cm	70cm
50cm	120cm	70cm
50cm	120cm	70cm
50cm	120cm	70cm
50cm	119cm	69cm

Berdasarkan Tabel 4.11 nilai yolo cenderung stabil namun memiliki nilai yang tidak sesuai. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 69,8 cm. Presentase Error yang dihasilkan yaitu sebesar 139%. Ketidak sesuaian nilai ini disebabkan oleh visibilitas bounding box yang terbatas pada jarak 50cm sehingga menyebabkan hasil yang tidak sesuai.

Tabel 4.12: Hasil Pengujian kesesuaian Jarak dengan Landmark Bahu 50cm

Distance	*MediaPipe Shoulder*	*Difference*
50cm	65cm	15cm
50cm	65cm	15cm
50cm	64cm	14cm
50cm	67cm	17cm
50cm	66cm	16cm
50cm	66cm	16cm
50cm	62cm	12cm
50cm	61cm	11cm
50cm	65cm	15cm
50cm	63cm	13cm
50cm	67cm	17cm
50cm	64cm	14cm
50cm	63cm	13cm
50cm	67cm	17cm
50cm	67cm	17cm

Berdasarkan Tabel 4.12 nilai jarak menggunakan landmark Bahu cenderung stabil dan namun menunjukan nilai yang tidak sesuai pada jarak 50cm. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 14,8 cm. Presentase Error yang dihasilkan yaitu sebesar 29,6%. Hasil ini menunjukan performa landmark bahu kurang optimal dalam pendeteksian pada jarak 50cm.

Tabel 4.13: Hasil Pengujian kesesuaian Jarak dengan Landmark Lengan 50cm

Distance	*MediaPipe Hand*	*Difference*
50cm	52cm	2cm
50cm	52cm	2cm
50cm	51cm	1cm
50cm	53cm	3cm
50cm	55cm	5cm
50cm	55cm	5cm
50cm	55cm	5cm
50cm	55cm	5cm
50cm	51cm	1cm
50cm	50cm	0
50cm	50cm	0
50cm	50cm	0
50cm	50cm	0
50cm	50cm	0
50cm	50cm	0

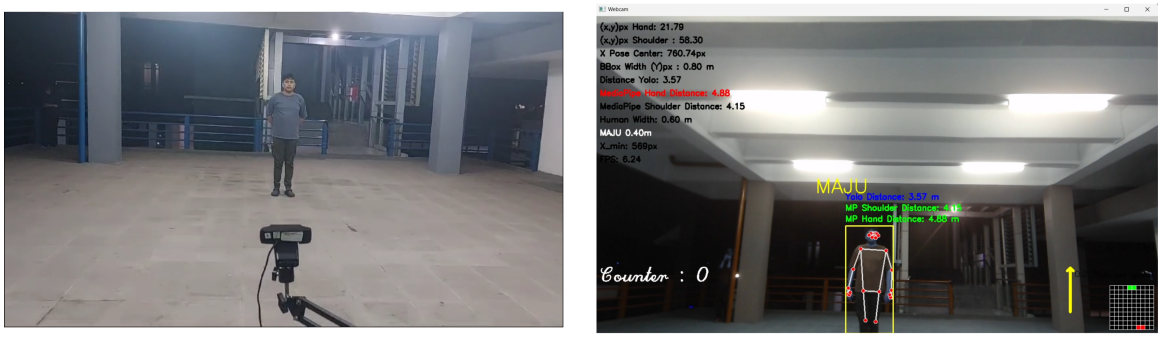
Berdasarkan Tabel 4.13 nilai jarak menggunakan landmark lengan cenderung stabil dan memiliki nilai yang sesuai dengan jarak nyata. Dengan rata - rata nilai *difference* atau selisih nilai sebesar 1,93 cm. Presentase Error yang dihasilkan yaitu sebesar 3,86%. Hasil ini menunjukan performa landmark lengan tergolong optimal dalam pendeteksian pada jarak 50cm.

Distance	Yolo Bbox Error %	MediaPipe Shoulder Error %	MediaPipe Hand Error %
150cm	2,13%	3,37%	15,86%
100cm	20,8%	2,2%	3,53%
50cm	139%	29,6%	3,86%

Tabel 4.5.3 merupakan tabel nilai presentase error keseluruhan. Dimana pada masing - masing jarak nilai error terbesar pada 150cm ialah Landmark lengan dengan presentase 15,86%, dan terkecil pada Yolo Bbox pada 2,13%. Nilai error terbesar pada jarak 100cm adalah Yolo Bbox dengan presentase 20,8% , dan terkecil pada landmark Bahu dengan presentase 2,2%. Nilai error terbesar pada jarak 50cm adalah Yolo Bbox dengan presentase sebesar 139%, dan terkecil landmark lengan dengan presentase 3,86%.

4.6 Performa Keberhasilan Penghindaran

Pengujian ini dilakukan pengetesan terhadap kursi roda dalam menghindari Manusia secara real time. Pengetesan ini akan dilakukan Pada Tower 2 ITS. Manusia yang dideteksi Diam dan tidak bergerak. Gambar 4.23 merupakan contoh foto sampel yang akan digunakan.

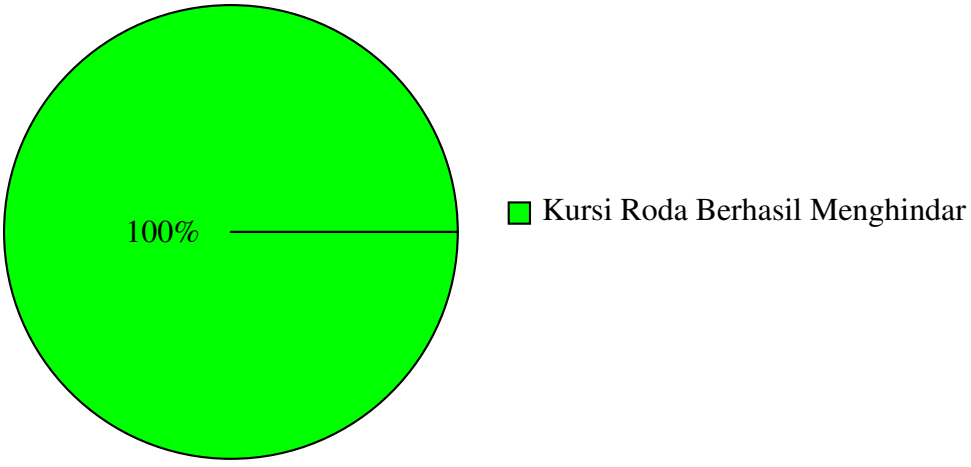


Gambar 4.23: Sampel Pengujian performa keberhasilan penghindaran

Pada pengujian ini diambil 30 kali sampel dan dicatat keberhasilan penghindaran. Setiap hasil akan dicatat dalam tabel yang akan diambil sebuah hasil yaitu Kursi Roda Berhasil Menghindar atau Objek tidak terdeteksi dan Gagal Menghindar. Berikut Tabel hasil yang didapatkan.

