PERANCANGAN SISTEM OBSTACLE AVOIDANCE BERBASIS DEEP LEARNING DAN SISTEM NAVIGASI AUTONOMOUS MOBILE ROBOT (AMR)

Hafiz Alifian Aqsha[*)], Hadha Afrisal dan Yosua Alvin A. S.

Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro, Semarang, Indonesia

[*)]*E-mail: hafizalifian@students.undip.ac.id*

Abstrak

Pertumbuhan pesat e-commerce menuntut efisiensi sistem manajemen pergudangan. Perusahaan kini mengadopsi teknologi robotik seperti Autonomous Mobile Robot (AMR) untuk menyederhanakan proses penyimpanan dan pengambilan barang, menggantikan peran forklift yang mahal. Penelitian ini merancang sistem AMR berbasis algoritma A untuk navigasi lokal yang dilengkapi deteksi objek dan obstacle avoidance. Fokus penelitian meliputi deteksi objek statis (kardus) dan dinamis (orang), serta navigasi adaptif berbasis hasil deteksi. Model YOLOv8 digunakan untuk deteksi objek, dengan performa unggul mencapai kecepatan 22 FPS, yang ditingkatkan menjadi 33 FPS menggunakan TensorRT. Sistem ini mampu mendeteksi dan menghindari halangan hingga jarak 300 cm, dengan navigasi AMR yang secara dinamis memperbarui jalur pada jarak aman 100 cm. Hasil menunjukkan integrasi AI pada AMR meningkatkan efisiensi pergudangan otomatis, memberikan solusi praktis untuk tantangan di industri logistik modern.*

Kata kunci: AMR, YOLOv8, TensorRT, obstacle avoidance, navigasi.

Abstract

The rapid growth of e-commerce demands efficiency in warehousing management systems. Companies are now adopting robotic technologies such as Autonomous Mobile Robot (AMR) to simplify the process of storing and retrieving goods, replacing the role of expensive forklifts. This research designs an AMR system based on the A algorithm for local navigation with object detection and obstacle avoidance. The research focuses on static (cardboard) and dynamic (people) object detection, and adaptive navigation based on the detection results. The YOLOv8 model was used for object detection, with superior performance reaching a speed of 22 FPS, which was improved to 33 FPS using TensorRT. The system is able to detect and avoid obstacles up to 300 cm away, with AMR navigation dynamically updating the path at a safe distance of 100 cm. Results show the integration of AI on AMR improves the efficiency of automated warehousing, providing practical solutions to challenges in the modern logistics industry.*

Keywords: AMR, YOLOv8, TensorRT, obstacle avoidance, navigation.

1. Pendahuluan

Sistem Manajemen Pergudangan menjadi hal penting dalam perkembangan industri terutama dalam bisnis E-Commerce. Seiring berjalannya waktu pertumbuhan bisnis online sangat cepat dan semakin banyak. Selain itu permintaan konsumen yang semakin meningkat dapat membuat pengusaha e-commerce tidak dapat memenuhi kebutuhan atau permintaan konsumen tersebut. Sehingga banyak perusahaan e-commerce yang mulai investasi untuk menerapkan sistem manjemen pergudanagan otomatis dengan robot seperti Automated Guided Vehicle (AGV) dan Autonomous Mobile Robot (AMR). Dengan memanfaatkan teknologi mutakhir tersebut diharapkan dapat meningkatkan efisiensi pergerakan produk untuk memenuhi pesanan yang ada seperti penyimpanan, mengurangi pengambilan barang yang salah dan tetap kompetitif di pasar online [1].

Tujuan dari penelitian ini adalah untuk merancang sistem deteksi objek untuk visual *Autonomous Mobile Robot* (AMR), serta mengembangkan sistem obstacle avoidance untuk meningkatkan keamanan dan efisiensi AMR dalam operasionalnya. Selain itu, penelitian ini juga bertujuan untuk merancang sistem navigasi lokal yang terintegrasi dengan algoritma A* pada AMR, guna meningkatkan mobilitas dan efektivitas dalam menentukan jalur pemindahan barang. Dengan demikian, diharapkan AMR

dapat beroperasi secara lebih optimal dalam sistem manajemen pergudangan otomatis.

Navigasi diperlukan untuk lokalisasi dan pemetaan simultan (SLAM), yang mengharuskan robot bergerak cukup cerdas untuk mengumpulkan informasi tentang lingkungan sekitar, membuat peta, dan menggunakan peta tersebut untuk memperkirakan lokasinya. Oleh karena itu, navigasi sangatlah penting [2]. Pengambilan pesanan dengan bantuan AMR menawarkan beberapa manfaat, termasuk rute yang lebih pendek bagi operator, peningkatan kinerja, kemudahan pengoperasian, serta peningkatan keandalan dan kemampuan beradaptasi [3].

Penerapan AMR dalam sistem manajamen pergudangan industri dapat meningkatkan proses produksi ataupun logistik dari perusahaan tersebut. AMR dapat memindahkan barang secara otomatis dari satu titik ke titik lainnya sesuai jalur yang sudah dikenali sebelumnya. Dengan memanfaatkan teknologi tersebut, dapat menggantikan peran forklift yang memiliki harga dan biaya maintanance yang tinggi. Namun dalam kenyataannya sering kali terjadi hal yang tidak terduga seperti terdapat suatu benda atau objek yang menghalangi jalur AMR dalam memindahkan barang sehingga dapat menghambat sistem manajemen pergudangan menjadi tidak efisien.

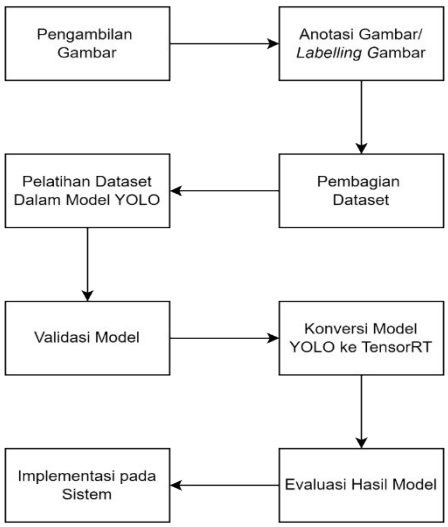
Berdasarkan permasalahan tersebut, diperlukan sistem visual dan navigasi yang baik dari AMR. Dengan menambahkan kamera yang dilengkapi dengan sistem obstacle avoidance berbasis Deep Learning pada AMR diharapkan dapat membantu AMR dalam mendeteksi dan menghindari benda yang menghalangi jalur pemindahan barang. Selain itu perancangan navigasi dapat meningkatkan mobilitas AMR dalam menentukan jalur pemindahan barang.

