

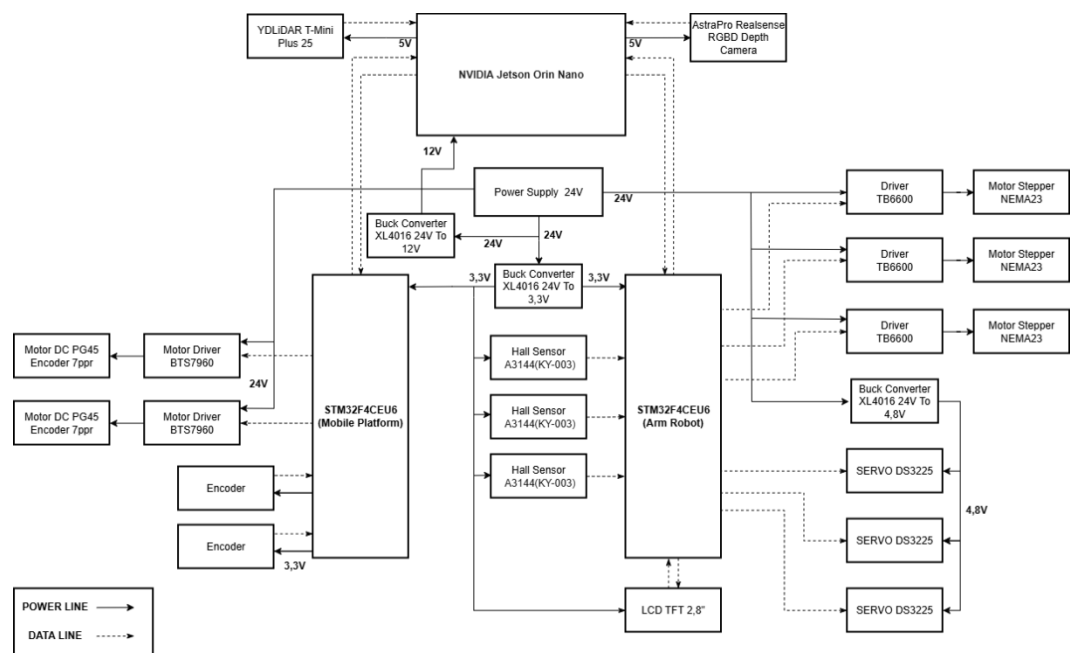
BAB III

METODE PENELITIAN

3.1 Diagram Blok

3.1.1 Diagram Blok Komponen

Diagram blok komponen menyajikan ilustrasi umum dari rancangan sistem visi untuk deteksi, segmentasi, dan penentuan target cabai merah matang (*Capsicum annuum L.*) berbasis YOLOv8n-Seg menggunakan kamera Astra Pro RGB dengan konfigurasi *eye-in-hand*. Sistem diproses oleh NVIDIA Jetson Orin Nano dan digunakan sebagai masukan pada sistem *Visual Servoing* yang diimplementasikan pada *Robot Mobile Chilli Picker*. Diagram blok komponen dapat dilihat pada **Gambar 3.1**.



Gambar 3.1 Diagram Blok Sistem Keseluruhan

Diagram blok pada **Gambar 3.1** tersusun atas sejumlah komponen. Berikut dibawah ini adalah penjelasan singkat komponen yang digunakan:

a. Baterai 24 V

Baterai 24 V berfungsi sebagai sumber daya utama sistem. Tegangan dari baterai didistribusikan ke seluruh komponen melalui rangkaian pengaman dan konverter tegangan sesuai kebutuhan masing-masing perangkat.

b. Fuse 15 A

Fuse 15 A berfungsi sebagai pengaman utama rangkaian listrik. Komponen ini memutus aliran listrik ketika terjadi arus berlebih atau hubungan singkat.

c. Power Switch

Power switch digunakan untuk menghubungkan dan memutus sumber daya utama sistem. LED 24 V digunakan sebagai indikator bahwa sistem telah memperoleh suplai daya.

d. Terminal Distribusi

Terminal distribusi berfungsi sebagai titik pembagian tegangan 24 V menuju aktuator dan rangkaian *stepdown converter*.

e. Stepdown Converter 24 V ke 12 V

Stepdown converter 24 V ke 12 V digunakan untuk menurunkan tegangan sumber menjadi 12 V sebagai suplai NVIDIA Jetson Orin Nano dan kipas pendingin.

f. Stepdown Converter 24 V ke 5 V

Stepdown converter 24 V ke 5 V digunakan untuk menyediakan tegangan 5 V bagi mikrokontroler dan beberapa perangkat elektronik pada sistem.

g. Stepdown Converter 24 V ke 3,3 V

Stepdown converter 24 V ke 3,3 V digunakan untuk menyediakan tegangan kerja bagi sensor yang membutuhkan suplai 3,3 V, seperti Hall Sensor.

h. NVIDIA Jetson Orin Nano

NVIDIA Jetson Orin Nano berfungsi sebagai unit komputasi utama sistem visi. Perangkat ini menjalankan pemrosesan citra, YOLOv8n-Seg, penentuan target, serta perhitungan koreksi gerakan menggunakan *Visual Servoing*. Jetson juga digunakan untuk menjalankan GUI dan komunikasi dengan STM32.