Tabel 4.14: Tabel Performa Keberhasilan Penghindaran

Percobaan	Hasil
1	Kursi Roda Berhasil Menghindar
2	Kursi Roda Berhasil Menghindar
3	Kursi Roda Berhasil Menghindar
...	...
30	Kursi Roda Berhasil Menghindar



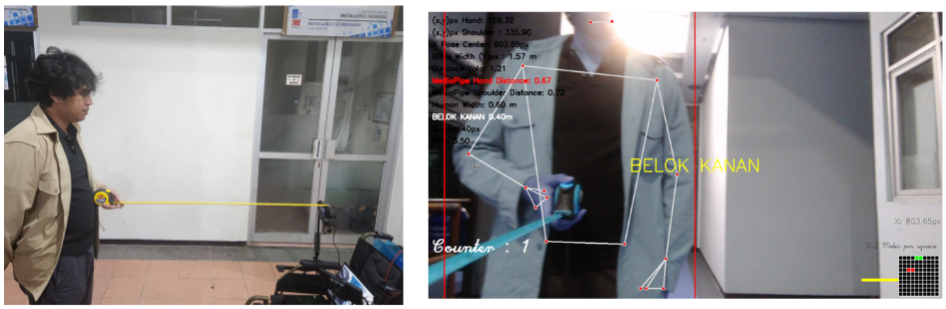
Gambar 4.24: Pie Chart Performa Keberhasilan Penghindaran Kursi Roda

Pada Tabel 4.14 dan Pie Chart 4.24Hasil yang didapatkan Penghindaran berhasil sebanyak 30 kali. Sehingga didapatkan presentase keberhasilan yang didapatkan dari hasil uji ini sebesar 100%. Hasil ini membuktikan sistem yang dibuat mampu untuk mendeteksi dan menghindari manusia dengan baik.

4.7 Performa Akurasi Penghindaran

Pengujian akurasi jarak penghindaran dilakukan dengan mengukur perbandingan antara jarak pengukuran saat kursi roda melakukan penghindaran yang dihasilkan sistem dan jarak kursi roda saat penghindaran terhadap Manusia pada dunia nyata. Jarak yang ditetapkan dalam Penghindaran ialah pada 100 cm atau 1 meter, yang berarti kursi roda harus menghindar pada jarak yang ditentukan apabila ada Manusia yang terdeteksi.

Pada gambar 4.25 telah dilakukan pengukuran menggunakan meteran untuk mendapatkan jarak saat terjadi penghindaran dan juga mengambil jarak pada sistem dengan mendokumentasikan penghindaran menggunakan software OBS lalu dicatat nilai jarak landmark lengan. Penggunaan jarak landmark lengan didasari oleh Pengujian kesesuaian jarak dimana landmark lengan menghasilkan nilai yang paling optimal dijarak yang ditetapkan yaitu 100cm atau 1m.



Gambar 4.25: Hasil Pengukuran Jarak saat penghindaran pada dunia nyata dan pada sistem

Jarak yang telah diukur lalu dibandingkan sehingga menghasilkan Tabel 4.15 yang berisi nilai jarak pengukuran nyata saat penghindaran (Real Distance), Jarak landmark lengan pada sistem yang telah dicatat (Mp Hands), Error nyata, dan error sistem.

Adapun tujuan dilakukannya pengujian ini yaitu untuk mengetahui kesalahan jarak yang dihasilkan dari sistem dan juga mengetahui penyebab kesalahan pengukuran jarak tersebut.

Tabel 4.15: Hasil Performa Akurasi Penghindaran

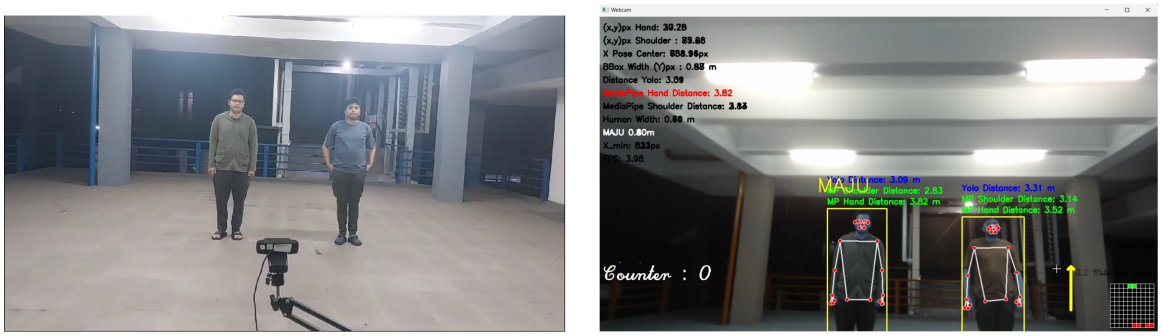
Default Avoidence Distance	Real Distance	Mp Hands	Real Error	Mp Hands Error
100cm	68cm	70cm	32cm	30cm
100cm	75cm	81cm	25cm	19cm
100cm	67cm	71cm	33cm	29cm
100cm	88cm	93cm	12cm	7cm
100cm	79cm	84cm	21cm	16cm
100cm	65cm	67cm	35cm	33cm
100cm	54cm	55cm	46cm	45cm
100cm	59cm	61cm	41cm	39cm
100cm	55cm	58cm	45cm	42cm
100cm	59cm	67cm	41cm	33cm

Dapat dilihat pada Tabel 4.15 didapatkan rata- rata error pengukuran nyata sebesar 33,1 cm dan rata rata error pengukuran sistem sebesar 29,3. Dimana hasil yang didapatkan tidak sesuai dengan jarak yang ditetapkan yaitu 100cm atau 1 meter, dan cenderung mengalami penurunan akurasi.

Penurunan ini disebabkan oleh beberapa faktor, yaitu posisi kamera yang terguncang seiring pengujian, delay pada sistem, laptop yang tidak di charge membuat pemakaian gpu menjadi terbatas dan penurunan performa laptop selama pengujian yang diakibatkan oleh panas laptop yang meningkat seiring dengan waktu pengujian menyebabkan menurunnya FPS yang didapatkan.

4.8 Performa Keberhasilan Penghindaran dengan Dua Obstacle

Pengujian ini dilakukan pengetesan terhadap kursi roda dalam menghindari 2 Manusia sekaligus secara real time. Adapun beberapa tujuan dalam pengujian ini, yang pertama yaitu seberapa baik sistem dalam mendeteksi dua manusia sekaligus, dan seberapa baik model dalam mendeteksi dengan variasi sampel yang berbeda. Pengetesan ini akan dilakukan Pada Tower 2 ITS. Manusia yang dideteksi Diam dan tidak bergerak. berikut merupakan contoh foto sampel yang akan digunakan.



Gambar 4.26: Sampel Pengujian performa keberhasilan penghindaran dua obstacle

Pada tabel 4.16 pengujian ini diambil 10 kali sampel dan dicatat keberhasilan penghindaran. Setiap hasil akan dicatat dalam tabel yang akan diambil sebuah hasil yaitu Kursi Roda

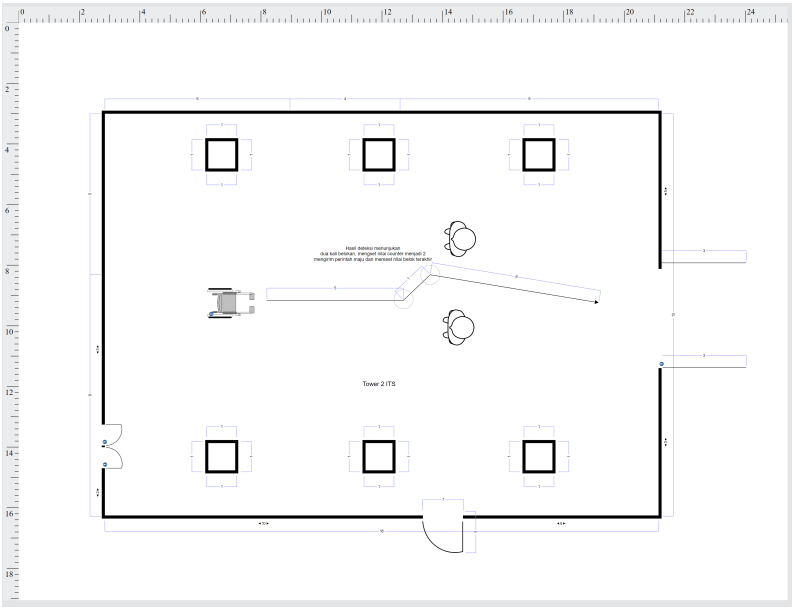
Berhasil Menghindar atau Objek tidak terdeteksi dan Gagal Menghindar. Berikut Tabel hasil yang didapatkan.