2. Metode

2.1. Perancangan Sistem Deteksi Objek

Metode untuk merancang sistem deteksi objek digunakan salah satu bagian dari *deep learning*. *Deep Learning* merupakan salah satu metode deteksi objek yang memiliki perkembangan yang sangat pesat dan dapat diandalkan. Hal ini dibuktikan dengan semakin banyaknya teknologi yang memanfaatkan metode deteksi ini untuk meningkatkan performa sistem deteksi yang dibuat. Dalam melakukan tugas untuk mendeteksi objek dengan akurat melalui pengembangan neural network multiple layer disebut juga dengan *Deep Learning* yang merupakan bagian dari kecerdasan buatan dan machine learning [4]. Jaringan saraf tiruan, yang merupakan algoritma yang mereplikasi struktur dan fungsi otak layaknya manusia, adalah subjek *Deep Learning*, bagian dari machine learning [5]. Pada dasarnya, *deep learning* adalah jaringan ANN (*Artificial Neural* Network) dengan tiga lapisan atau lebih. Jaringan ini dapat mengatasi masalah yang sulit dipecahkan oleh algoritma pembelajaran mesin konvensional dan memahami dari jumlah data yang sangat besar [6]. Teknologi ini dimanfaatkan peneliti dalam berbagai bidang seperti di bidang sosial, industri hingga kesehatan.

Dalam merancang sistem deteksi objek diperlukan sebuah *dataset* yang berisikan data latih berupa gambar. Data latih yang digunakan sebagai bahan objek deteksi pada rancangan sistem makalah ini yaitu berupa Manusia (*person*) dan Kardus (*box*). Data-data tersebut dikumpulkan sehingga menjadi 2 kelas data yang nantinya dilakukan proses latih (*training data*) melalui layanan komputasi cloud *Google Colab*. Proses perancangan sistem deteksi objek dapat dilihat pada Gambar 1.



Gambar 1. Proses perancangan sistem deteksi objek.

Berdasarkan Gambar 1. di atas untuk membuat sistem deteksi objek harus melalui beberapa tahapan. Langkah pertama adalah mengumpulkan gambar dan melakukan anotasi setiap gambar pada frame objek yang ingin di deteksi. Kemudian akan menjadi sebuah dataset yang siap untuk di latih ke dalam bentuk model arsitektur YOLO. Proses pelatihan dataset memiliki jangka *epoch* tertentu dan hasil latih terbaik akan digunakan sebagai acuan dalam melakukan deteksi objek pada sistem.

2.1.1. Deteksi Objek dengan YOLOv8

Salah satu dari versi YOLO yang cukup baru yaitu YOLOv8. YOLOv8 merupakan versi yang baru dari model pendeteksian objek dalam kelompok arsitektur YOLO (*You Only Look Once*) yang dikenal dengan kecepatan dan akurasinya yang cukup baik [7]. Kecepatan deteksi yang cepat adalah salah satu fitur yang menonjol dari metode ini. Para penulis menyatakan bahwa YOLO dapat mencapai 45 *frame* per detik, dengan efisiensi yang lebih tinggi dalam versi cepat. Dengan kata lain, dibandingkan dengan sistem *real-time* yang sebanding, 155 *frame* per detik dua kali lipat dari rata-rata presisi rata-rata, atau

mAP (*mean Average Precision*) [8]. YOLOv8 ini dikembangkan oleh Ultralytics. Ada banyak mode yang tersedia pada Ultralytics YOLOv8 yang dapat digunakan untuk berbagai tugas. Beberapa mode ini meliputi pelatihan untuk mengajari model YOLOv8 pada *dataset* khusus, validasi untuk mengecek keefektifan model YOLOv8 yang telah dilatih, prediksi untuk memproses gambar atau video baru dengan menggunakan model YOLOv8 yang sudah terlatih, ekspor untuk mengonversi model YOLOv8 ke format yang bisa digunakan secara lebih luas, pelacakan untuk melacak objek secara *real-time* menggunakan model YOLOv8, serta pengukuran kinerja melalui mode benchmark untuk mengevaluasi kecepatan dan akurasi YOLOv8 [9].

Gambar-gambar yang berkaitan dengan target objek dikumpulkan menjadi sebuah *dataset*. Gambar tersebut harus memiliki beberapa variasi seperti ukuran objek pada gambar, orientasi gambar, dan pencahayaan gambar agar menghasilkan model yang bagus. Dalam penelitian ini telah dikumpulkan gambar sebanyak 6028 gambar. Dari gambar-gambar tersebut akan menjadi sebuah *dataset* yang dibagi menjadi 3 bagian jenis gambar yaitu data latih, data validasi, dan data uji. Pembagian dari 6028 gambar untuk data latih sebesar 70% atau 4219 gambar, data validasi sebesar 20% atau 1206 gambar, dan data uji sebesar 10% atau 603 gambar. Pembagian porsi *dataset* dapat dilihat pada Tabel 1.

Tabel 1. Pembagian jenis data pada *dataset*

Jenis Data	Jumlah
Data Latih	4219 gambar
Data Validasi	1206 gambar
Data Uji	603 gambar

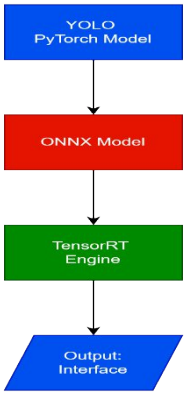
YOLO menggunakan model terintegrasi untuk mendeteksi objek dengan membuat jaringan saraf tunggal memprediksi beberapa *bounding boxes* (kotak pembatas) dan probabilitas kelas di dalamnya secara bersamaan. Sistem ini bekerja dengan cara gambar *input* dibagi ke dalam kisi-kisi S × S melalui sistem YOLO. Salah satu sel *grid* bertugas mendeteksi objek jika pusatnya terletak di dalam sel *grid* tersebut. Kotak pembatas dan *confidence score* (nilai kepercayaan) setiap kotak pembatas diprediksi untuk setiap sel *grid*. Tingkat kepercayaan dan akurasi model dalam mengidentifikasi keberadaan objek di dalam kotak ditunjukkan oleh *confidence score*. Setiap kotak pembatas terdiri dari lima prediksi yaitu koordinat (x, y) yang menunjukkan pusat kotak relatif terhadap batas sel *grid*, dimana (w, h) adalah lebar dan tinggi kotak relatif terhadap gambar. *Confidence* adalah nilai yang menggambarkan seberapa baik kotak prediksi bertepatan dengan kotak *ground truth*, diukur dengan *Intersection over Union* (IoU). Tiap sel *grid* juga memprediksi probabilitas kelas, terkondisi apakah sel *grid* tersebut berisi objek atau tidak, serta hanya satu probabilitas kelas yang diprediksi per sel *grid* tanpa mempertimbangkan jumlah kotak pembatas [10].