i. Astra Pro RGB Camera

Astra Pro digunakan untuk memperoleh citra RGB cabai sebagai masukan sistem visi. Kamera dipasang dengan konfigurasi *eye-in-hand* sehingga bergerak mengikuti pergerakan lengan robot.

j. STM32F411CEU6 Robot arm

STM32F411CEU6 pada lengan robot berfungsi sebagai pengendali gerakan aktuator. Mikrokontroler menjalankan proses *homing*, *inverse kinematics*, *forward kinematics*, serta mengeksekusi perintah gerak yang diterima dari Jetson.

k. Motor driver TB6600

Motor driver TB6600 berfungsi sebagai penghubung antara STM32 dengan motor NEMA 23. *Driver* menerima sinyal *pulse* dan *direction* untuk mengatur arah serta jumlah langkah motor.

l. Motor Stepper NEMA 23

Motor NEMA 23 digunakan sebagai aktuator pada beberapa sendi utama lengan robot. Gerakan motor digunakan untuk mengatur posisi lengan agar dapat menjangkau target cabai.

m. Motor Servo DS3225

Motor Servo DS3225 digunakan untuk menggerakkan bagian *wrist* dan *gripper*. Posisi servo diatur menggunakan sinyal PWM dari STM32F411CEU6.

n. Hall Sensor

Hall Sensor digunakan sebagai sensor referensi posisi pada lengan robot. Sensor ini membantu menentukan posisi awal setiap sendi saat proses *homing*.

o. STM32F411CEU6 AGV

STM32F411CEU6 pada AGV berfungsi sebagai pengendali pergerakan *mobile platform*. Mikrokontroler mengatur motor penggerak serta membaca data sensor dan *encoder* yang digunakan pada sistem AGV.

p. BTS7960 Motor driver

BTS7960 merupakan *motor driver* yang digunakan untuk mengatur arah dan kecepatan Motor DC PG45. Pengendalian dilakukan berdasarkan sinyal PWM dari STM32F411CEU6.

q. Motor DC PG45 Encoder

Motor DC PG45 digunakan sebagai penggerak roda pada AGV. *Encoder* pada motor memberikan informasi putaran roda sebagai *feedback* untuk pengendalian pergerakan robot.

r. YDLIDAR T-Mini Plus 25

YDLIDAR T-Mini Plus 25 digunakan untuk memperoleh informasi jarak dan kondisi lingkungan di sekitar AGV. Data hasil pemindaian dikirimkan ke NVIDIA Jetson Orin Nano melalui USB.

s. IMU GY-25

IMU GY-25 digunakan untuk memperoleh informasi orientasi dan perubahan arah AGV selama bergerak.

t. TFT LCD Module 2,8 Inch

TFT LCD Module 2,8 inch digunakan untuk menampilkan informasi dan status sistem kepada pengguna. Komunikasi dengan STM32 dilakukan melalui antarmuka SPI.

u. *Cooling fan* DC 12 V

Kipas pendingin 12 V digunakan untuk membantu menjaga temperatur perangkat elektronik agar tetap stabil selama sistem beroperasi.

Komponen yang berada pada **Gambar 3.1** memerlukan sumber tegangan dan arus yang cukup untuk mengaktifkan alat selama minimal satu jam untuk mengambil data, berikut **Tabel 3.1** yang menunjukkan kebutuhan daya sistem.

Tabel 3.1 Sumber Daya yang Dibutuhkan Sistem

No.	Komponen	Tegangan (V)	Arus (A)	Daya $P = V \times I$ (W)
1	NVIDIA Jetson Orin Nano	12	2,08	25,00
2	Astra Pro RGB Camera	5	0,44	2,20
3	YDLIDAR T-Mini Plus 25	5	1,00	5,00
4	STM32F411CEU6 Robot arm	5	0,10	0,50
5	STM32F411CEU6 AGV	5	0,10	0,50
6	Servo DS3225 (3 buah)	4,8	5,70	28,50
7	Hall Sensor (3 buah)	3,3	0,06	0,20
8	TFT LCD Module 2,8 inch	3,3	0,15	0,75
9	IMU GY-25	5	0,03	0,15
10	Cooling fan DC	12	0,09	1,08
11	Pilot lamp LED	24	0,02	0,48
12	TB6600 dan NEMA 23 (3 set)	24	3,30	79,20
13	BTS7960 dan Motor DC PG45 (2 set)	24	5,00	120,00
Total				263,56 W

Berdasarkan **Tabel 3.1**, total daya yang dibutuhkan sistem adalah sebesar 263,56 W. Jika sistem dirancang beroperasi selama 1 jam (tt), maka energi yang dibutuhkan dapat dihitung menggunakan Persamaan (3.1).

$$E = P \times t \quad (3.1)$$

$$E = 263,56 \times 1 \quad (3.2)$$

$$E = 263,56 \text{ Wh}$$

Selanjutnya, kapasitas baterai dalam satuan *Ampere-hour* (Ah) dapat dihitung berdasarkan energi yang dibutuhkan sistem dan tegangan baterai sebesar 24 V, seperti ditunjukkan pada Persamaan (3.3).