Tabel 4.16: Tabel Performa Keberhasilan Penghindaran dua obstacle

Percobaan	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

Berdasarkan Hasil yang didapatkan Penghindaran berhasil sebanyak 10 kali. Sehingga didapatkan presentase yang didapatkan dari hasil uji ini sebesar 100%. Hasil ini membuktikan sistem yang dibuat mampu untuk mendeteksi dan menghindari dua obstacle manusia.

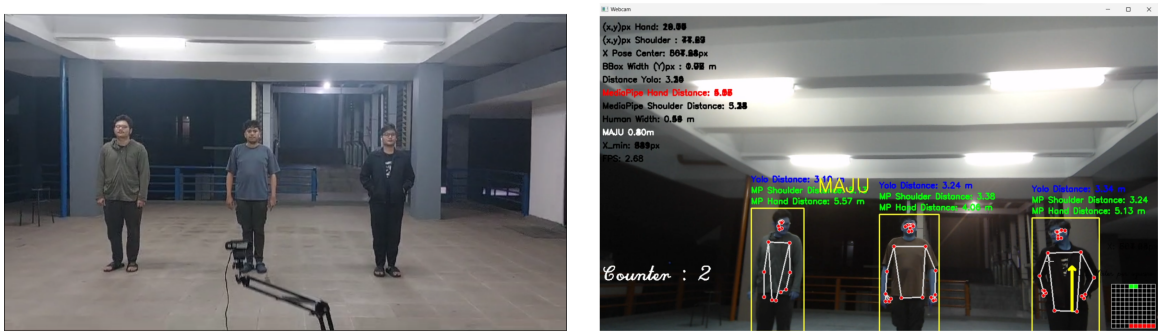
Pada Gambar 4.27 skematik dari salah satu hasil pengujian yang dilakukan, hasil ini menunjukan bahwa kursi roda mampu melakukan penghindaran pada celah yang terdapat diantara obstacle manusia yang terdeteksi.



Gambar 4.27: Skematik penghindaran hasil pengujian dua obstacle

4.9 Performa Keberhasilan Penghindaran dengan Tiga Obstacle

Pengujian ini dilakukan pengetesan terhadap kursi roda dalam menghindari 3 Manusia sekaligus secara real time. Adapun beberapa tujuan dalam pengujian ini, yang pertama yaitu seberapa baik sistem dalam mendeteksi tiga manusia sekaligus, dan seberapa baik model dalam mendeteksi dengan variasi sampel yang berbeda. Pengetesan ini akan dilakukan Pada Tower 2 ITS. Manusia yang dideteksi Diam dan tidak bergerak. Gambar 4.28 merupakan samepl pengujian dan screenshot sistem pada saat pengujian.



Gambar 4.28: Sampel Pengujian performa keberhasilan penghindaran tiga obstacle

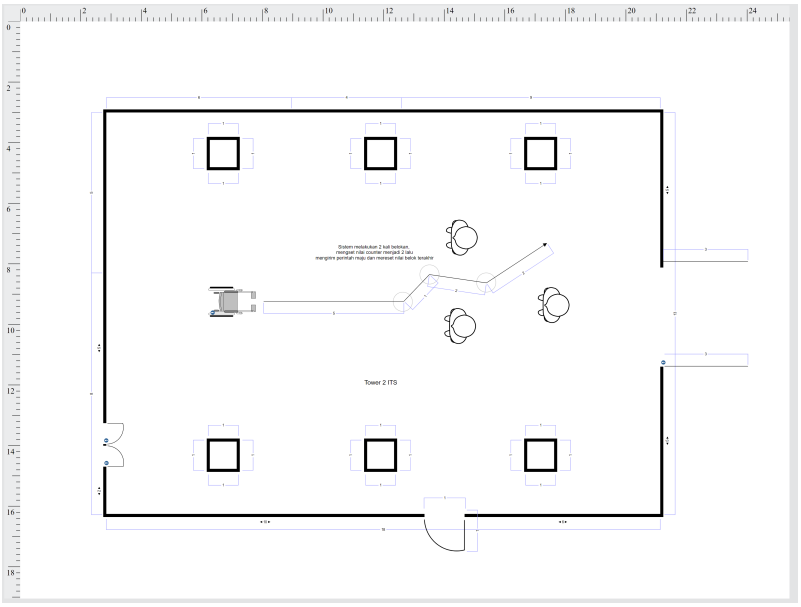
Pada tabel 4.17 pengujian ini diambil 5 kali sampel dan dicatat keberhasilan penghindaran. Setiap hasil akan dicatat dalam tabel yang akan diambil sebuah hasil yaitu Kursi Roda Berhasil Menghindar atau Objek tidak terdeteksi dan Gagal Menghindar. Berikut Tabel hasil yang didapatkan.

Tabel 4.17: Tabel Performa Keberhasilan Penghindaran Tiga obstacle

Percobaan	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

Berdasarkan Hasil yang didapatkan Penghindaran berhasil sebanyak 5 kali. Sehingga didapatkan presentasi yang didapatkan dari hasil uji ini sebesar 100%. Hasil ini membuktikan sistem yang dibuat mampu untuk mendeteksi dan menghindari tiga obstacle manusia.

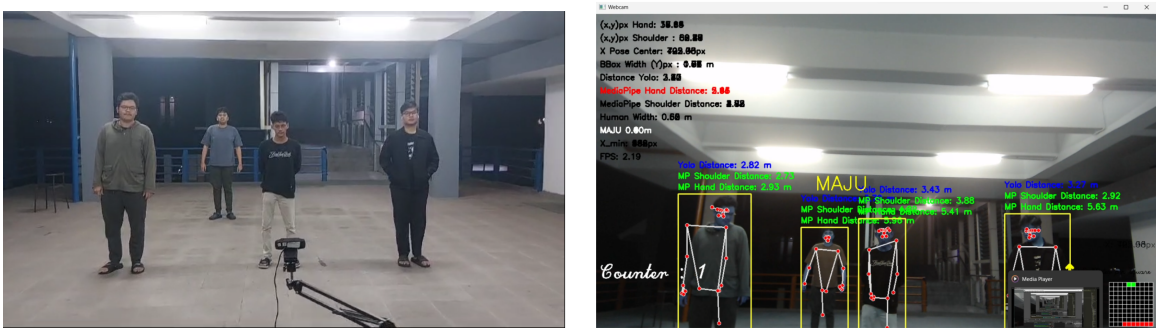
Pada Gambar 4.29 skematik dari salah satu hasil pengujian yang dilakukan, hasil ini menunjukan bahwa kursi roda mampu melakukan penghindaran pada celah yang terdapat diantara obstacle manusia yang terdeteksi.



Gambar 4.29: Skematik penghindaran hasil pengujian empat obstacle

4.10 Performa Keberhasilan Penghindaran dengan Empat Obstacle

Pengujian ini dilakukan pengetesan terhadap kursi roda dalam menghindari 4 Manusia sekaligus secara real time. Adapun beberapa tujuan dalam pengujian ini, yang pertama yaitu seberapa baik sistem dalam mendeteksi empat manusia sekaligus, dan seberapa baik model dalam mendeteksi dengan variasi sampel yang berbeda. Pengetesan ini akan dilakukan Pada Tower 2 ITS. Manusia yang dideteksi Diam dan tidak bergerak. Gambar 4.30 merupakan sampel pengujian dan screenshot sistem pada saat pengujian.