Ekstensibilitas YOLOv8 adalah fitur utamanya. Para peneliti yang mengerjakan proyek YOLO akan sangat diuntungkan dengan kemampuan YOLOv8 untuk bekerja dengan dan beralih di antara semua versi YOLO, sehingga memudahkan untuk membandingkan kinerjanya [11].

2.1.2. Deteksi Objek dengan TensorRT

Sebuah ekosistem API untuk inferensi deep learning berkinerja tinggi disediakan oleh NVIDIA TensorRT. Untuk aplikasi komersial, TensorRT menawarkan latensi rendah dan kecepatan tinggi dari hasil optimasi model dan proses inferensi. Ekosistem TensorRT terdiri dari TensorRT-LLM, TensorRT Model Optimizer, dan TensorRT Cloud [12].TensorRT juga tersedia pada Tensorflow dengan nama TF-TRT. Keduanya menggunakan definisi jaringan TensorRT yang sama, namun cara memperoleh grafiknya sangat berbeda. TF-TRT mengimpor dan mengoptimalkan grafik pelatihan Tensorflow (format model tersimpan dalam format frozengraph). Sedangkan TensorRT dari NVIDIA mengubah dan mengoptimalkan format model menjadi format parser ONNX. Kemudian mengubahnya ke dalam bentuk format TensorRT [13]. Kedua metode tersebut memiliki cakupan pengoptimalan yang berbeda sehingga mengakibatkan perbedaan performa. TF-TRT mengoptimasi model pada sebagian grafik. Sedangkan TensorRT mengoptimasi model pada seluruh grafik. Sehingga secara teoritis TensorRT memiliki proses kecepatan yang lebih cepat.

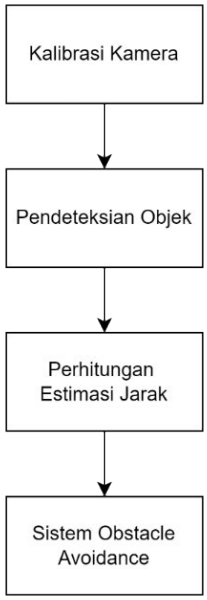
Dengan menggunakan TensorRT dapat mempercepat jaringan saraf dalam menggunakan algoritma yang menyederhanakan arsitektur jaringan tanpa mengubah fungsionalitas jaringan, dan memanfaatkan kekuatan GPU NVIDIA untuk mempercepat komputasi [14]. Pada peneletian ini menggunakan mesin NVIDIA Jetson Orin Nano Dev Kit. Pada mesin tersebut terdapat format atau *framework Deep Learning* yang dapat mengoptimalkan penggunaan *deep learning* seperti model YOLO yang bernama TensorRT. Format tersebut dikembangkan NVIDIA dalam bentuk perangkat lunak mutakhir, dirancang khusus untuk mempercepat proses prediksi *deep learning*. Kemampuannya yang luar biasa menjadikannya pilihan tepat untuk aplikasi *real-time* seperti deteksi objek, di mana kecepatan dan performa menjadi kunci utama. Oleh karena itu, model YOLOv8 di konversikan ke dalam bentuk TensorRT dengan format file.engine. Proses konversi ditunjukkan pada Gambar 2.



Gambar 2. Proses konversi model YOLO menjadi TensorRT

2.2. Perancangan Sistem Obstacle Avoidance

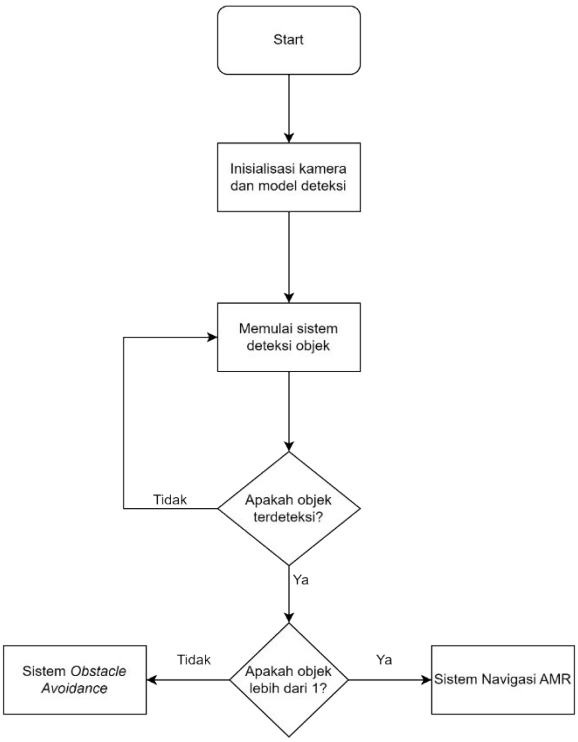
Sistem *Obstacle Avoidance* di rancang dalam rangka meningkatkan mobilitas dan keamanan pada AMR. Sistem ini dirancang menggunakan bahasa pemrograman python. Tujuan dari sistem ini seperti namanya yaitu untuk menghindari halangan pada jalur pemindahan barang. Sistem *Obstacle Avoidance* menggunakan beberapa sistem yaitu deteksi objek, estimasi jarak, dan pengiriman data. Pada tahapan perancangan sistem *obstacle avoidance* terdapat beberapa proses yang perlu dilalui seperti pada Gambar 3.



Gambar 3. Proses perancangan sistem *obstacle avoidance*

2.3. Perancangan Sistem Navigasi

Sistem navigasi lokal di rancang untuk mengatasi kondisi buntu atau *stuck* karena jalur pemindahan rak terhalang. Sistem ini dirancang menggunakan bahasa pemrograman python. Tujuan dari sistem ini seperti namanya yaitu untuk melakukan pencarian jalur navigasi terbaru dengan mengaktifkan A* pada web berdasarkan pembacaan sistem visual atau deteksi objek. Sistem navigasi menggunakan beberapa sistem yaitu deteksi objek, estimasi jarak, pembacaan celah antar objek, dan pengiriman data. Pada tahapan perancangan sistem navigasi terdapat beberapa proses yang perlu dilalui seperti pada Gamabar 4.



