Implementasi Turret yang Dikendalikan YOLOv8 Untuk Identifikasi dan Tracking Target Bergerak

Alifio Noerifanza
School of Electrical Engineering
Telkom University
Surabaya, Indonesia
alifionsz@gmail.com

Susijanto Tri Rasmana
School of Electrical Engineering
Telkom University
Surabaya, Indonesia
susijanto@telkomuniversity.ac.id

Chaironi Latif
School of Electrical Engineering
Telkom University
Surabaya, Indonesia
chaironi@telkomuniversity.ac.id

kemajuan teknologi di bidang Artificial Intelligence membuka kesempatan baru bagi mesin perang otonom untuk membantu tugas tentara untuk melawan musuh. Hal ini telah dibuktikan dalam perang Rusia-Ukraina dengan penggunaan drone pengintai dan drone Kamikaze. Oleh karena itu negara-negara besar berlomba untuk menguasai bidang AI baik di sisi software maupun hardware. Walaupun level penggunaan Artificial Intelligence masih dalam tahap awal, sistem tracking dan targeting yang digunakan untuk menghancurkan kendaraan tempur sudah dalam penggunaan contohnya adalah drone Switchblade dan KUB-BLA. Drone-drone tersebut dapat mengidentifikasi dan mengklasifikasikan benda-benda yang dilihat oleh kameranya sehingga dapat membedakan mana target dan bukan targetnya. Dalam penelitian ini, implementasi turret yang dikontrol oleh YOLOv8 sukses diimplementasikan menggunakan NVIDIA Jetson Nano dan faktor-faktor yang mempengaruhi performa sistem deteksi telah teridentifikasi. Untuk performa dari Turret itu sendiri ada parameter yang mencapai harapan dan ada yang tidak. Untuk confidence di semua pengujian telah mencapai harapoan kecuali pengujian kedua horizontal, dimana Tingkat confidence hanya 0.689 dari 0.7 yang diharapkan. Untuk deviasi semua mencapai harapan yaitu rata-rata dibawah 40 pixel namun semua hasil runtime tidak mencapai harapan dengan hasil rata-rata terbaik hanya 601 ms sedangkan harapan adalah 200 ms. Akurasi juga tidak ada yang mencapai harapan dengan hasil terbaik adalah 27.1% dari 70% yang diharapkan. Rendahnya akurasi dan lambatnya runtime dikarenakan deteksi yang tidak stabil dan keterbatasan hardware NVIDIA Jetson menyebabkan turret memiliki kesulitan untuk tracking secara akurat. Diharapkan kedepannya dapat menggunakan kamera yang lebih baik disertai hardware yang mempunyai spesifikasi yang baik.

Kata kunci: ***Artificial Intelligence, Turret, Targeting, Tracking***

I. PENDAHULUAN

Kemajuan teknologi di bidang Artificial Intelligence membuka kesempatan baru bagi mesin perang otonom untuk membantu tugas tentara untuk melawan musuh. Hal ini telah dibuktikan dalam perang Rusia-Ukraina dengan penggunaan drone pengintai, drone Kamikaze dan negara-negara besar berlomba untuk menguasai bidang AI baik di sisi software maupun hardware [1] [2] [3] [4]. Bidang object recognition dari AI adalah bidang yang menjanjikan dikarenakan besar potensi penggunaannya di bagian sipil maupun militer [5] [6] [7] [8] [9]. Bidang ini memungkinkan mesin untuk mengenali dan mengklasifikasikan suatu objek menggunakan sensor optik atau yang biasa disebut kamera, serta sensor lainnya [6] [7] [8] [9] [10]. Hal ini memberikan kemampuan adaptasi yang lebih bagi mesin dimana ia dapat "diajari" untuk mengenali benda-benda serta kondisi benda yang ada didepannya [7] [9] [11]. Masalah yang harus dijawab adalah Bagaimana Implementasi Turret yang dikendalikan oleh YOLOv8 deteksi objek, Apa saja faktor-faktor yang mempengaruhi Yolov8 untuk tracking, dan Bagaimana performa turret yang dikontrol oleh YOLOv8 deteksi objek dalam kondisi nyata di lapangan.

Tujuan penelitian ini adalah mengimplementasikan metode dan bahan untuk membuat turret otonom yang dikontrol oleh YOLOv8 deteksi objek, mengidentifikasi faktor-faktor yang mempengaruhi performa program targeting yang dilengkapi oleh YOLOv8 dalam deteksi objek. Dan menilai performa turret yang dikontrol oleh YOLOv8 dalam kondisi nyata di lapangan.

II. KAJIAN TEORI

Menyajikan dan menjelaskan teori-teori yang berkaitan dengan variabel-variabel penelitian. Berikut adalah kajian teori penelitian ini:

i. Turret

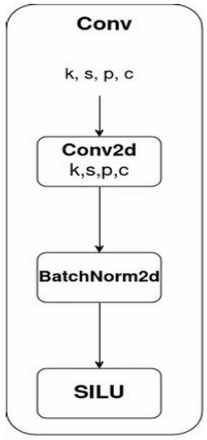
Turret adalah wadah penyangga senjata yang mampu berputar agar senjata dapat ditembakkan ke segala arah [12]. Turret juga dapat berupa kubah untuk melindungi kru atau operator sekaligus melindungi mekanisme senjata itu sendiri. Turret dapat dipersenjatai dengan satu atau lebih senjata seperti senapan, 1ernam, 1ernam, roket, dan rudal. Turret bisa diawaki oleh manusia, bisa dikendalikan dari jarak jauh, atau bekerja secara otonom. Dalam penelitian ini Turret menjadi objek utama penelitian dimana Turret menggunakan control dan targeting yang diserahkan kepada program targeting yang menggunakan algoritma deteksi objek YOLOv8 yang berjalan dalam Embedded sistem. Walaupun dikendalikan secara otomatis oleh program, control turret dapat diambil alih oleh operator jika keadaaan

mengharuskannya. Sistem control juga mempunyai GUI agar operator mengetahui proses targeting dan memudahkan pengecekan status serta control turret. Gambar 2.1 adalah gambar contoh turret:

GAMBAR 1
(RUPA GUN TURRET MILIK PT.RESPATI, PERUSAHAAN MILITER INDONESIA)

ii. Yolov8

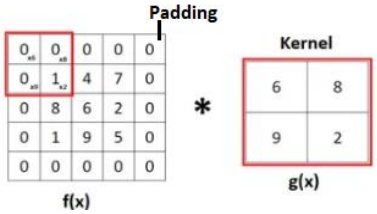
Yolo (*You Only Look Once*) adalah algoritma Artificial Intelligence pengenalan objek yang dapat mendeteksi secara cepat yang dikembangkan dari Arsitektur *Convolutional Neural Network* (CNN) [7] [6]. *Convolutional Neural Network* bekerja dengan *Convolutional Layer* yang mempunyai satu set Filter (dapat disebut Kernel) yang dapat diatur untuk mendeteksi pola tertentu [6] [9]. Gambar 3.2 dibawah ini adalah proses *Convolutional layer* Yolov8:



Penjelasan:
1. K: Jumlah filter (Kernel). Untuk mendeteksi feature dari objek
2. S: Stride. Besarnya step (pergeseran) setiap kernel saat menganalisa gambar.
3. P: Padding. penambahan pojokan bervalue 0 untuk memudahkan Analisa gambar. Juga untuk menjaga informasi agar tidak hilang/skip
4. C: Channel input, untuk setiap warna. Di gambar berwarna maka channel RGB dan YOLOV8 dapat menganalisa warna.
5. Batchnorm adalah normalisasi menyebabkan error

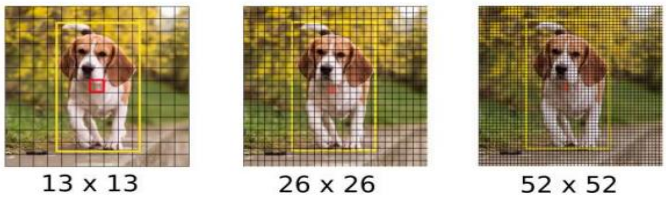
GAMBAR 2
(CONVOLUTION LAYER YOLOV8)

Setiap Filter memiliki ukuran persegi yang sama yang kemudian membaca gambar dalam format warna grayscale atau RGB. Untuk pinggiran data akan ditambahkan Padding agar data di pinggir gambar dapat diproses dengan baik. Berikut adalah penggambaran dari Kernel dan Padding.



GAMBAR 3
KERNEL DAN PADDING DI SISTEM CNN

Dan ini dibawah sebagai contoh pembagian gambar oleh *Convolutional Layer*:



GAMBAR 4
(ILUSTRASI PEMBAGIAN CONVOLUTION LAYER YOLO UNTUK DETEKSI OBJEK)

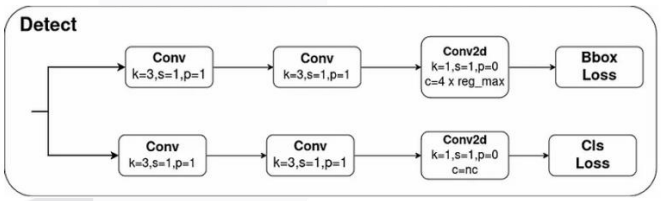
Saat gambar diproses oleh CNN, Filter tersebut akan membaca setiap pixel yang ada didalam area kerja filter tersebut diikuti dengan Fungsi Aktivasi yang akan menghasilkan peta fitur dari gambar tersebut dari step Conv2d. berikut adalah rumus Conv2d:

$$Y(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X(i+m, j+n) \times K(m,n) + b$$

Dimana X(I,j) adalah input gambar, K(m,n) adalah matrix Kernel, b adalah bias, dan M,N adalah ukuran kernel. Peta fitur ini akan menghasilkan input untuk pembuatan Weights. Weights adalah parameter yang memproses input menjadi data yang bisa digunakan untuk menyalakan *Activation Function* para *Neural Network* yang berfungsi untuk menghasilkan prediksi akhir [6] [9]. YOLOv8 menggunakan Fungsi Aktivasi 2ernama SiLU. Rumus SILU adalah:

$$SiLU(x) = x\left(\frac{1}{1+e^{-x}}\right)$$

Kemudian Setelah gambar diproses oleh CNN maka YOLOv8 akan melanjutkan proses ke layer deteksi. Ini adalah flowchart blok deteksi:



GAMBAR 1
(FLOWCHART BLOK DETEKSI)

Dalam layer ini hasil dari proses di layer CNN adalah dua output yaitu BBox loss dan Class loss. BBox adalah hasil koordinat dari deteksi dan Class adalah tipe objek yang telah dideteksi dan Loss disini adalah perbedaan antara hasil deteksi dan tabel Ground truth yaitu tabel referensi hasil Training yang kemudian berefek pada nilai Confidence. YOLOv8 dipilih sebagai AI Deteksi Objek karena memiliki kemampuan untuk Tracking serta lebih cepat dan efisien dalam processing power dibandingkan versi-versi sebelumnya dan memiliki akurasi yang lebih baik. Output Koordinat Bounding Box juga dapat digunakan sebagai perhitungan targeting sehingga memudahkan pembuatan program targeting Turret yang mengontrol pergerakan servo dengan informasi deviasi antara titik pusat kamera dan titik pusat Bounding Box.

iii. Nvidia Jetson Nano

NVIDIA Jetson Nano adalah salah satu platform embedded sistem yang dirancang untuk pengembangan aplikasi kecerdasan buatan dan deep learning. Jetson

memiliki konsumsi daya yang tergolong rendah yaitu 10watt sampai 30watt dan menyediakan input charger microUSB 12v 2ampere. Meskipun begitu, Nvidia Jetson dilengkapi dengan Graphic Processing Unit (GPU) Maxwell dengan 128 core CUDA yang mempunyai performa yang baik dalam aplikasi kecerdasan buatan dan deep learning. Dengan prosesor quad-core ARM Cortex-A57 berkecepatan 1.43 GHz, prosesor ini cukup kuat untuk menjalankan berbagai beban kerja AI dan komputasi parallel. Jetson Nano juga memiliki RAM 4 GB LPDDR4 yang memadai untuk menangani model deep learning dan aplikasi AI yang cukup kompleks. [13]. Jetson Nano dipilih karena mempunyai Pin GPIO yang dapat digunakan untuk control Servo Motor, mempunyai port embedded camera, serta memiliki fitur CUDA GPU yang dapat membantu performa Artificial *Neural Network*. NVIDIA Jetson Nano akan digunakan untuk menjalankan program targeting dengan algoritma YOLOv8 serta menjalankan program GUI. Dalam penelitian ini, Operating Sistem Jetson Nano yang digunakan adalah Ubuntu 20.04 dikarenakan library Ultralytics membutuhkan Python versi 3.8 keatas. Berikut adalah spesifikasi dari NVIDIA Jetson Nano:

TECHNICAL SPECIFICATIONS

GPU	NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores
CPU	Quad-core ARM Cortex-A57 MPCore processor
Memory	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Storage	16 GB eMMC 5.1
Video Encode	250MP/sec 1x 4K @ 30 (HEVC) 2x 1080p @ 60 (HEVC) 4x 1080p @ 30 (HEVC) 4x 720p @ 60 (HEVC) 9x 720p @ 30 (HEVC)
Video Decode	500MP/sec 1x 4K @ 60 (HEVC) 2x 4K @ 30 (HEVC) 4x 1080p @ 60 (HEVC) 8x 1080p @ 30 (HEVC) 9x 720p @ 60 (HEVC)
Camera	12 lanes (3x4 or 4x2) MIPI CSI-2 D-PHY 1.1 (1.5 Gb/s per pair)
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I^2C, I^2S, SPI, UART
Mechanical	69.6 mm x 45 mm 260-pin edge connector

GAMBAR 2
(DETAIL NVIDIA JETSON DENGAN SEBAGAI SPESIFIKASI [13])

iv. Servo Motor

Motor servo adalah sebuah motor DC dengan sistem umpan balik tertutup di mana posisi rotor-nya akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri dari sebuah motor DC, serangkaian gear, potensiometer, dan rangkaian kontrol. Servo motor dapat dikontrol dengan kecepatan, torque dan posisi tergantung pada nilai yang diberikan untuk mengendalikan motor secara presisi [14]. Keuntungan servo motor dari motor biasa adalah kontrol posisi dan kecepatan yang lebih baik dari motor biasa. menawarkan kemampuan perubahan posisi yang fleksibel. Servo motor tidak Stall karena ada koreksi dari Encoder. Motor jenis Servo Motor dipilih untuk menjadi penggerak azimuth dan elevasi Turret dikarenakan karakteristik tersebut, serta input dapat dimasukkan dengan nilai sudut sehingga lebih memudahkan dalam control dan menampilkan status turret dalam GUI. Servo motor yang akan digunakan dalam penelitian adalah MG995S. Servo motor MG995S akan dikontrol lewat motor driver module PCA9685 yang terhubung ke NVIDIA Jetson sehingga memudahkan algoritma targeting untuk mengendalikan azimuth dan elevasi turret. MG995S juga dilengkapi dengan sistem gear yang full metal sehingga meningkatkan durabilitas dan kemampuan untuk menahan beban atau inersia pergerakan turret. Berikut adalah spesifikasi dari servo motor MG995S serta foto penampakannya:

Fitur Servo Motor MG995S

- Berat: 55 g
- Dimensi: 40.7 x 19.7 x 42.9 mm approx.
- Torsi Terhenti: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Kecepatan pengoperasian: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Tegangan pengoperasian: 4.8 V to 7.2 V
- Lebar pita mati: 5 µs
- Desain bantalan bola ganda yang stabil dan tahan guncangan
- Kisaran suhu: 0 ºC – 55 ºC
- Sudut rotasi: 180deg. (+- 90 dari tengah)

GAMBAR 7
(SERVO MOTOR MG995 [14])

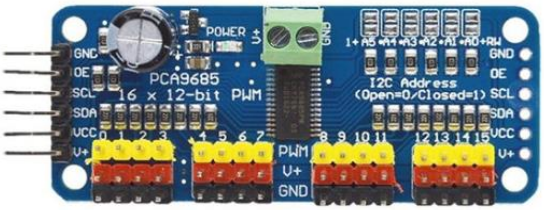
v. Driver Servo Motor PCA9685

Driver servo adalah sebuah modul untuk membantu Jetson mengontrol Servo motor dikarenakan library Python Jetson GPIO tidak memungkinkan untuk mengontrol PWM. Kontrol Driver Servo PCA9685 ini menggunakan library Adafruit_Servokit dan komunikasi I2C [15]. Pin Servo akan disambungkan ke pin V+, GND dan PWM dari driver servo motor. Dalam penelitian ini, dari 16 pin yang tersedia hanya dua pin yang akan digunakan yaitu untuk control azimuth dan elevasi dari turret tersebut. Berikut adalah Fitur dari PCA9685 dan gambar modulnya:

1. Modul dikontrol menggunakan sistem I2C sehingga hanya membutuhkan kabel SDA dan SCL saja untuk berkomunikasi dengan NVIDIA Jetson Nano.
2. PCA9685 hanya membutuhkan arus kecil untuk pengoperasiannya.
3. Tersedianya library Python untuk mengontrol driver motor ini sehingga memudahkan pemrograman secara keseluruhan dikarenakan hasil perhitungan dari program targeting dapat langsung diberikan ke modul ini oleh program Python untuk mengontrol servo.
4. Memiliki pin power dan ground tersendiri yang menyediakan daya eksternal langsung ke motor

servo sehingga tidak mengganggu operasi Jetson atau sistem lainnya.

5. Memiliki dimensi yang kecil yaitu Panjang 6cm dan lebar 2,5cm sehingga dapat dimasukkan kedalam frame turret dengan mudah.



GAMBAR 8
(DRIVER SERVO MOTOR PCA9685 [15])

III. METODE

3.1 Alur Penelitian

Penelitian dilakukan dengan pembuatan dan pemrograman turret kemudian dilanjutkan dengan pengambilan data. Secara keseluruhan, kegiatan penelitian terbagi menjadi 6 tahap besar yaitu:

1. Persiapan Model Deteksi Objek.

Pada tahap ini dilakukan persiapan model deteksi objek, yang mencakup tiga kegiatan utama yaitu persiapan custom dataset untuk training algoritma YOLOv8 yang kemudian dilanjutkan dengan Training custom model YOLOv8 yang dilakukan dengan program yang ada didalam library Ultralytics. Dengan menggunakan Labels, gambar dataset serta file config.yaml yang telah diisi dengan pengaturan training, program akan melakukan training custom model dari template model yolov8n.pt. Terakhir adalah Pengujian kesuksesan training yang dilakukan dengan test model terhadap Sebagian dari dataset itu sendiri. Kesuksesan dinilai dari kemampuan AI mendeteksi objek dengan benar setelah test tersebut.

2. Tahap Programming Sistem Deteksi dan GUI.

Tahap ini mencakup semua kegiatan programming dalam Python untuk menunjang kegiatan deteksi objek. Library yang digunakan adalah Ultralytics, Multiprocessing, Numpy, dan Tkinter.

3. Persiapan NVIDIA Jetson.

Pada tahap ini dilakukan persiapan NVIDIA Jetson sebagai Embedded sistem pada Turret. Dikarenakan bahasa pemrograman yang dipakai adalah Python, maka ada beberapa hal yang harus disiapkan sebelum dapat menjalankan program sistem deteksi GUI dan kontrol turret.

4. Tahap Desain Turret

Pada tahap ini dilakukan desain Turret dimana desain harus dapat mengakomodasi laser pointer,Nvidia Jetson, dan kamera yang akan dipasang satu paket dalam satu kotak penyimpanan serta penempatan servo motor dan alat kontrol seperti keyboard, mouse dan monitor.