Gambar 4.30: Sampel Pengujian performa keberhasilan penghindaran Empat obstacle

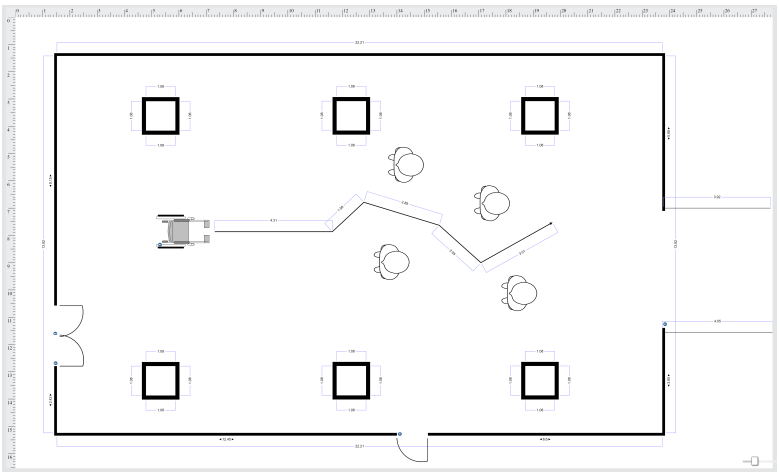
Pada tabel 4.18 pengujian ini diambil 3 kali sampel dan dicatat keberhasilan penghindaran. Setiap hasil akan dicatat dalam tabel yang akan diambil sebuah hasil yaitu Kursi Roda Berhasil Menghindar atau Objek tidak terdeteksi dan Gagal Menghindar. Berikut Tabel hasil yang didapatkan.

Tabel 4.18: Tabel Performa Keberhasilan Penghindaran Empat obstacle

Percobaan	Hasil
1	Kursi Roda Berhasil Menghindar
2	Kursi Roda Berhasil Menghindar
3	Kursi Roda Berhasil Menghindar

Berdasarkan Hasil yang didapatkan Penghindaran berhasil sebanyak 3 kali. Sehingga didapatkan presentasi yang didapatkan dari hasil uji ini sebesar 100%. Hasil ini membuktikan sistem yang dibuat mampu untuk mendeteksi dan menghindari empat obstacle manusia.

Pada Gambar 4.31 skematik dari salah satu hasil pengujian yang dilakukan, hasil ini menunjukan bahwa kursi roda mampu melakukan penghindaran pada celah yang terdapat diantara obstacle manusia yang terdeteksi.



Gambar 4.31: Skematik penghindaran hasil pengujian empat obstacle

4.11 Pembahasan Hasil

Pada bagian ini, akan dibahas hasil dari pengujian yang telah dilakukan terhadap sistem kursi roda otonom berbasis YOLOv8 untuk pendeteksian manusia dan penghindaran rintangan. Pembahasan ini mencakup analisis performa deteksi, akurasi pengukuran jarak, kecepatan pemrosesan, dan keberhasilan penghindaran rintangan.

4.11.1 Performa Deteksi Objek

Berdasarkan hasil pengujian performa model menggunakan confusion matrix, terlihat bahwa model YOLOv8 mampu mendeteksi objek manusia dengan tingkat akurasi yang baik. Pada pelatihan dengan 100 epoch, model memiliki box loss sebesar 0.82978 pada tahap training dan 1.199 pada tahap validasi. Skor mAP (mean Average Precision) pada threshold 0.5 mencapai 81.85%, menunjukkan kemampuan model untuk mendeteksi objek dengan ketepatan yang tinggi.

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

dalam mendeteksi manusia, meskipun masih terdapat beberapa kesalahan deteksi.

4.11.2 Kecepatan Pemrosesan (FPS)

Pengujian kecepatan pemrosesan (FPS) dilakukan pada dua perangkat, yaitu laptop dan NUC. Hasil pengujian menunjukkan bahwa nilai rata-rata FPS pada laptop adalah 13.140, sedangkan pada NUC adalah 6.111. Perbedaan ini menunjukkan bahwa laptop memiliki kemampuan pemrosesan yang lebih baik dibandingkan NUC dalam hal kecepatan deteksi objek.

4.11.3 Response Time

Pengujian response time dilakukan dengan mengukur waktu yang dibutuhkan dari saat pengiriman sinyal hingga motor kursi roda mulai bergerak. Hasil pengujian menunjukkan bahwa rata-rata waktu delay yang didapatkan adalah 0.2494 detik, dengan rata-rata inference time sekitar 139.4899 ms atau 0.1394899 detik. Delay ini dapat dianggap cukup kecil dan masih dalam batas toleransi untuk aplikasi real-time, meskipun dapat ditingkatkan untuk mencapai respons yang lebih cepat.

4.11.4 Kesesuaian Jarak Deteksi

Pengujian kesesuaian jarak deteksi dilakukan pada tiga jarak berbeda, yaitu 150 cm, 100 cm, dan 50 cm. Hasil pengujian menunjukkan bahwa:

- Pada jarak 150 cm, deteksi menggunakan YOLOv8 memiliki rata-rata perbedaan (difference) sebesar 3.2 cm, sedangkan deteksi menggunakan MediaPipe shoulder dan hand memiliki perbedaan masing-masing sebesar 5.06 cm dan 23.8 cm.

- Pada jarak 100 cm, deteksi menggunakan YOLOv8 memiliki perbedaan sebesar 20.8 cm, MediaPipe shoulder sebesar 2.2 cm, dan MediaPipe hand sebesar 3.53 cm.

- Pada jarak 50 cm, deteksi menggunakan YOLOv8 memiliki perbedaan sebesar 69.8 cm, MediaPipe shoulder sebesar 14.8 cm, dan MediaPipe hand sebesar 1.93 cm.

Hasil ini menunjukkan bahwa deteksi menggunakan YOLOv8 dan MediaPipe shoulder lebih akurat pada jarak yang lebih jauh (150 cm dan 100cm), sedangkan MediaPipe hand lebih akurat pada jarak yang lebih dekat (50 cm).

Pada masing - masing jarak nilai error terbesar pada 150cm ialah Landmark lengan dengan presentase 15,86%, dan terkecil pada Yolo Bbox pada 2,13%. Nilai error terbesar pada jarak 100cm adalah Yolo Bbox dengan presentase 20,8% , dan terkecil pada landmark Bahu dengan presentase 2,2%. Nilai error terbesar pada jarak 50cm adalah Yolo Bbox dengan presentase sebesar 139%, dan terkecil landmark lengan dengan presentase 3,86%.

4.11.5 Performa Keberhasilan Penghindaran

Pengujian performa keberhasilan penghindaran dilakukan dengan 30 kali percobaan pada objek manusia yang diam. Hasil pengujian menunjukkan bahwa kursi roda berhasil menghindar sebanyak 30 kali tanpa kegagalan, memberikan tingkat keberhasilan sebesar 100%. Hal ini menunjukkan bahwa sistem kursi roda otonom yang dirancang mampu melakukan penghindaran rintangan dengan sangat baik.

4.11.6 Performa Akurasi Penghindaran

Pengujian performa akurasi penghindaran dilakukan dengan didapatkan rata- rata error pengukuran nyata sebesar 33,1 cm dan rata rata error pengukuran sistem sebesar 29,3cm . Dimana hasil yang didapatkan tidak sesuai dengan jarak yang ditetapkan yaitu 100cm atau 1 meter, dan cenderung mengalami penurunan akurasi.

Penurunan ini disebabkan oleh beberapa faktor, yaitu posisi kamera yang terguncang seiring pengujian, delay pada sistem, laptop yang tidak di charge membuat pemakaian gpu menjadi terbatas dan penurunan performa laptop selama pengujian yang diakibatkan oleh panas laptop yang meningkat seiring dengan waktu pengujian menyebabkan menurunnya FPS yang didapatkan

4.11.7 Performa Penghindaran dengan 2, 3, dan 4 obstacle

Pengujian performa penghindaran dengan lebih dari satu manusia, dilakukan 10 kali percobaan pada kondisi dua obstacle, 5 kali percobaan pada tiga obstacle, dan 3 kali pada empat obstacle. Hasil pengujian menunjukan bahwa baik pada 2, 3 dan 4 obstacle menunjukan presentase keberhasilan sebesar 100%. Dimana hal ini berarti pada masing - masing pengujian kursi roda berhasil menghindar dengan baik tanpa kegagalan. Hal tersebut menunjukan bahwa sistem yang dibuat mampu mendeteksi variasi manusia yang berbeda, dan mampu menghindari lebih dari satu manusia.