Gambar 4. Proses perancangan sistem navigasi

Pada rancangan sistem navigasi AMR digunakan algoritma A* (A-star) sebagai *path finding*. Algoritma ini memiliki cara kerja dengan menggabungkan pencarian jalur terpendek (seperti Dijkstra) dengan heuristik (seperti Breadth-First Search) untuk menemukan jalur yang paling efisien dari titik awal ke titik tujuan. Tidak seperti algoritma Djikstra, algoritma A-Star memindahkan sebuah node dari daftar terbuka ke daftar tertutup dengan memperkenalkan potensi node, yang merupakan perkiraan jarak yang harus ditempuh untuk mencapai node tujuan. Alasan utama dari peningkatan efisiensi algoritma A-Star adalah karena potensi simpul mengindikasikan orientasi simpul tujuan. Fondasi sistem navigasi kendaraan, yang telah digunakan oleh manusia, robot, kendaraan jalan raya, kendaraan udara tak berawak, dan lain-lain, adalah algoritma A-star [15]. Pada sistem navigasi ini algoritma tersebut akan bekerja ketika terdapat 2 halangan atau lebih yang menghalangi jalur pemindahan rak. Dalam algoritma ini terdapat pembacaan koordinat posisi halangan berdasarkan titik tengah objek pada *frame* dalam piksel. Setelah mendapatkan koordinat x dari masing-masing objek akan diklasifikasikan sesuai dengan posisi objek. Terdapat tiga kondisi dalam menentukan jalur navigasi lokal. Pertama kondisi saat objek berada di tengah dan kanan, kondisi kedua saat objek berada di tengah dan kiri, dan terakhir kondisi saat kedua objek berada di tengah.

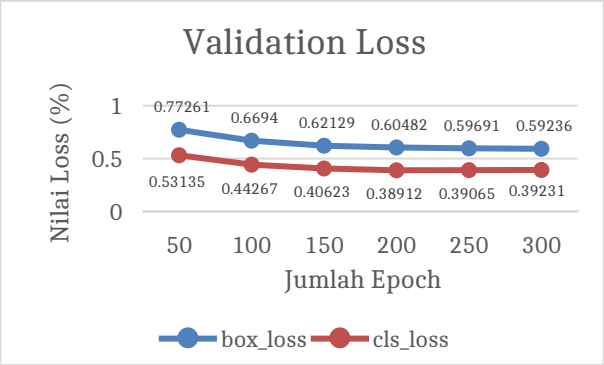
3. Hasil dan Pembahasan

3.1. Pengujian Sistem Visual AMR

Pengujian ini dilakukan untuk mengetahui bahwa sistem visual robot dapat bekerja dengan baik sesuai model yang dibuat. Pengujian ini juga dilakukan untuk mengetahui kemampuan deteksi objek beserta nilai *loss* (*error*) yang dihasilkan melalui model YOLOv8 dan TensorRT. Selain itu terdapat juga pengujian algoritma estimasi jarak objek terhadap kamera untuk mengetahui nilai error yang dihasilkan pada jarak tertentu.

3.1.1. Pengujian Deteksi Objek dengan YOLOv8

Pengujian sistem deteksi objek dengan model YOLOv8 dilakukan dengan menggunakan perintah predict dalam library YOLO setelah proses latih data selesai. Terdapat 3 parameter yang digunakan dalam pengujian ini yaitu, nilai loss, tingkat keberhasilan prediksi, dan nilai FPS secara real-time. Pengujian nilai loss dilakukan dengan membandingkan nilai error tiap epoch latih data. Grafik nilai loss model ditunjukan pada Gambar 5.



Gambar 5. Grafik nilai loss model YOLOv8

Berdasarkan Gambar 5, diketahui bahwa latih data dilakukan sebanyak 300 epoch dengan nilai loss selalu mengalami penurunan seiring bertambahnya jumlah epoch. Terdapat 2 nilai loss dalam latih data yaitu box loss dan class loss. Masing-masing nilai loss ini memiliki arti sendiri, box loss membantu model dalam mempelajari posisi dan ukuran kotak pembatas (bounding box) di sekitar objek yang terdeteksi serta meminimalisir kesalahan antara kotak prediksi dengan kotak aslinya (ground truth). Sedangkan class loss berkaitan dengan akurasi klasifikasi objek. Semakin kecil nilai loss, maka semakin tinggi akurasi model dalam mendeteksi objek.
Dari gambar tersebut didapati bahwa nilai box loss dan classes loss terndah berada pada epoch 300 dengan nilai 0.59 dan 0.39. Berdasarkan dengan konsep tersebut didapati bahwa kedua nilai loss terkecil berada saat epoch mencapai 300 kali iterasi. Sehingga model terbaik yang akan digunakan yaitu model dengan proses latih data sebanyak 300 epoch.

Tingkat keberhasilan model ditunjukkan dengan menghitung nilai Akurasi, Presisi, Recall, dan F1-Score di tiap kelasnya. Untuk mengetahui keakuratan model dalam mendeteksi dapat dilihat melalui tingkat akurasi. Selain akurat, model juga harus tepat dalam mendeteksi objek. Ketepatan pendeteksian ditunjukkan melalui presentase presisi. Rasio kebenaran model dalam memprediksi kelas sesuai dengan kelas aslinya dapat dilihat melalui persentase recall. Sedangkan F1-Score merupakan nilai rata-rata harmonis dari presisi dan recall. Nilai F1-Score mengindikasikan tingkat keberhasilan model dalam memprediksi. Semakin besar nilai F1-Score maka semakin baik model dalam mendeteksi objek. Hasil perhitungan tingkat keberhasilan model dapat dilihat pada Tabel 2.

Tabel 2. Tingkat keberhasilan model berdasarkan confusion matrix

Class	Akurasi (%)	Presisi (%)	Recall (%)	F1-Score (%)
Box	96%	94,1%	97,8%	95,8%
Person	88,9%	86,7%	87,7%	87%

Pada tabel 2 dapat diketahui nilai akurasi, presisi, recall, dan F1-score dari masing-masing kelas. Akurasi yang didapatkan pada kelas box dan person masing-masing sebesar 96% dan 88,9%. Tingkat presisi yang didapat sebesar 94,1% pada kelas box dan 86,7% pada kelas person. Selain itu recall yang dihasilkan dari kelas box sebesar 97,8% dan kelas person sebesar 87,7%. Terakhir terdapat nilai F1-score dari kelas box sebesar 95,8% dan 87% pada kelas person. Berdasarkan konsep tingkat keberhasilan model dapat disimpulkan bahwa kelas box memiliki tingkat keberhasilan prediksi lebih besar daripada kelas person. Tingkat keberhasilan model juga dihitung saat iterasi proses latih data. Nilai-nilai tersebut dapat dilihat melalui Tabel 3.