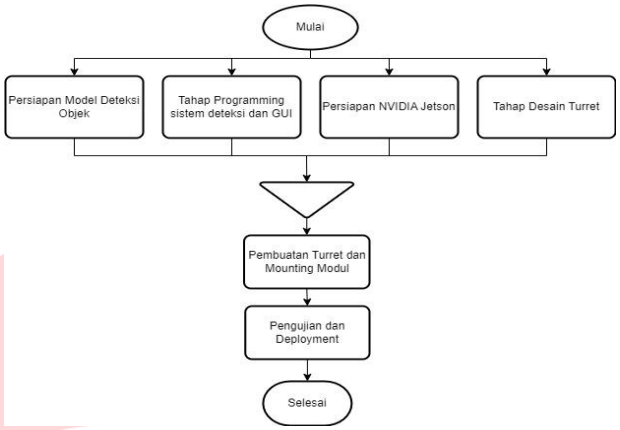
5. Pembuatan Turret dan Mounting Modul-Modul ke Turret.

Pada tahap ini dilakukan pembuatan frame turret dan mounting modul yang diperlukan untuk turret dapat berjalan dengan baik. Dalam tahap ini juga dilakukan troubleshooting dan penyesuaian ulang desain turret jika terjadi masalah.

6. Pengujian dan Deployment.

Pada tahap ini dilakukan pengujian akhir serta pengambilan data performa dari turret tersebut. Turret yang sudah beroperasi akan diuji dalam 3 ronde yang terbagi dalam tiga tipe pergerakan.

Tahap-tahap diatas ada yang dilakukan secara parallel untuk menghemat waktu, dan ada yang dilakukan setelah tahap lain selesai. Berikut adalah *Flowchart* kegiatan penelitian:



GAMBAR 9
(ALUR PENELITIAN)

3.2 Desain Sistem

1. Persiapan Model Deteksi Objek.

Pada tahap ini dilakukan persiapan model deteksi objek, yang mencakup tiga kegiatan utama yaitu:

- Persiapan custom dataset untuk training algoritma YOLOv8 dilakukan dengan mengambil foto drone yang akan menjadi Dataset. Akan ada 521 gambar Drone sedang terbang dan 882 Gambar Background yang akan dipersiapkan untuk training algoritma YOLOv8. Setelah foto-foto telah diambil maka akan dilakukan anotasi dimana gambar diberi bounding box dan label. Output dari proses ini adalah Labels dan gambar hasil anotasi yang akan menjadi dataset untuk custom model.
- Training custom model YOLOv8 dilakukan dengan program yang ada didalam library Ultralytics. Dengan menggunakan Labels hasil anotasi, gambar-gambar dataset serta file config.yaml yang telah diisi dengan pengaturan training, program akan melakukan training custom model dari template model yolov8n.pt. yolov8n adalah model Yolov8 yang dipilih karena mempunyai kecepatan deteksi yang lebih cepat namun mempunyai akurasi yang lebih rendah dari model lain. berikut adalah tabel performa dari model-model YoloV8:

TABEL 1
(KECEPATAN PERFORMA DETEKSI MODEL-MODEL YOLOV8 [15])

Model	Kecepatan dengan CPU (dalam millisecond)	Kecepatan dengan NVIDIA A100 (dalam millisecond)	Deskripsi
YOLOv8n (nano)	80.4	0.99	Model ini yang paling ringan dalam komputasi

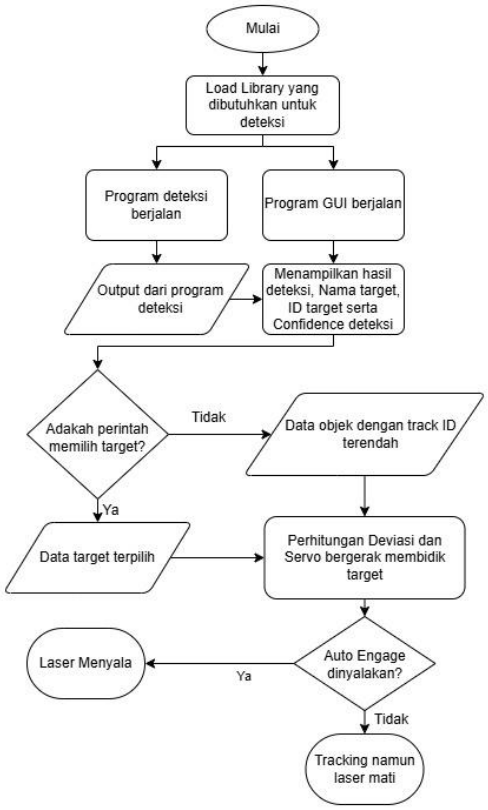
			Digunakan untuk tugas yang membutuhkan kecepatan proses deteksi
YOLOv8s (small)	128.4	1.20	Model ini yang ringan dalam komputasi dengan akurasi yang lebih baik dari versi nano
YOLOv8m (medium)	234.7	1.83	Model ini yang sedang dalam beban komputasi dengan akurasi yang sedang
YOLOv8l (large)	375.2	2.39	Model ini berat dalam komputasi namun performa lebih baik dari versi sedang
YOLOv8x (extra large)	479.1	3.53	Model ini yang paling berat dalam komputasi dengan akurasi yang terbaik..

- Setelah kedua tahap sebelumnya selesai, dilakukalah pengujian kesuksesan training dilakukan dengan video test.mp4 dimana model yang telah selesai training ditugaskan untuk mendeteksi objek didalam video yang telah ditentukan. Kesuksesan dinilai dari kemampuan custom model YOLOv8 mendeteksi objek dengan benar di dalam video. hasil deteksi dalam pengujian dicatat dan ditabelkan

2. Tahap Programming Sistem Deteksi dan GUI.

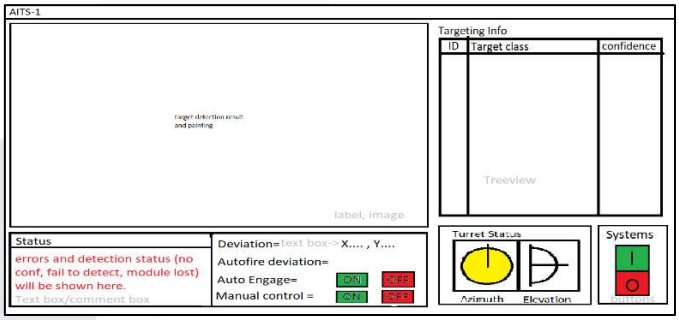
Tahap ini mencakup semua kegiatan programming dalam Python untuk menunjang kegiatan deteksi objek. Library yang digunakan adalah Ultralytics, multiprocessing, Numpy, dan Tkinter. Keluaran dari tahap ini adalah program sistem deteksi dan GUI dari Turret itu sendiri. Terdapat dua kegiatan dalam tahap ini yaitu:

- Tahap Programming sistem deteksi yang menggunakan custom mode. Tugas program ini adalah untuk dapat mengekstrak data targeting yang berupa Track ID, Track yang terpilih, Nama Track, Koordinat, Detection Confidence, dan Deviasi dari Aimpoint kamera. Program deteksi berjalan secara parallel dengan GUI menggunakan sistem Threading. Turret akan bergerak saat koordinat bounding box deteksi tidak ditengah layar, dengan deviasi -40, dan 40 Pixel dari pusat. Berikut adalah flowchart cara kerja sistem deteksi dan GUI Turret:



GAMBAR 10
(FLOWCHART CARA KERJA SISTEM DETEKSI DAN GUI TURRET)

- Tahap Programming GUI untuk Turret. Tujuan GUI adalah memudahkan pengguna untuk mengontrol turret serta menampilkan hasil deteksi. Walaupun sistem Auto-Engage ada, tetapi operator dapat mematikannya untuk tidak menembak secara otomatis jika tidak diberi perintah. Berikut adalah gambar layout dari program GUI:



GAMBAR 11
(CONTOH LAYOUT GUI YANG DIRENCANAKAN)

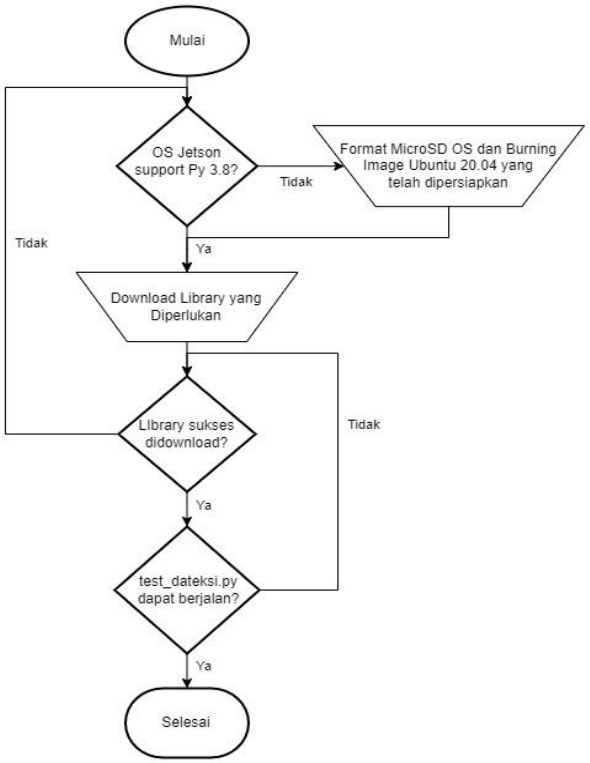
3. Persiapan NVIDIA Jetson.

Pada tahap ini dilakukan persiapan NVIDIA Jetson sebagai Embedded sistem pada Turret. Dikarenakan bahasa pemrograman yang dipakai adalah Python, maka ada beberapa hal yang harus disiapkan sebelum dapat menjalankan program sistem deteksi dan GUI turret. Pertama, diperlukan library Gstreamer dari Google untuk memproses gambar yang kamera IMX219-77 CSI tangkap agar dapat diproses oleh program Python di NVIDIA Jetson. Kemudian menginstal library-library Python diperlukan yaitu:

1. Ultralytics dan requirementnya
2. GPIO

3. Adafruit_servokit
4. PIL
5. Tkinter
6. Multiprocessing
7. pyTorch versi yang menunjang CUDA

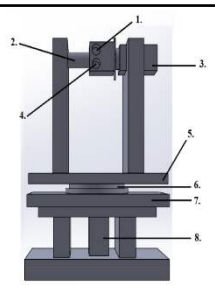
Setelah library yang dibutuhkan sukses di-download, maka akan dilakukan pengujian dengan menggunakan model pre-trained yolo versi nano yaitu Yolov8n.pt untuk verifikasi apakah semua proses berjalan dengan lancar. Ini dilakukan agar dapat mengetahui apakah ada error, bug dan versi library yang memiliki konflik dan memperbaiki masalah terkait dependency program deteksi objek. Dibawah ini adalah flowchart kegiatan persiapan NVIDIA Jetson sebagai Embedded Sistems:



GAMBAR 12
(FLOWCHART KEGIATAN PERSIAPAN NVIDIA JETSON)

4. Tahap Desain Turret

Pada tahap ini dilakukan desain Turret dimana desain harus dapat mengakomodasi laser pointer,Nvidia Jetson, dan kamera yang akan dipasang satu paket dalam satu kotak penyimpanan serta penempatan servo motor dan alat kontrol seperti keyboard, mouse dan monitor. Berikut adalah hasil desain yang direncanakan:



Keterangan:
1. Lubang Kamera
2. Shaft penyangga elevasi

GAMBAR 12
(DESAIN TURRET YANG SELESAI DIRENCANAKAN)

5. Pembuatan Turret dan Mounting Modul-Modul ke Turret.

Pada tahap ini dilakukan pembuatan frame turret dan mounting modul yang diperlukan untuk turret dapat berjalan dengan baik. Kegiatan ini terbagi menjadi tiga kegiatan utama yaitu:

1. Pembuatan Frame turret

Pembuatan frame turret dilakukan untuk mengakomodasi housing hardware pendukung objek deteksi. Pembuatan frame dilakukan dengan bantuan jasa tukang las, dikarenakan frame terbuat dari plat logam dan balok besi sehingga harus di-las untuk disatukan dan di bor untuk dapat memasang baut. Bentuk struktur turret mengikuti hasil desain turret di tahap sebelumnya.

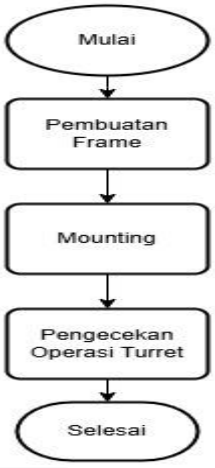
2. Mounting Modul-modul pada Frame Turret

Mounting dilakukan setelah frame selesai dibuat. Komponen internal tidak dipasang secara permanen untuk memungkinkan reposisi dan desain ulang di masa depan.

3. Pengecekan Operasi Turret

Pengecekan operasi turret dilakukan dengan memberikan beberapa miniatur dan gambar untuk dideteksi dan di track oleh turret. Pengecekan operasi turret dianggap sukses jika turret berhasil melakukan tracking objek dan sistem Auto-Engage laser berfungsi dengan baik dan tidak akan konflik komponen dan kabel saat pergerakan turret terjadi.