[Halaman ini sengaja dikosongkan]

BAB V
PENUTUP

5.1 Kesimpulan

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

1. Model dengan Metrics tertinggi yang telah di-training dengan berbagai konfigurasi adalah model dengan skor mAP di IoU 0.5 tertinggi sebesar 81.85%. Adapun nilai yang didapatkan ini sudah cukup baik untuk melakukan penghindaran dilihat pada hasil performa penghindaran yang sangat baik.
2. Performa NUC dalam pengujian FPS menghasilkan Nilai yang lebih rendah ketimbang Laptop pribadi Penulis dengan selisih 7.029 fps.
3. Rata - rata delay yang didapatkan pada pengujian adalah sekitar 0.2494 detik dan rata-rata nilai inference yang didapatkan 139.4899 ms atau 0.1394 detik.
4. Hasil menunjukkan bahwa deteksi menggunakan *Bounding box* dan Landmark bahu lebih akurat pada jarak yang lebih jauh (150 cm dan 100cm), sedangkan landmark lengan lebih akurat pada jarak yang lebih dekat (50 cm). Dimana rata-rata *difference* terbaik bounding box yaitu pada jarak 150cm sebesar 3.2 cm, rata-rata *difference* landmark bahu terbaik yaitu pada jarak 100cm sebesar 2.2 cm dan rata-rata *difference* landmark lengan terbaik yaitu pada jarak 50cm sebesar 1.93 cm.
5. Hasil Performa Deteksi menunjukan hasil yang memuaskan dalam 30 sampel pengujian. Dengan presentasi keberhasilan sebesar 100% yang menunjukan bahwa sistem yang dibuat dapat menghindari manusia dengan sangat baik.

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 Device yang memiliki performa yang lebih baik untuk fps yang lebih tinggi
3. Meningkatkan performa grid deteksi dengan melakukan penyesuaian yang lebih detail untuk pemetaan yang lebih baik lagi.
4. Menggunakan pendingin device saat melakukan pengujian di ruangan terbuka agar tidak mengalami penurunan performa.
5. Menambah jenis obstacle yang akan dihindari, melakukan penambahan kelas pada model dan penyesuaian pada manuver penghindaran sesuai dengan jenis obstacle yang akan dihindari.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] K. Daring, *Kbbi vi daring*, 2016. [Online]. Available: https://kbbi.kemdikbud.go.id/entri/lumpuh.

[2] P. Pansawira, *Kelumpuhan - gejala, penyebab, dan mengobati - alodokter*, Apr. 2022. [Online]. Available: https://www.alodokter.com/kelumpuhan#:~:text=Kondisi%20ini%20dapat%20disebabkan%20oleh,Penanganan%20kelumpuhan%20tergantung%20pada%20penyebabnya..

[3] J. H. Choi, Y. Chung, and S. Oh, "Motion control of joystick interfaced electric wheelchair for improvement of safety and riding comfort," *Mechatronics*, vol. 59, pp. 104–114, 2019.

[4] L. Lecrosnier, R. Khemmar, N. Ragot, *et al.*, "Deep learning-based object detection, localisation and tracking for smart wheelchair healthcare mobility," *International journal of environmental research and public health*, vol. 18, no. 1, p. 91, 2021.

[5] W. K. Wijaya, I. K. Somawirata, and R. P. M. D. Labib, "Deteksi objek menggunakan yolo v3 untuk keamanan pada pergerakan kursi roda elektrik," *Nucleus Journal*, vol. 1, no. 2, pp. 94–98, 2022.

[6] M. F. Haidar and F. Utaminingrum, "Deteksi plat nama ruangan untuk kendali kursi roda pintar menggunakan yolov5 dan easyocr berbasis tx2," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 7, no. 2, pp. 658–662, Mar. 2023. [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/12272.

[7] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, ISSN: 2504-4990. DOI: 10.3390/make5040083. [Online]. Available: http://dx.doi.org/10.3390/make5040083.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[9] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[10] C. Lugaresi, J. Tang, H. Nash, *et al.*, "Mediapipe: A framework for building perception pipelines," *CoRR*, vol. abs/1906.08172, 2019. arXiv: 1906.08172. [Online]. Available: http://arxiv.org/abs/1906.08172.

[11] J.-W. Kim, J.-Y. Choi, E.-J. Ha, and J.-H. Choi, "Human pose estimation using mediapipe pose and optimization method based on a humanoid model," *Applied Sciences*, vol. 13, no. 4, 2023, ISSN: 2076-3417. [Online]. Available: https://www.mdpi.com/2076-3417/13/4/2700.

[12] D. Shah, *Intersection over union (iou): Definition, calculation, code*, 2023.

[13] I. Parlikar, "Making scientific illustrations with blender," *Biopatrika*, 2022. [Online]. Available: https://biopatrika.com.

[14] L. Neumann and B. Jung, "Blainder—a blender ai add-on for generation of semantically labeled depth-sensing data," *Sensors*, vol. 21, p. 2144, 6 2021. DOI: 10.3390/s21062144. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2144.

[15] C. Moraes, "3d virtual planning for rhinoplasty using a free add-on for open-source software," *Aesthetic Surgery Journal*, 2021. [Online]. Available: https://academic.oup.com/aestheticsurgeryjournal/article/35/13/2323/5210870.

[16] S. E. Wyciślik-Wilson, "Obs studio review," *TechRadar*, 2022. [Online]. Available: https://www.techradar.com/reviews/obs-studio.

[17] YouTube, *The essentials: A beginner's guide to obs (open broadcaster software)*, 2024. [Online]. Available: https://www.youtube.com/watch?v=FwUtwxs-xro.

[18] "Obs studio," *Academic Technologies, University of North Texas*, [Online]. Available: https://academictechnologies.unt.edu/software/obs-studio.

[19] Muhammad, "Analisis driver h-bridge menggunakan relay pada motor bldc konstruksi axial flux (celah ganda)," 2018. [Online]. Available: https://repository.unej.ac.id/bitstream/handle/123456789/89217/Muhammad%20-%20161910201115.pdf?sequence=1&isAllowed=y.

[20] I. Ekatama, "Perancangan sistem kontrol motor kursi roda secara nirkabel berbasis esp32," Ph.D. dissertation, Institut Teknologi Sepuluh Nopember, 2024.