Tabel 3. Tingkat keberhasilan model tiap epoch

Epoch	Recall (%)	Presisi (%)	Mean Average Precision (mAP @50) (%)	Mean Average Precision (mAP @50-95) (%)
50	88%	87%	93%	73%
100	89%	91%	95%	78%
150	91%	91%	95%	80%
200	91%	92%	95%	81%
250	91%	93%	95%	82%
300	91%	93%	95%	82%

Berdasarkan hasil tingkat keberhasilan model dari tabel 4.3, dapat diketahui bahwa nilai recall, presisi, dan mAP selalu meningkat seiring bertambahnya epoch atau iterasi latih data. Hal ini menandakan bahwa semakin lama data di latih, maka semakin tinggi tingkat keberhasilan model dalam memprediksi kelas yang dibuat. mean Average Precision (mAP) merupakan parameter yang mengindikasikan keberhasilan model dalam mendeteksi objek sesuai kelas yang ada. Terdapat 2 jenis mAP yang digunakan pada latih data. Pertama mAP @50, mengukur rata-rata presisi dalam mendeteksi objek dengan ambang batas Intersection over Union (IoU) sebesar 0.5. Sedangkan mAP @50-95 mengukur presisi rata-rata dalam mendeteksi objek dengan ambang batas IoU yang berkisar antara 0,5 hingga 0,95. Nilai ambang batas menentukan model dalam mendeteksi objek. Jika nilai kepercayaan (Confidence Score) pada objek yang dideteksi lebih kecil daripada nilai ambang batas, maka model tidak akan menganggap objek tersebut. Sehingga semakin tinggi nilai ambang batas, semakin kecil tingkat keberhasilan model dalam memprediksi kelas atau mendeteksi objek.

Pengujian FPS pada model YOLO dilakukan secara real-time dengan mendeteksi objek melalui kamera. Nilai FPS bergantung pada mesin yang digunakan karena model YOLO memiliki komputasi yang berat dan cepat sehingga dalam memanfaatkan computer vision diperlukan mesin yang canggih dalam menjalankan komputasi AI. Hasil dari komputasi deteksi objek model YOLO pada NVIDIA Jetson Orin Nano dapat dilihat pada Tabel 4.

Tabel 4. Hasil kecepatan deteksi model YOLOv8

Class	Inference Time Means (ms)	FPS	mAP 50(B)	Size (pixels)
Box	45	22	0,95	640x640
Person	45	22	0,93	640x640

Berdasarkan hasil pengujian yang didapat pada tabel 4.4, nilai mAP dengan ambang batas IoU sebesar 0,5 yaitu 0,95 di kelas box dan 0,93 di kelas person. Kedua kelas memiliki kecepatan deteksi yang sama yaitu 45 ms atau sebesar 22 FPS (Frame Per Second). Hasil evaluasi ini dilakukan dengan metode predict atau prediksi menggunakan model YOLOv8 yang sudah dilatih dengan ukuran gambar sebesar 640x640. Dengan demikian dapat disimpulkan bahwa nilai kepercayaan model dalam mendeteksi objek sesuai kelasnya cukup tinggi namun tidak dengan kecepatan deteksi.

3.1.2. Pengujian Deteksi Objek dengan TensorRT

Pengujian sistem deteksi objek dengan TensorRT dilakukan dengan cara seperti pada model YOLOv8, namun pengujian pada sistem deteksi ini hanya dilakukan untuk membandingkan nilai FPS, nilai kepercayaan, dan nilai rata-rata kecepatan deteksi. Hasil dari pengujian tersebut dapat dilihat pada tabel 4.5 berikut.

Tabel 5. Hasil pengujian kecepatan deteksi model TensorRT

Class	Inference Time Mean (ms)	FPS	mAP 50(B)	Size (pixels)
Box	30	33	0,94	640x640
Person	30	33	0,90	640x640

Berdasarkan tabel 5 dapat diketahui bahwa nilai FPS yang dihasilkan sebesar 33 FPS dan lebih besar dibandingkan model YOLOv8. Selain itu kecepatan rata-rata deteksi objek mencapai 30 ms, sehingga model TensorRT dapat dikatakan lebih cepat daripada model YOLOv8 berdasarkan hasil pengujian yang telah dilakukan.

3.2. Pengujian Sistem Obstacle Avoidance

Pengujian pada sistem obstacle avoidance dilakukan untuk mengetahui bahwa robot dapat menghindari halangan berdasarkan objek yang dideteksi oleh sistem visual AMR. Pengujian ini juga dilakukan untuk mengetahui kemampuan jangkauan deteksi sistem dan nilai kepercayaan model berdasarkan jarak deteksi. Selain itu terdapat juga pengujian algoritma berdasarkan kondisi objek untuk mengetahui keberhasilan pengiriman data secara real-time ke mikrokontroler sesuai dengan kondisi objek.

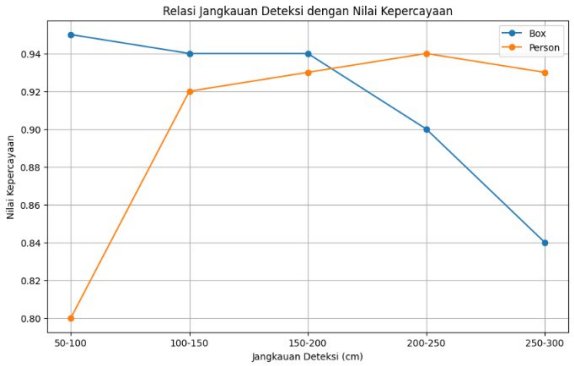
3.2.1. Pengujian Jangkauan Deteksi

Pengujian ini dilakukan dengan mengukur jangkauan deteksi objek terhadap kamera dan hubungan nilai kepercayaan berdasarkan jarak deteksi. Berikut merupakan hasil nilai kepercayaan model dalam jarak tertentu dapat dilihat pada Tabel 6.

Tabel 6. Nilai kepercayaan model terhadap jarak objek

Class	Jangkauan Deteksi (cm)	Nilai Kepercayaan (0-1)
Box	50-100	0,95
	100-150	0,94
	150-200	0,94
	200-250	0,90
	250-300	0,84
Person	50-100	0,80
	100-150	0,92
	150-200	0,93
	200-250	0,94
	250-300	0,93

Berdasarkan hasil pengujian nilai kepercayaan model mendeteksi objek dalam jarak tertentu pada tabel 6, dapat diketahui bahwa nilai kepercayaan berubah-ubah seiringnya jauhnya objek dari kamera. Pengujian dilakukan pada masing-masing kelas dengan jarak jangkauan deteksi 50 cm hingga 300 cm. Model dapat bekerja secara optimal di rentang 150 hingga 200 cm pada kedua kelas. Dengan demikian, dapat disimpukan bahwa semakin besar jarak antara objek dengan kamera maka nilai kepercayaan akan semakin kecil pada kelas box. Namun hal tersebut berbanding terbalik pada kelas person yang disebabkan ruang pandang (Field of View) kamera yang terbatas, sehingga kamera tidak dapat menangkap manusia secara keseluruhan ke dalam frame pada jarak cukup dekat. Berikut perbandingan nilai kepercayaan terhadap jangkauan deteksi di tiap kelas dapat dilihat pada Gambar 6.