Berikut adalah flowchart kegiatan pembuatan frame turret dan mounting modul:

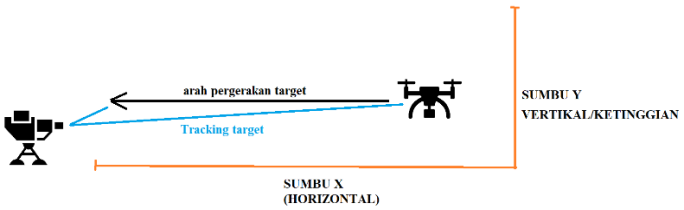


GAMBAR 13
(FLOWCHART KEGIATAN PEMBUATAN FRAME)

6. Pengujian dan Deployment.

Pada tahap ini dilakukan pengujian akhir serta pengambilan data performa dari turret tersebut. Turret yang sudah beroperasi akan diuji dalam 3 ronde yang terbagi dalam tipe pergerakan yang digambarkan dibawah ini:

1. Tracking objek bergerak horizontal

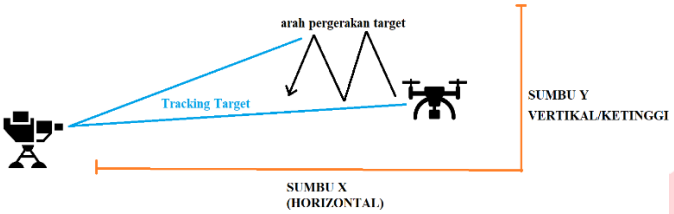


GAMBAR 13
(PENGUJIAN TRACKING SECARA HORIZONTAL)

2. Tracking objek bergerak vertikal

GAMBAR 14
(PENGUJIAN TRACKING SECARA VERTIKAL)

3. Tracking objek bergerak dengan acak



GAMBAR 15
(GAMBAR PENGUJIAN TRACKING SECARA RANDOM)

Teknik pengambilan data dilakukan dengan observasi, logging internal dari proses Python serta observasi status dari GUI dan kemampuan turret secara langsung. Hasil data pengujian didapat dari 3 ronde pengujian dimana Turret ditugaskan untuk melakukan tracking target yang berupa drone. Kemudian. Informasi runtime dan ketepatan deteksi, deviasi tracking objek, dan akurasi *engagement* dikumpulkan

IV. HASIL DAN PEMBAHASAN

1. Persiapan Model Deteksi Objek.

Model awal adalah YoloV8-Nano yang ditraining ulang menggunakan dataset custom dataset terdiri atas 521 gambar Drone sedang terbang dan 882 Gambar Background. Gambar untuk Dataset diambil di daerah Pandansari, Wagir, Kabupaten Malang. Semua Dataset yang diambil saat cuaca cerah. Berikut adalah contoh gambar-gambar yang digunakan untuk Dataset:

TABEL 2
(CONTOH-CONTOH GAMBAR YANG AKAN DIGUNAKAN DALAM DATASET)

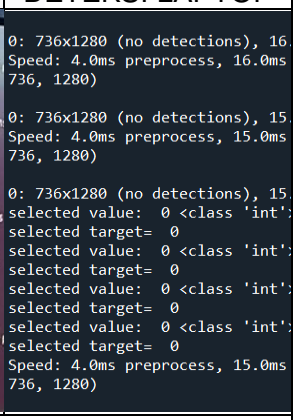
DRONE UNTUK DATASET


BACKGROUND UNTUK DATASET


2. Tahap Programming Sistem Deteksi dan GUI.

Tahap ini mencakup semua kegiatan programming program Python untuk menjalankan kegiatan deteksi objek. Library yang digunakan adalah Ultralytics, Multiprocessing, Numpy, dan Tkinter. Program ini sukses dengan GUI dan program deteksi berjalan secara paralel bersamaan dengan cara menggunakan library Multiprocessing. Programming dilakukan di Laptop. Kedua program deteksi menggunakan proses CUDA sehingga menggunakan GPU untuk proses deteksi, bukan menggunakan CPU. Berikut adalah dokumentasi hasil dan perbandingannya:

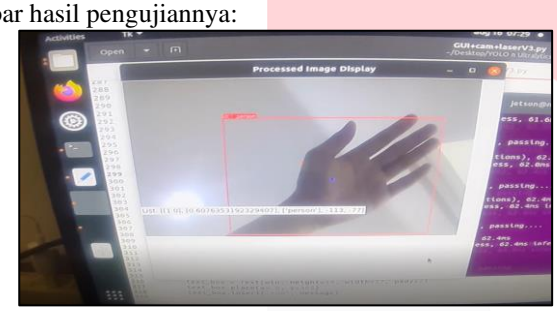
TABEL 3
(DOKUMENTASI PERBANDINGAN KECEPATAN DETEKSI SERTA HASIL AKHIR GUI)

KECEPATAN DETEKSI JETSON	KECEPATAN DETEKSI LAPTOP
	

HASIL GUI DETEKSI

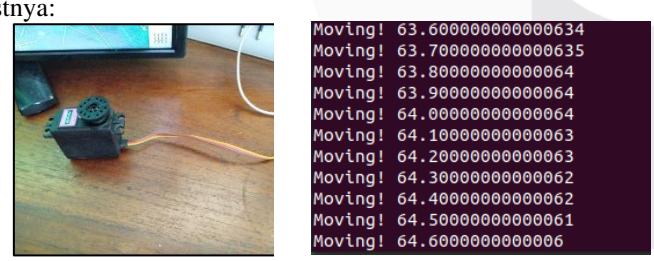

3. *Persiapan NVIDIA Jetson.*

Pada tahap ini dilakukan persiapan NVIDIA Jetson sebagai Embedded sistem pada Turret. Dikarenakan bahasa pemrograman yang dipakai adalah Python, maka ada beberapa hal yang harus disiapkan sebelum dapat menjalankan program sistem deteksi dan GUI turret. Setelah membuat bootloader Ubuntu 20.04 ke dalam kartu SD untuk Jetson, dilakukan instalasi library-library Python yang dibutuhkan dengan Pip. Untuk G-Streamer dilakukan instalasi menggunakan Bash. Untuk membuat program deteksi menggunakan GPU, library Torch harus diinstall dengan versi khusus yang mendukung pemrosesan menggunakan CUDA. CUDA adalah library program dari NVIDIA untuk membuat GPU yang diproduksi NVIDIA dapat digunakan untuk segala proses yang berkaitan dengan Artificial Intelligence terutama *Neural Network*. Setelah selesai instalasi, dilakukan pengujian menjalankan objek deteksi dengan kamera menggunakan model YoloV8n.pt untuk verifikasi bahwa tahap ini sukses. Berikut adalah gambar hasil pengujiannya:



GAMBAR 16
(KAMERA JETSON BERHASIL MENGAMBIL GAMBAR TANGAN DAN PROGRAM DETEKSI OBJEK BERHASIL MENGINDENTIFIKASI OBJEK BERUPA ORANG)

Kemudian dilanjutkan dengan instalasi library ”adafruit_servokit” untuk dapat menjalankan servo yang mengontrol azimuth dan elevasi turret. Instalasi berhasil sehingga Jetson dapat mengontrol Servo. Berikut adalah hasil testnya:



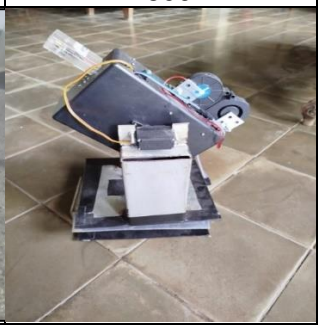
(a) (b)

GAMBAR 17
(DOKUMENTASI PENGUJIAN SERVO. (A) SERVO BERHASIL BERGERAK MENGIKUTI PROGRAM. (B) SCREENSHOT PROGRAM KONTROL SERVO)

4. *Pembuatan Turret dan Mounting Modul-Modul ke Turret.*

Pada tahap ini berjalan mulus dan Frame Turret berhasil dibuat dengan kemampuan elevasi 120 derajat elevasi dan 360 derajat azimuth. Kipas pengingin berhasil dipasang diatas turret dan modul driver servo serta kamera tertanam didalam turret untuk. Namun, saat dipasangkan servo motor kemampuan maksimal berkurang menjadi 240 derajat azimuth dan saat dikontrol melalui library “adafruit_servokit” hanya mampu bergerak sampai 160 derajat azimuth dalam kenyataannya walau dalam kontrol Jetson diperintahkan untuk mencapai 180 derajat. Berikut adalah gambar hasil akhir dari pembuatan Turret:

TABEL 4
(GAMBAR-GAMBAR TURRET SETELAH SELESAI DIBUAT)

Turret telah selesai dibuat	Setelah mounting modul
	

5. *Pengujian dan Deployment.*

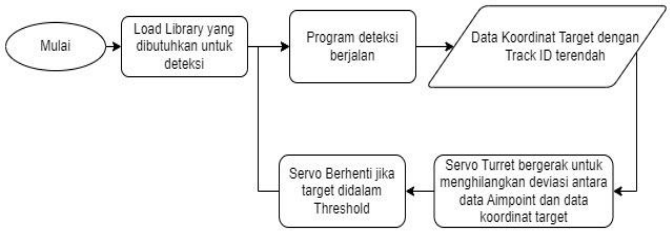
Pengujian dilakukan di dua tempat, yaitu di dekat Gazebo Telkom University Surabaya dan di Masjid al-takhobbar Telkom University Surabaya. Kondisi cuaca dekat Gazebo adalah cerah tanpa awan, sedangkan kondisi cuaca dalam test kedua di masjid al-takhobbar adalah mendung dengan backlight bagi kamera. hasil pengujian adalah log berupa file teks (.txt) dari turret yang melacak pergerakan drone yang ditulis dalam format CSV agar memudahkan pengolahan data. Pengujian ini bertujuan untuk mengevaluasi performa Turret dalam kondisi nyata yang tidak optimal. Tangkapan layar saat menjalankan program deteksi dilakukan oleh HP android dikarenakan program sudah menggunakan kemampuan maksimum hardware Jetson sehingga tidak bisa menjalankan program lain.

A. Test pertama

Test pertama dilakukan dengan drone digerakkan dengan tangan secara horizontal dan vertikal, dikarenakan kecepatan angin tidak memungkinkan drone untuk melayang dengan stabil. Program berjalan dengan lancar namun GUI memakan terlalu banyak RAM dan processing power sehingga menyebabkan notifikasi RAM maksimum, dan memberi peringatan overcurrent yang berujung pada *sistem freeze* setelah mencapai durasi 1 menit, mengakibatkan hilangnya data log internal karena Jetson harus di restart dengan cara mencabut kabel powernya. Oleh karena ini penulis akan menghilangkan fitur GUI untuk menjalankan software lebih baik di pengujian berikutnya. Dibawah ini adalah hasil tangkapan layarnya:

GAMBAR 18
(TANGKAPAN LAYAR GUI DARI PROGRAM DETEKSI OBJEK OLEH HP ANDROID)

Kemudian setelah dihilangkan GUI program berjalan lebih stabil dengan log internal dapat berjalan. Dikarenakan adanya perubahan cara kerja sistem maka flowchart berubah menjadi seperti yang diperlihatkan dibawah ini:



GAMBAR 19
(FLOWCHART CARA KERJA SISTEM SETELAH GUI DIHILANGKAN)

Dan berikut adalah hasil tangkapan layar pengujian tanpa GUI:

GAMBAR 20
(TANGKAPAN LAYAR DARI PENGUJIAN KEDUA DI GAZEBO)

Program berjalan tanpa mengalami freeze dan menghasilkan log berupa file teks (.txt) yang ditulis dalam format CSV dilampirkan. Kemudian Test kedua dilakukan di masjid Al-Takhobbar dikarenakan masalah di kabel power serta cuaca yang memburuk mengharuskan pengujian dilakukan dekat dengan stopkontak dan dibawah naungan. Cuaca mendung dengan backlight dari matahari. Pengujian berjalan sesuai dengan rencana yaitu dibagi tiga ronde yaitu horizontal, vertikal, dan random. Berikut adalah hasilnya:

A. Pengujian di Gazebo

Deviasi: Rata-rata deviasi X sebesar -10.7 pixel dan deviasi Y sebesar 19.5 pixel menunjukkan akurasi yang cukup baik dalam lintasan horizontal. Rata-rata confidence sebesar 0,714. Performa mencapai yang diharapkan. Rata-rata waktu antara deteksi adalah 0,75 detik mencerminkan sistem memiliki respons yang jauh dari yang diharapkan dari 0.20 detik (200 milisecond). Dan Akurasi tracking: Rata-Rata laser I/O adalah 0.271 yaitu hanya 27,1% dari total kejadian deteksi dimana Turret sukses membidik kearah target.

B. Pengujian di masjid Al-Takhobbar

1. Pengujian Horizontal

Deviasi: Rata-rata deviasi X sebesar -1 pixel dan deviasi Y sebesar 11,89 pixel menunjukkan akurasi yang cukup baik dalam lintasan horizontal. Rata-rata confidence sebesar 0,689. Performa belum mencapai yang diharapkan. Rata-rata waktu antara deteksi adalah 2,64 detik mencerminkan sistem memiliki respons yang jauh dari yang diharapkan. Dan Akurasi tracking: Rata-Rata laser I/O adalah 0.182 yaitu hanya 18,2% dari total kejadian deteksi dimana Turret sukses membidik kearah target.