LAMPIRAN

Kode Program

Program 5.1: Program Untuk Penghindaran Manusia

```
from enum import Enum
from ultralytics import YOLO
import cv2
import math
import mediapipe as mp
import numpy as np
import socket
import time
import datetime
import csv
from mediapipe.python.solutions.pose import PoseLandmark


    def delay(seconds):
        start_time = time.time()
        while time.time() - start_time < seconds:
            pass

    csv_file_path = "log.csv"
    csv_columns = ["Direction", "Distance", "Timestamp"]
    host = "192.168.4.1"
    port = 80

    mp_pose = mp.solutions.pose
    pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.6)
    mp_drawing = mp.solutions.drawing_utils

    cap = cv2.VideoCapture(0)
    cap.set(3, 1366)
    cap.set(4, 768)

    k = 10.622
    k2 = 24.222

    focal_length_pixel = 481
    tinggi_objek_nyata = 181

    frame_count = 0
    start_time = time.time()
    fps = 0

    model = YOLO("best100epoch.pt")

    def hitung_jarak(tinggi_bounding_box, focal_length_pixel, tinggi_objek_nyata):
        if tinggi_bounding_box == 0:
```

```
46              return float('inf')
47          jarak = (tinggi_objek_nyata * focal_length_pixel) / ↩
                tinggi_bounding_box
48          return jarak / 100
49
50      def hitung_lebar_objek(lebar_bounding_box, jarak_objek, ↩
            focal_length_pixel):
51          if lebar_bounding_box == 0 or focal_length_pixel == 0:
52              return 0
53          lebar_objek = (lebar_bounding_box * jarak_objek) / ↩
                focal_length_pixel
54          return lebar_objek
55
56      def convert_coordinates(outputs, img_width, img_height):
57          boxes = []
58          for detection in outputs:
59              x_center, y_center, width, height = detection['x_center'], ↩
                    detection['y_center'], detection['width'], detection['↩
                    height']
60
61              x_min = (x_center - width / 2) * img_width
62              y_min = (y_center - height / 2) * img_height
63              x_max = (x_center + width / 2) * img_width
64              y_max = (y_center + height / 2) * img_height
65
66              boxes.append((x_min, y_min, x_max, y_max))
67          return boxes
68
69      def hitung_jarak_euclidean(landmark1, landmark2, lebar_img):
70          jarak_pix = math.sqrt((landmark1.x - landmark2.x) ** 2 + (↩
                landmark1.y - landmark2.y) ** 2) * lebar_img
71          return jarak_pix
72
73      def hitung_lebar_mediapipe(pose_results, lebar_img):
74          if pose_results.pose_landmarks:
75              bahu_kiri = pose_results.pose_landmarks.landmark[mp_pose.↩
                    PoseLandmark.LEFT_SHOULDER]
76              bahu_kanan = pose_results.pose_landmarks.landmark[mp_pose.↩
                    PoseLandmark.RIGHT_SHOULDER]
77              jarak_pix = hitung_jarak_euclidean(bahu_kiri, bahu_kanan, ↩
                    lebar_img)
78
79              faktor_konversi = 0.00087
80              lebar_m = jarak_pix * faktor_konversi
81
82              return lebar_m
83          return 0
84
85      def hitung_jarak_vertikal(pose_results, tinggi_img):
86          if pose_results.pose_landmarks:
87              pusar = pose_results.pose_landmarks.landmark[mp_pose.↩
                    PoseLandmark.NOSE]
88
89              jarak_vertikal = (1 - pusar.y) * tinggi_img
90              return jarak_vertikal
91          return 0
92
```

```
93      def draw_grid(img, detection_status, camera_position):
94          grid_size = 100
95          start_x = img.shape[1] - grid_size - 10
96          start_y = img.shape[0] - grid_size - 10
97
98          cell_size = int(grid_size / 10)
99
100         for i in range(10):
101             for j in range(10):
102                 cell_color = (0, 0, 255) if detection_status[i][j] else ↩
                        (0, 0, 0)
103                 cv2.rectangle(img, (start_x + j * cell_size, start_y + i ↩
                        * cell_size),
104                               (start_x + (j + 1) * cell_size, start_y + (↩
                                  i + 1) * cell_size), cell_color, -1)
105
106         for pos in camera_position:
107             x, y = pos
108             cv2.rectangle(img, (start_x + x * cell_size, start_y + y * ↩
                    cell_size),
109                           (start_x + (x + 1) * cell_size, start_y + (y + ↩
                              1) * cell_size), (0, 255, 0), -1)
110
111         for i in range(11):
112             cv2.line(img, (start_x, start_y + i * cell_size), (start_x + ↩
                    grid_size, start_y + i * cell_size), (255, 255, 255), 1)
113             cv2.line(img, (start_x + i * cell_size, start_y), (start_x + ↩
                    i * cell_size, start_y + grid_size), (255, 255, 255), 1)
114
115         text_x2 = start_x - 10
116         text_x = start_x - 80
117         text_y = start_y - 20
118         text_y2 = start_y - 80
119         cv2.putText(img, "0.2 Meter per square", (text_x,text_y), cv2.↩
                FONT_HERSHEY_SCRIPT_SIMPLEX, 0.6, (0,0,0), 1)
120         cv2.putText(img, f"X: {posisi_horizontal_piksel:.2f}px", (text_x2↩
                , text_y2), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 1)
121
122         return img
123
124     def determine_direction(detection_grid, grid_size=10):
125         if not np.any(detection_grid):
126             return 'Maju', 0, "No detection, moving forward"
127
128         mid_point = grid_size // 2
129         left_count = np.sum(detection_grid[:, :mid_point])
130         right_count = np.sum(detection_grid[:, mid_point:])
131         manusia = "manusia terlalu jauh"
132
133         if left_count > right_count:
134             direction = 'Kanan'
135             manusia = "Manusia di Kiri"
136
137         elif right_count > left_count:
138             direction = 'Kiri'
139             manusia = "Manusia di Kanan"
140
```

```
        else:
            direction = 'Kanan'
            manusia = "Manusia di tengah"

        return direction, abs(right_count - left_count) * 0.2, manusia

    class Direction(Enum):
        MAJU = 1
        BERHASIL_KE_KANAN = 2
        BERHASIL_KE_KIRI = 3

    current_direction = Direction.MAJU
    detection_grid = np.zeros((10, 10), dtype=bool)
    camera_position = [(4, 0), (5, 0)]
    newtex = None
    last_detection_time = time.time()
    counter = 0
    moment_of_truth = False
    while True:
        success, img = cap.read()
        if not success:
            break

        frame_count += 1
        current_time = time.time()

        if current_time - start_time >= 1:
            fps = frame_count / (current_time - start_time)
            frame_count = 0
            start_time = current_time

        detection_grid.fill(False)

        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        start_time_yolo = time.time()

        results = model.predict(img, stream=True)

        end_time_yolo = time.time()

        response_time_yolo = end_time_yolo - start_time_yolo
        print(f"Waktu respons YOLO: {response_time_yolo:.5f} detik")

        with open('log.csv', mode='a') as csv_file:
            writer = csv.DictWriter(csv_file, fieldnames=csv_columns)
            for r in results:
                boxes = r.boxes

                for box in boxes:
                    confidence = math.ceil((box.conf[0] * 100)) / 100
                    if confidence > 0.7:
                        cls = int(box.cls[0])
                        if cls == 0:
                            x1, y1, x2, y2 = box.xyxy[0]
                            x1, y1, x2, y2 = map(int, [x1, y1, x2, y2])
                            color = (255, 0, 0)
```

```
198
199                 tinggi_bounding_box = y2 - y1
200                 lebar_bounding_box = x2 - x1
201                 jarak_yolo = hitung_jarak(tinggi_bounding_box↩
                        , focal_length_pixel, tinggi_objek_nyata)
202
203                 lebar_objek = hitung_lebar_objek(↩
                        lebar_bounding_box, jarak_yolo, ↩
                        focal_length_pixel)
204                 cv2.putText(img, f"X_min: {x1}px", (10, 300),↩
                         cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0),↩
                         2)
205                 cv2.putText(img, f"BBox Width (Y)px : {↩
                        lebar_objek:.2f} m", (10, 120), cv2.↩
                        FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 2)
206                 cv2.putText(img, f"Yolo Distance: {jarak_yolo↩
                        :.2f} m", (x1, y1 - 60), cv2.↩
                        FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 2)
207                 cv2.putText(img, f"Distance Yolo: {jarak_yolo↩
                        :.2f}", (10, 150), cv2.↩
                        FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 2)
208
209                 start_time_mediapipe = time.time()
210                 current_time = time.time()
211
212                 person_img = img[y1:y2, x1:x2]
213                 person_img_rgb = cv2.cvtColor(person_img, cv2↩
                        .COLOR_BGR2RGB)
214                 pose_results = pose.process(person_img_rgb)
215
216                 end_time_mediapipe = time.time()
217                 response_time_mediapipe = end_time_mediapipe ↩
                        - start_time_mediapipe
218                 print(f"Waktu respons MediaPipe: {↩
                        response_time_mediapipe:.5f} detik")
219                 if pose_results.pose_landmarks:
220                     mp_drawing.draw_landmarks(img[y1:y2, x1:↩
                            x2], pose_results.pose_landmarks, ↩
                            mp_pose.POSE_CONNECTIONS)
221
222                     lebar_img = person_img.shape[1]
223                     siku_kanan = pose_results.pose_landmarks.↩
                            landmark[mp_pose.PoseLandmark.↩
                            RIGHT_ELBOW]
224                     pergelangan_kanan = pose_results.↩
                            pose_landmarks.landmark[mp_pose.↩
                            PoseLandmark.RIGHT_WRIST]
225                     bahu_kanan = pose_results.pose_landmarks.↩
                            landmark[mp_pose.PoseLandmark.↩
                            RIGHT_SHOULDER]
226                     bahu_kiri = pose_results.pose_landmarks.↩
                            landmark[mp_pose.PoseLandmark.↩
                            LEFT_SHOULDER]
227                     pusar = pose_results.pose_landmarks.↩
                            landmark[mp_pose.PoseLandmark.NOSE]
228                     jarak_pixbahu = hitung_jarak_euclidean(↩
                            bahu_kanan, bahu_kiri, lebar_img)
```

```
                jarak_pix = hitung_jarak_euclidean(
                    siku_kanan, pergelangan_kanan,
                    lebar_img)
                lebar_img = img.shape[1]
                tinggi_img = img.shape[0]
                lebar_m = hitung_lebar_mediapipe(
                    pose_results, lebar_img)
                grid_size = 10
                jarak_maksimum = 10 * 0.2
                cv2.putText(img, f"Human Width: {lebar_m
                    :.2f} m", (10,240), cv2.
                    FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0),
                    2)

                if jarak_pix > 0:
                    jarak_mediapipe = (k / jarak_pix) *
                        10
                    jarak_mediapipebahu = (k2 /
                        jarak_pixbahu) * 10

                    posisi_horizontal_piksel =
                        pose_results.pose_landmarks.
                        landmark[mp_pose.PoseLandmark.NOSE
                        ].x * lebar_img
                    grid_x = int(((x1 + 350) / lebar_img)
                         * 10)
                    pusar = pose_results.pose_landmarks.
                        landmark[mp_pose.PoseLandmark.NOSE
                        ]
                    posisi_vertikal_piksel = pusar.y *
                        tinggi_img
                    grid_y = int((jarak_maksimum -
                        jarak_mediapipe) / 0.2)