Gambar 6. Perbandingan nilai kepercayaan model antar kelas terhadap jarak

3.2.2. Pengujian Algoritma Berdasarkan Kondisi Objek

Pengujian berikutnya yaitu menguji performa pembacaan sistem berdasarkan posisi halangan dalam jarak 75 cm. Dalam pengujian ini disertakan juga kecepatan model dalam mendeteksi dan nilai kepercayaan. Hasil dari pengujian tersebut dapat dilihat pada Tabel 7.

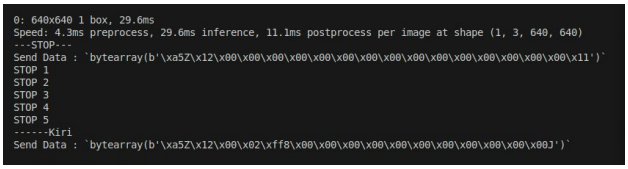
Tabel 7. Hasil pengujian algoritma berdasarkan kondisi objek

Kondisi	Posisi Objek	Koordinat Objek x (px)	Rata-Rata Waktu Pembacaan (ms)	Nilai Kepercayaan (0-1)	Keterangan
Statis	Kanan	[562]	29,6	0,86	Berhasil
		[458]	28,3	0,88	Berhasil
		[382]	29,2	0,92	Berhasil
	Kiri	[073]	29,3	0,84	Berhasil
		[148]	30,3	0,92	Berhasil
		[289]	39,0	0,94	Berhasil
Dinamis	Kanan-Kiri	-	40,7	0,86	Berhasil

Berdasarkan Tabel 7, dapat diketahui bahwa pengujian algoritma obstacle avoidance dilakukan dengan beberapa kondisi. Kondisi pertama ketika halangan bersifat statis (diam) dan posisi objek berada di sebelah kanan. Pada pengujian tersebut didapatkan rata-rata waktu pembacaan sebesar 43,4 ms dengan nilai kepercayaan sebesar 0,88. Algoritma obstacle avoidance pada kondisi ini berhasil menghindari halangan yang ada pada jarak 75 cm di sebelah kanan robot. Hasil dari algoritma dengan kondisi pertama dapat dilihat pada Gambar 7 berikut.

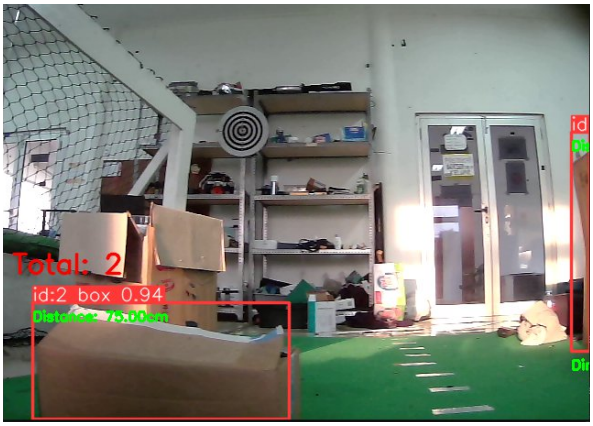
Gambar 7. Hasil pendeteksian kondisi objek statis kanan

Setelah model berhasil mendeteksi dan memprediksi jarak beserta arah objek, program akan mengirimkan perintah STOP selama 5 detik dan mengirim perintah direction sesuai pembacaan sistem. Jika objek berada di kanan, maka program akan mengirim perintah untuk bergerak ke kiri. Perintah-perintah tersebut dikirimkan ke mikrokontroler dalam bentuk bytearray. Data bytearray dikirimkan dengan fungsi data_parse.send_command yang berisikan ID serial, perintah gerak maju atau mundur yang ditandai dengan nilai positif untuk maju dan negatif untuk mundur serta berlaku juga pada perintah gerak kanan atau kiri. Hasil pengiriman data berdasarkan kondisi pertama dapat dilihat pada Gambar 8.



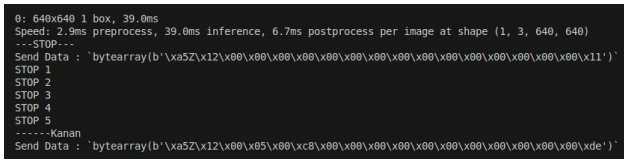
Gambar 8. Hasil pengiriman data berdasarkan kondisi pertama

Kondisi kedua adalah ketika objek bersifat statis dan posisi objek di sebelah kiri robot. Pada pengujian tersebut didapatkan rata-rata waktu pembacaan sebesar 42.3 ms dengan nilai kepercayaan model dalam mendeteksi sebesar 0.94. Hasil dari algoritma dalam kondisi kedua dapat dilihat pada Gambar 9.



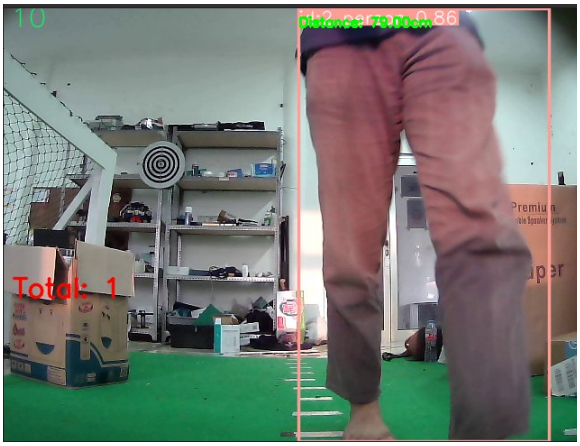
Gambar 9. Hasil pendeteksian kondisi objek statis kiri

Berdasarkan hasil pembacaan sistem obstacle avoidance pada kondisi kedua seperti gambar 4.5 dapat diketahui bahwa sistem berhasil mendeteksi objek yang berada di kiri robot dan mengirimkan perintah ke mikrokontroler. Hasil pengiriman data pada kondisi kedua dapat dilihat melalui Gambar 10.