2. Pengujian Vertikal

Deviasi: Rata-rata deviasi X sebesar 11.34 pixel dan deviasi Y sebesar 4,97 pixel menunjukkan beberapa tantangan dalam mempertahankan akurasi untuk perubahan elevasi. Rata-rata confidence sebesar 0,716 dan mencapai harapan. Waktu rata-rata sebesar 0,601 detik menunjukkan performa yang jauh dari yang diharapkan namun lebih baik pengujian horizontal. Dan Akurasi tracking: Rata-Rata laser I/O adalah 0.27 yaitu hanya 27% dari total kejadian deteksi dimana Turret sukses membidik kearah target.

3. Pengujian Random

Deviasi: Deviasi X rata-rata 18.73 pixel dan deviasi Y rata-rata 0.461 pixel menunjukkan turret memiliki kesulitan yang lebih besar dalam tracking pergerakan acak. Confidence rata-rata 0,724 dan memenuhi harapan. Waktu rata-rata 2,52 detik menunjukkan kecepatan respons masih jauh dari harapan. Akurasi tracking: Rata-Rata laser I/O adalah 0 yaitu tidak ada kejadian deteksi dimana Turret sukses membidik kearah target.

V. KESIMPULAN

Turret yang dikontrol oleh YOLOv8 atau AI objek deteksi lainnya telah berhasil diimplementasikan. Turret memiliki kemampuan pergerakan azimuth 160^0 dan elevasi 160^0 dan dapat melakukan tracking target. Namun, jika menggunakan GUI, sistem freeze dapat terjadi dikarenakan keterbatasan kemampuan komputasi dan RAM NVIDIA Jetson Nano. Ditemukan juga faktor-faktor yang mempengaruhi program deteksi dan tracking yaitu:

1. GUI meningkatkan konsumsi RAM dan CPU walaupun sistem deteksi sudah menggunakan GPU. Di laptop program yang menggunakan GUI memakai RAM 5.5 GB sedangkan yang tidak memakai GUI mencapai 3GB sedangkan RAM Nvidia Jetson Nano hanya 4GB
2. Efek kondisi cuaca pada kamera mengurangi kemampuan deteksi.
3. Efek background yang kompleks mengurangi kemampuan deteksi.
4. Kecepatan target semakin cepat mengurangi kemampuan deteksi.
5. Kesulitas tracking dikarenakan deteksi tidak stabil.
6. Model memiliki Dataset terbatas yaitu hanya 521 gambar drone dan 882 background dan perbedaan tempat pengambilan dataset dan deployment.
7. Bias dalam dataset dimana gambar background lebih banyak dari gambar target.

Untuk performa dari Turret itu sendiri ada parameter yang mencapai harapan dan ada yang tidak. Untuk confidence di semua pengujian telah mencapai harapoan kecuali pengujian kedua horizontal, dimana Tingkat confidence hanya 0.689 dari 0.7 yang diharapkan. Untuk deviasi semua mencapai harapan yaitu rata-rata dibawah 40 pixel namun semua hasil runtime tidak mencapai harapan dengan hasil rata-rata terbaik hanya 601 ms sedangkan harapan adalah 200 ms. Akurasi juga tidak ada yang mencapai harapan dengan hasil terbaik adalah 27.1% dari 70% yang diharapkan. Dapat disimpulkan bahwa sistem belum memuaskan terutama untuk penggunaan lapangan terutama dalam militer dikarenakan waktu proses yang lama dan akurasi yang rendah.

REFERENSI

[1] Y. Zhu, "Target Detection Based on Deep Learning," *Journal of Physics: Conference Series,* 2022.

[2] D. Yang, "Research and Implementation of Embedded Real time Target Detection Algorithm Based on Deep Learning," *Journal of Physics: Conference Series,* 2021.

[3] L. S. Xiuli Du, "A Lightweight Military Target Detection Algorithm Based on Improved YOLOv5," *MDPI Electronics,* 2022.

[4] B. Pong, "The Art of Drone Warfare," *Journal of War & Culture,* pp. 377-387, 2022.

[5] T. J. Oleksii Kostenko, "Problems of Using Autonomous Military AI Against the Background of Russia's Military Aggression Against Ukraine," *Baltic Journal of Legal and Social Sciences,* no. 4, 2022.

[6] A. Noerifanza, "Analisa Kelayakan Modul Esp32 Sebagai Kamera untuk Pengenalan Objek Sehari-hari," *Journal of Computer, Electronic, and Telecommunication,* vol. 3, 2022.

[7] in *Metodologi penelitian kualitatif*, Bandung, Remaja Karya, 2002, p. 410.

[8] H. Liu, "A Military Object Detection Model of UAV Reconnaissance Image and Feature Visualization," *Applied Science,* 2022.

[9] I. A. R. Lembang, "Indonesia Military Research and Development in Dealing With The Sixth Generation Warfare : The Use of Artificial Intelligence in War Operations," *East Asian Journal of Multidisciplinary Research (EAJMR),* vol. 2, pp. 649-660, 2023.

[10] S. Kuswadi, "Gun turret automatic weapon control system design and realization," *International Symposium on Electronics and Smart Devices (ISESD),* pp. 30-34, 2016.

[11] D. Kunertova, "The war in Ukraine shows the game-changing," *Bulletin of the Atomic Scientists,* pp. 95-102, 2023.

[12] J. Johnson, "Artificial intelligence & future warfare: implications for international security," *Defense & Security Analysis,* 2019.

[13] H. Fiyad, "Real Time Embedded Target Detection and Warning Systems," *Journal of Physics: Conference,* 2019.

[14] B. Earl, "Adafruit PCA9685," [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/16-channel-pwm-servo-driver.pdf. [Accessed 8 january 2025].

[15] M. K. Anwar, "Deep Features Representation for Automatic Targeting System of Gun Turret," in *International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, Bali, 2018.

[16] Ultralytics, "Ultralytics YOLOv8," [Online]. Available: https://github.com/ultralytics/ultralytics. [Accessed 24 June 2024].

[17] OMRON, "Technical Explanation for Servomotors and Servo Drives," [Online]. Available: https://www.ia.omron.com/data_pdf/guide/14/servo_tg_e_1_1.pdf. [Accessed 14 April 2024].

[18] BeneWake, "SEN0259 Datasheet," [Online]. Available: https://www.farnell.com/datasheets/3963986.pdf. [Accessed 06 May 2024].

[19] NVIDIA, "https://developer.nvidia.com/embedded/jetson-nano," Nvidia Corporation, [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano. [Accessed 06 May 2024].