                    grid_y = max(0, min(9 - grid_y, 9))
                    lebar_grid = max(1, int(lebar_m /
                        0.2))
                    grid_x = min(max(grid_x, 0), 9)
                    for i in range(max(0, grid_x -
                        lebar_grid // 2), min(10, grid_x +
                         lebar_grid // 2)):
                         for j in range(max(0, grid_y),
                            min(10, grid_y + lebar_grid //
                             2)):
                             detection_grid[j][i] = True

                    cv2.putText(img, f"Counter : {counter
                        }", (10,600), cv2.
                        FONT_HERSHEY_SCRIPT_COMPLEX, 1.5 ,
                         (255,255,255), 2)
                    cv2.putText(img, f"MP Hand Distance:
                        {jarak_mediapipe:.2f} m", (x1, y1
                        - 10), cv2.FONT_HERSHEY_SIMPLEX,
                        0.6, (0, 255, 0), 2)
                    cv2.putText(img, f"MediaPipe Hand
                        Distance: {jarak_mediapipe:.2f}",
                        (10, 180), cv2.
                        FONT_HERSHEY_SIMPLEX, 0.6, (0, 0,
```

```
                    255), 2)
256                 print(f"Jarak : {jarak_mediapipe:.2f}↩
                        ")
257                 cv2.putText(img, f"MP Shoulder ↩
                        Distance: {jarak_mediapipebahu:.2f↩
                        }", (x1,y1 - 35), cv2.↩
                        FONT_HERSHEY_SIMPLEX, 0.6, ↩
                        (0,255,0), 2)
258                 cv2.putText(img, f"MediaPipe Shoulder↩
                         Distance: {jarak_mediapipebahu:.2↩
                        f}", (10 , 210), cv2.↩
                        FONT_HERSHEY_SIMPLEX,0.6, (0,0,0)↩
                        ,2)
259                 cv2.putText(img, f"FPS: {fps:.2f}", ↩
                        (10, 330), cv2.↩
                        FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, ↩
                        0), 2)
260                 cv2.putText(img, f"(x,y)px Hand: {↩
                        jarak_pix:.2f}", (10, 30), cv2.↩
                        FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, ↩
                        0), 2)
261                 cv2.putText(img, f"(x,y)px Shoulder :↩
                         {jarak_pixbahu:.2f}", (10, 60), ↩
                        cv2.FONT_HERSHEY_SIMPLEX, ↩
                        0.6,(0,0,0),2 )
262                 cv2.putText(img, f"X Pose Center: {↩
                        posisi_horizontal_piksel:.2f}px", ↩
                        (10, 90), cv2.FONT_HERSHEY_SIMPLEX↩
                        , 0.6, (0, 0, 0), 2)
263
264             direction, distance, manusia = ↩
                    determine_direction(detection_grid)
265             if not np.any(detection_grid):
266                 if current_time - last_detection_time > ↩
                        1:
267                     pesan = "MAJU"
268                     print("Fallback: Mengirim perintah ↩
                            Maju")
269                 last_detection_time = current_time
270             else:
271                 if jarak_mediapipe < 1.0 or ↩
                        jarak_mediapipebahu < 1.0 or ↩
                        jarak_yolo < 1.0:
272                     if direction == 'Kiri':
273                         pesan = "BELOK KIRI"
274                         color = (0, 0, 255)
275                         current_direction = Direction.↩
                                BERHASIL_KE_KIRI
276                     elif direction == 'Kanan':
277                         pesan = "BELOK KANAN"
278                         color = (0, 0, 255)
279                         current_direction = Direction.↩
                                BERHASIL_KE_KANAN
280                 else:
281                     pesan = "MAJU"
282                     color = (51, 255, 255)
283                     current_direction = Direction.MAJU
```

```
284
285                         if pesan != newtex:
286                             with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
287                                 s.connect((host, port))
288                                 delay(0.5)
289                                 arah = 'C\n'
290                                 s.send(arah.encode('utf-8'))
291                                 time.sleep(0.5)
292                                 date = datetime.datetime.now()
293                                 print(date)
294                                 agung = ("Stop")
295                                 data = {
296                                     "Direction": agung,
297                                     "Distance": distance,
298                                     "Timestamp": date.strftime("%Y-%m-%d %H:%M:%S")
299                                 }
300                                 writer.writerow({"Direction": agung, "Distance": distance, "Timestamp": date.strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]})
301
302                                 arrow_start_x = img.shape[1] - 200  # Mengatur posisi panah di sebelah kiri grid
303                                 arrow_start_y = img.shape[0] - 50
304                                 if pesan == "BELOK KIRI":
305                                     arah = 'E\n'
306                                     date = datetime.datetime.now()
307                                     agung = ("Kiri")
308                                     start_point = (arrow_start_x, arrow_start_y)
309                                     end_point = (arrow_start_x - 100,arrow_start_y)
310                                     print(date)
311                                     print("kiri")
312                                     counter += 1
313                                 elif pesan == "BELOK KANAN":
314                                     arah = 'A\n'
315                                     date = datetime.datetime.now()
316                                     start_point = (arrow_start_x, arrow_start_y)
317                                     end_point = (arrow_start_x + 100,arrow_start_y)
318                                     agung = ("Kanan")
319                                     print(date)
320                                     print("kanan")
321                                     counter += 1
322                                 else:
323                                     arah = 'B\n'
324                                     date = datetime.datetime.now()
325                                     start_point = (arrow_start_x, arrow_start_y)
326                                     end_point = (arrow_start_x, arrow_start_y - 100)
327                                     agung = ("Maju")
```

```
                                print(date)

                            s.send(arah.encode('utf-8'))
                            newtex = pesan

                            data = {
                                "Direction": agung,
                                "Distance": distance,
                                "Timestamp": date.strftime("%Y-%m-%d %H:%M:%S")
                            }
                            writer.writerow({"Direction": agung, "Distance": distance, "Timestamp": date.strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]})

                    print(f"FPS: {fps:.2f}")
                    cv2.arrowedLine(img, start_point, end_point, (0, 255, 255), 5)
                    direction, distance, manusia = determine_direction(detection_grid)
                    cv2.putText(img, f"{pesan} {distance:.2f}m", (10, 270), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
                    cv2.putText(img, f"{pesan}",(500,400 ), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 255, 255), 2 )
                    cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)
                    print(manusia)
                    img = draw_grid(img, detection_grid, camera_position)


        if not any(boxes) or boxes == []:
            cv2.putText(img, f"Manusia Tidak Terdeteksi", (500,400 ), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 255, 255), 2)
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                s.connect((host, port))
                delay(0.5)
                if counter >= 2:
                    arah = 'B\n'
                    s.send(arah.encode('utf-8'))
                    current_direction = Direction.MAJU
                    counter -= counter
                    moment_of_truth = True

                if current_direction == Direction.MAJU:
                    arah = 'B\n'
                    date = datetime.datetime.now()
                    agung = ("Maju")
                    s.send(arah.encode('utf-8'))
                else :
                    print(date)
                    arah = 'B\n'
```

```
370                         date = datetime.datetime.now()
371                         agung = ("Maju")
372                         print(date)
373                         s.send(arah.encode('utf-8'))
374                         delay(3)
375
376                         arah = 'C\n'
377                         s.send(arah.encode('utf-8'))
378                         delay(1)
379                         if current_direction == Direction.↩
                                BERHASIL_KE_KIRI:
380                             arah = 'A\n'
381                             cv2.putText(img, f"Telah Berhasil ↩
                                    Menghindar, Mengecek Posisi Manusia", ↩
                                    (400,600 ), cv2.FONT_HERSHEY_SIMPLEX, ↩
                                    1.5, (0, 255, 255), 1)
382                             start_point = (arrow_start_x, ↩
                                    arrow_start_y)
383                             end_point = (arrow_start_x - 100, ↩
                                    arrow_start_y)
384                         elif current_direction == Direction.↩
                                BERHASIL_KE_KANAN:
385                             arah = 'E\n'
386                             cv2.putText(img, f"Telah Berhasil ↩
                                    Menghindar, Mengecek Posisi Manusia", ↩
                                    (400,600 ), cv2.FONT_HERSHEY_SIMPLEX, ↩
                                    1.5, (0, 255, 255), 1)
387
388                             start_point = (arrow_start_x, ↩
                                    arrow_start_y)
389                             end_point = (arrow_start_x + 100, ↩
                                    arrow_start_y)
390                         else:
391                             arah = 'B\n'
392                             cv2.putText(img, f"Deteksi Error Perbaiki↩
                                     Kamera", (500,600 ), cv2.↩
                                    FONT_HERSHEY_SIMPLEX, 1.5, (0, 255, ↩
                                    255), 1)
393                             start_point = (arrow_start_x, ↩
                                    arrow_start_y)
394                             end_point = (arrow_start_x, arrow_start_y↩
                                     - 100)
395                             current_direction = Direction.MAJU
396                         s.send(arah.encode('utf-8'))
397                         time.sleep(5)
398
399                         arah = 'C\n'
400                         s.send(arah.encode('utf-8'))
401                         delay(1)
402                         current_direction = Direction.MAJU
403
404
405                         cv2.arrowedLine(img, start_point, end_point, ↩
                                (0, 255, 255), 5)
406                         date = datetime.datetime.now()
407                         print(date)
408                         agung = ("Kembali ke Arah Utama")
```

```
                            print(f"Kembali ke arah: {agung}")

        cv2.imshow('Webcam', img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            counter = 0
        elif cv2.waitKey(1) & 0xFF == ord('a'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

Program 5.2: Program untuk Kalibrasi Focal Length