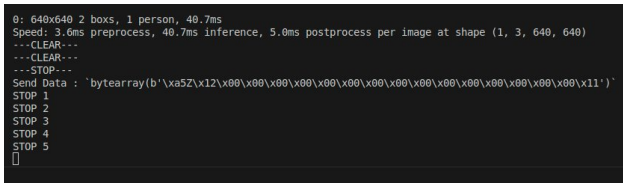
Gambar 10. Hasil pengiriman data berdasarkan kondisi kedua

Kondisi ketiga adalah kondisi saat objek bergerak dinamis. Pada pengujian ini manusia sebagai objek yang bergerak menuju robot saat robot berjalan. Hasil dari pengujian tersebut didapatkan bahwa algoritma tersebut berhasil mendeteksi dan melakukan perkiraan jarak pada objek dinamis. Hasil algoritma obstacle avoidance dalam kondisi ketiga dapat dilihat pada Gambar 11.



Gambar 11. Hasil pendeteksian kondisi objek dinamis

Pada Gambar 11 dapat dilihat bahwa sistem dapat bekerja sesuai algoritma yang dirancang. Hasil pengiriman data dapat dilihat pada Gambar 12. berikut.

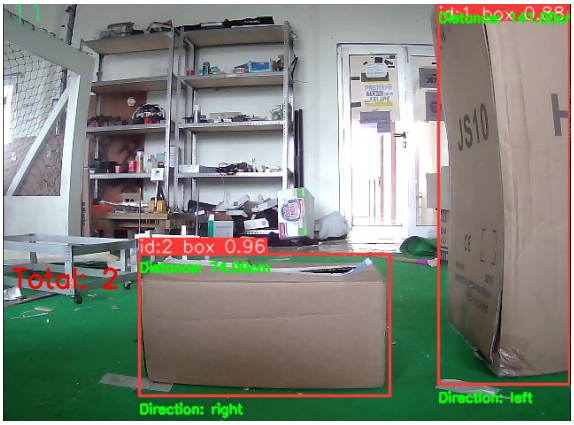


Gambar 12. Hasil pengiriman data berdasarkan kondisi ketiga

3.3. Pengujian Sistem Navigasi

Pada pengujian ini dilakukan untuk mengetahui bahwa sistem navigasi lokal robot dapat bekerja dengan optimal. Pengujian ini juga dilakukan untuk mengetahui algoritma navigasi lokal mampu memperbarui status kondisi jalur dan membuat jalur navigasi baru sesuai kondisi terbaru. Selain itu terdapat juga pengujian pengiriman data ke mikrokontroler untuk memberikan perintah atau status robot berdasarkan sistem visual secara *real-time*.

Dalam menguji performa algoritma navigasi lokal pada robot dilakukan dengan beberapa kondisi letak halangan. Pada pengujian dilakukan menggunakan kamera dan pembacaan letak halangan berdasarkan koordinat *x* dari *frame* dalam satuan pixel. Setelah koordinat halangan terbaca akan dikirimkan ke *web app* untuk memperbarui status jalur navigasi. Pengiriman data dilakukan menggunakan mqtt dengan tipe data *string* yang berisikan *header*, *set block/unblock*, jumlah *block*, *use_astar/no_astar*, posisi halangan. Hasil pendeteksian objek berdasarkan kondisi halangan pertama dapat dilihat pada Gambar 13.



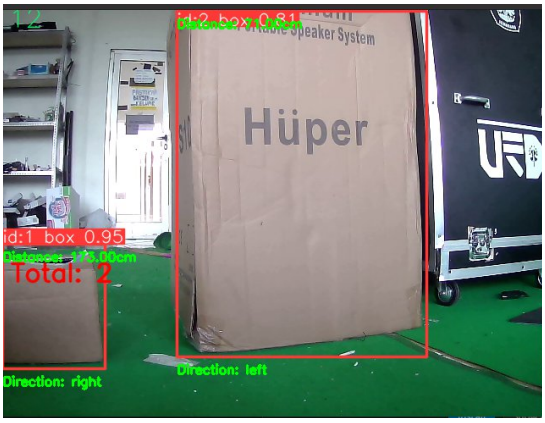
Gambar 13. Hasil pendeteksian objek dengan kondisi halangan pertama

Koordinat *x* titik tengah dari objek yang terbaca adalah 322 pada halangan tengah dan 530 pada halangan kanan. Ketika halangan berjarak 100 cm terhadap robot, program akan mengirimkan perintah STOP ke mikrokontroler dan mengirim koordinat halangan ke *web app*. Perintah STOP berhasil terkirim dalam bentuk *bytearray*. Sedangkan pengiriman koordinat halangan berhasil terkirim dalam bentuk *string* sesuai dengan posisi halangan. Hasil pengiriman data dapat ditunjukkan pada Gambar 14.

```
0: 640x640 2 boxs, 29.3ms
Speed: 2.8ms preprocess, 29.3ms inference, 7.7ms p
ostprocess per image at shape (1, 3, 640, 640)
[558, 308]
---BLOCKED---
MR
---STOP---
Send Data : `bytearray(b'\xa5Z\x12\x00\x19\x00\x01
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00+')`
Send `messages: A55A21040P00P02P00P03P01P02P01P03`
 to topic `astar/amrparams`
STOP 1
STOP 2
STOP 3
```

Gambar 14. Hasil pengiriman koordinat halangan di tengah-kanan ke dalam *web*

Dari hasil pengiriman data pada Gambar 14. dapat diketahui bahwa koordinat halangan yang terkirim yaitu ‘A55A21040P00P02P00P03P01P02P01P03’. Koordinat halangan atau *block* bersifat relatif terhadap posisi robot.
Pengujian kondisi kedua di mana posisi halangan berada di tengah dan kiri dari robot. Koordinat *x* titik tengah dari objek yang terbaca adalah 161 pada halangan kiri dan 340 pada halangan tengah. Hasil pendeteksian objek berdasarkan kondisi halangan tersebut dapat dilihat pada Gambar 15.



Gambar 15. Hasil pendeteksian objek dengan kondisi halangan kedua

Setelah pembacaan koordinat halangan berhasil, akan dilakukan pengiriman data koordinat objek berdasarkan posisi pixel ke *web* dalam bentuk koordinat *grid*. Hasil pegiriman data koordinat halangan dalam kondisi kedua dapat dilihat pada Gambar 16.