```
import cv2
import time
import math
from ultralytics import YOLO

# Inisialisasi model YOLOv8
model = YOLO("best100epoch.pt")

# Inisialisasi video capture
cap = cv2.VideoCapture(0)
cap.set(3, 1366)
cap.set(4, 768)

def hitung_tinggi_bounding_box(y1, y2):
    return y2 - y1

while True:
    # Capture frame dari video
    ret, frame = cap.read()
    if not ret:
        break

    # Convert frame ke RGB
    img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Deteksi objek menggunakan YOLO
    results = model.predict(img_rgb, stream=True)

    # Proses hasil deteksi
    for r in results:
        boxes = r.boxes
        for box in boxes:
            confidence = math.ceil((box.conf[0] * 100)) / 100
            if confidence > 0.7:
                cls = int(box.cls[0])
                if cls == 0:  # Jika kelas adalah manusia
                    x1, y1, x2, y2 = map(int, [box.xyxy[0][0], box.xyxy[0][1], box.xyxy[0][2], box.xyxy[0][3]])
                    tinggi_bounding_box = hitung_tinggi_bounding_box(y1, y2)

                    # Gambar bounding box dan tinggi di frame
                    cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)
```

```
                    cv2.putText(frame, f"Tinggi: {tinggi_bounding_box}px", (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 2)

    # Tampilkan frame
    cv2.imshow("Webcam", frame)

    # Tekan 'q' untuk keluar
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release video capture
cap.release()
cv2.destroyAllWindows()
```

Program 5.3: Program untuk Kalibrasi Euclidean Distance

```
import cv2
import math
import mediapipe as mp

# Inisialisasi MediaPipe Pose
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.6)
mp_drawing = mp.solutions.drawing_utils

# Inisialisasi video capture
cap = cv2.VideoCapture(0)
cap.set(3, 1366)
cap.set(4, 768)

def hitung_jarak_euclidean(landmark1, landmark2, lebar_img):
    jarak_pix = math.sqrt((landmark1.x - landmark2.x) ** 2 + (landmark1.y - landmark2.y) ** 2) * lebar_img
    return jarak_pix

while True:
    # Capture frame dari video
    ret, frame = cap.read()
    if not ret:
        break

    # Convert frame ke RGB
    img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Proses pose dengan MediaPipe
    pose_results = pose.process(img_rgb)

    if pose_results.pose_landmarks:
        # Gambar landmark dan koneksi pada frame
        mp_drawing.draw_landmarks(frame, pose_results.pose_landmarks, mp_pose.POSE_CONNECTIONS)

        # Hitung jarak piksel antara landmark siku dan pergelangan tangan kanan
        siku_kanan = pose_results.pose_landmarks.landmark[mp_pose.
```

```
            PoseLandmark.RIGHT_ELBOW]
        pergelangan_kanan = pose_results.pose_landmarks.landmark[mp_pose.↩
            PoseLandmark.RIGHT_WRIST]
        lebar_img = frame.shape[1]
        jarak_pix = hitung_jarak_euclidean(siku_kanan, pergelangan_kanan,↩
             lebar_img)

        # Gambar garis antara siku dan pergelangan tangan kanan
        x1, y1 = int(siku_kanan.x * lebar_img), int(siku_kanan.y * frame.↩
            shape[0])
        x2, y2 = int(pergelangan_kanan.x * lebar_img), int(↩
            pergelangan_kanan.y * frame.shape[0])
        cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

        # Tampilkan jarak piksel pada frame
        cv2.putText(frame, f"Jarak: {jarak_pix:.2f}px", (10, 30), cv2.↩
            FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 2)

    # Tampilkan frame
    cv2.imshow("Webcam", frame)

    # Tekan 'q' untuk keluar
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release video capture
cap.release()
cv2.destroyAllWindows()
```

[Halaman ini sengaja dikosongkan]

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]