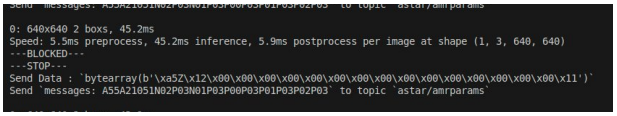
```
orin@orin: ~/vision 50x13
ostprocess per image at shape (1, 3, 640, 640)
[150, 377]
---BLOCKED---
MID-LEFT
---STOP---
Send Data : `bytearray(b'\xa5Z\x12\x00\x07\x00\x01
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x19')
`
Send `messages: A55A21040N01P02N01P03P00P02P00P03`
 to topic `astar/amrparams`
STOP 1
STOP 2
STOP 3
STOP 4
```

Gambar 16. Hasil pengiriman koordinat halangan di tengah-kiri ke dalam *web*

Berdasarkan Gambar 16. data yang terkirim adalah ‘A55A21040N01P02N01P03P00P02P00P03’. Dari pesan tersebut dapat disimpulkan bahwa jumlah *block* yang dikirim sebanyak 4 koordinat XY.
Pada pengujian kondisi ketiga yaitu saat posisi kedua halangan berada di tengah. Koordinat *x* dari masing-masing objek yang terbaca adalah 287 dan 351. Hasil pendeteksian objek berdasarkan kondisi halangan tersebut dapat dilihat pada Gambar 4.13 berikut.

Gambar 17. Hasil pendeteksian objek dengan kondisi halangan ketiga

Setelah program berhasil mendapatkan koordinat dari masing-masing objek. Robot akan berhenti sesuai dengan perintah yang dikirim oleh program. Kemudian program akan mengirimkan koordinat halangan tersebut ke *web* dalam bentuk *string* yang berisikan jumlah *block* dan koordinat halangan dalam *grid*. Hasil pengiriman data koordinat dalam kondisi halangan tersebut dapat dilihat pada Gambar 18.



Gambar 18. Hasil pengiriman koordinat halangan di tengah ke dalam *web*

Dengan demikian robot dapat menentukan posisi objek yang menghalangi jalur dan program dapat memperbarui *map* serta melakukan pencarian jalur ulang dengan algoritma A* pada *web*.

4. Kesimpulan

Berdasarkan hasil pengujian dapat disimpulkan bahwa model deteksi YOLOv8 yang dilatih sebanyak 300 epoch menghasilkan deteksi objek terbaik dengan nilai presisi, recall, dan akurasi masing-masing sebesar 91%, 93%, dan 95%, serta kecepatan deteksi rata-rata 45 ms (22 FPS), sementara model TensorRT memiliki kecepatan deteksi lebih cepat sebesar 30 ms (33 FPS). Sistem estimasi jarak bekerja optimal pada jarak 50-300 cm untuk kelas box dan 75-150 cm untuk kelas person dengan rata-rata error masing-masing 3.3% dan 13.4%. Sistem obstacle avoidance menunjukkan jangkauan deteksi hingga 300 cm, dengan nilai kepercayaan tertinggi untuk kelas box pada jarak 50-100 cm (95%) dan untuk kelas person pada jarak 200-250 cm (94%), serta berhasil menghindari halangan pada jarak 100 cm. Sistem navigasi AMR berhasil membaca posisi halangan dan mengirimkan koordinat ke web app, bekerja optimal dengan dua atau lebih halangan pada jarak 100 cm, dan mengaktifkan algoritma A* untuk memperbarui jalur navigasi.

Referensi

[1] A. Dhaliwal, "The Rise of Automation and Robotics in Warehouse Management," *Transforming Management Using Artificial Intelligence Techniques*, pp. 63–72, 2020.

[2] N. Rothong and C. Thongchaisuratkrul, "ROS Based Indoor AMR Design and Navigating Application," King Mongkut's University of Technology North Bangkok, Mar. 2020, pp. 291–294. doi: 10.14416/c.fte.2020.03.042.

[3] G. Pugliese, X. Chou, D. Loske, M. Klumpp, and R. Montemanni, "AMR-Assisted Order Picking: Models for Picker-to-Parts Systems in a Two-Blocks Warehouse," *Algorithms*, vol. 15, no. 11, Nov. 2022, doi: 10.3390/a15110413.

[4] J. W. G. Putra, "Pengenalan konsep pembelajaran mesin dan deep learning," *Tokyo. Jepang*, 2019.

[5] N. Rochmawati *et al.*, "Analisa Learning rate dan Batch size Pada Klasifikasi Covid Menggunakan Deep learning dengan Optimizer Adam." JIEET, vol. 05, no. 02, 2021

A. Raup, W. Ridwan, Y. Khoeriyah, Q. Yuliati Zaqiah, and U. Islam Negeri Sunan Gunung Djati Bandung, "Deep Learning dan Penerapannya dalam Pembelajaran." [Online]. Available: http://Jiip.stkipyapisdompu.ac.id

M. Monika, U. Rajender, A. Tamizhselvi, and A. S. Rumale, "REAL-TIME OBJECT DETECTION IN VIDEOS USING DEEP LEARNING MODELS," *ICTACT JOURNAL ON IMAGE AND VIDEO PROCESSING*, p. 2, 2023, doi: 10.21917/ijivp.2023.0441.

J. Chai, H. Zeng, A. Li, and E. W. T. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021, doi: 10.24433/CO.0411648.v1.

A. Glenn-Jocher, shcheklein, sergiuwaxmann, and Laughing-q, "Ultralytics YOLOv8 Docs: Home," https://docs.ultralytics.com/. Accessed: Apr. 15, 2024. [Online]. Available: https://docs.ultralytics.com/

K. Khairunnas, E. M. Yuniarno, and A. Zaini, "Pembuatan Modul Deteksi Objek Manusia Menggunakan Metode YOLO untuk Mobile Robot," *Jurnal Teknik ITS*, vol. 10, no. 1, pp. A50–A55, 2021.

H. Lou *et al.*, "DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor," *Electronics (Switzerland)*, vol. 12, no. 10, May 2023, doi: 10.3390/electronics12102323.

NVIDIA, "NVIDIA TensorRT Documentation." Accessed: Apr. 17, 2024. [Online]. Available: https://developer.nvidia.com/tensorrt#section-what-is-nvidia-tensorrt

H.-B. Yoo, † ·myeong-Suk Park, and † ·sang-Hoon Kim, "TensorRT와 SSD를 이용한 실시간 얼굴 검출방법 323 Real Time Face detection Method Using TensorRT and SSD," *KIPS Trans. Softw. and Data Eng*, vol. 9, no. 10, pp. 323–328, 2020, doi: 10.3745/KTSDE.2020.9.10.323.

S. O. Stepanenko and P. Y. Yakimov, "Using high-performance deep learning platform to accelerate object detection," 2019.

S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, "An improved A-Star based path planning algorithm for autonomous land vehicles," 2020, *SAGE Publications Inc.* doi: 10.1177/1729881420962263.