$$Ah = \frac{Wh}{V} \quad (3.4)$$

$$Ah = \frac{263,56}{24} \quad (3.5)$$

$$Ah = 10,98 \text{ Ah}$$

Berdasarkan perhitungan tersebut, kapasitas baterai minimum secara ideal adalah sebesar 10,98 Ah untuk waktu operasi selama 1 jam. Namun, pada kondisi sebenarnya terdapat rugi-rugi daya pada *stepdown converter* dan rangkaian distribusi daya. Dengan asumsi efisiensi sistem sebesar 90%, kebutuhan kapasitas baterai menjadi:

$$Ah_{efisiensi} = \frac{10,98}{0,9} \quad (3.6)$$

$$\begin{aligned} Ah_{efisiensi} \\ = 12,20 \text{ Ah} \end{aligned} \quad (3.7)$$

Selain itu, untuk mengantisipasi lonjakan arus pada motor *stepper*, motor DC, dan motor servo, digunakan *safety margin* sebesar 30%.

$$Ah_{real} = 12,20 \times 1,3 \quad (3.9)$$

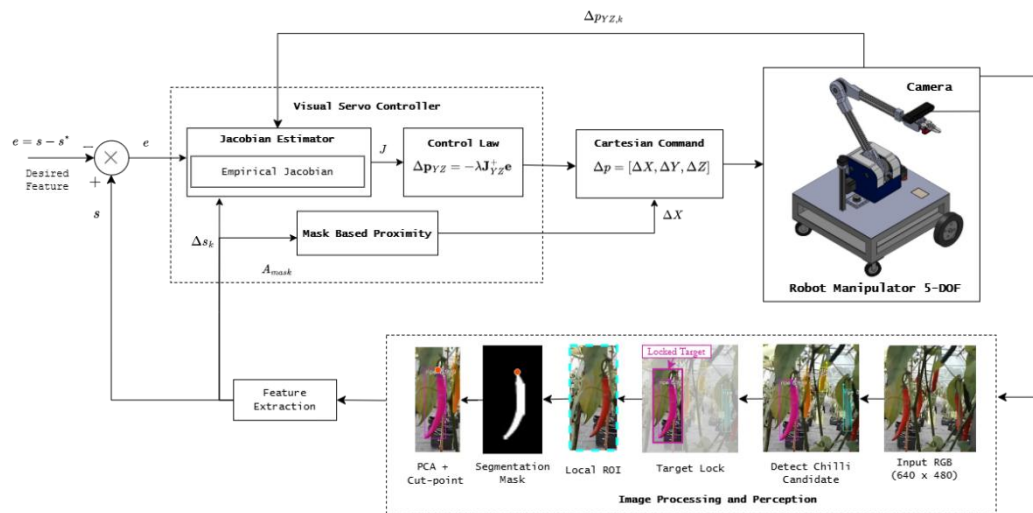
$$Ah_{real} = 15,86 \text{ Ah} \quad (3.10)$$

Berdasarkan hasil perhitungan total daya serta penambahan faktor efisiensi dan *safety margin*, sistem membutuhkan kapasitas baterai minimum sebesar 15,86 Ah pada tegangan 24 V untuk waktu operasi selama 1 jam. Oleh karena itu, kapasitas baterai yang digunakan harus lebih besar dari kebutuhan minimum agar mampu menyediakan daya secara stabil serta mengakomodasi lonjakan arus selama sistem beroperasi.

3.1.2 Diagram Blok Aritektur Kontrol Visual

Diagram blok menunjukkan arsitektur kontrol visual *closed-loop* yang digunakan untuk mengarahkan robot manipulator 5-DOF berdasarkan informasi citra dari kamera *eye-in-hand*. Sistem dibagi menjadi tiga bagian utama, yaitu *image processing and perception*, *visual servo controller*, dan

eksekusi gerak robot. Sistem membentuk *closed-loop visual feedback* karena informasi visual yang diperoleh kamera digunakan untuk menentukan koreksi gerak robot, kemudian perubahan posisi robot kembali diamati oleh kamera untuk menentukan koreksi berikutnya. Pola pengamatan-gerak-pengamatan ini membentuk kontrol *step-look*.



Gambar 3.2 Diagram Blok Arsitektur *Closed-Loop Step-look Uncalibrated image-based visual servoing* dengan *Fixed Empirical Jacobian* dan *Mask-Based Proximity* pada Robot Manipulator 5-DOF

Gambar 3.2 menunjukkan arsitektur kontrol visual yang digunakan pada Chimi-Bot. Berikut penjelasan tiap bagian utama diagram blok:

a. Image Processing and Perception

Bagian *image processing and perception* berfungsi memperoleh fitur visual target dari citra kamera. Kamera RGB Astra Pro dengan konfigurasi *eye-in-hand* menghasilkan citra beresolusi 640×480 piksel yang diproses pada NVIDIA Jetson Orin Nano menggunakan YOLOv8n-Seg untuk mendeteksi cabai dan menghasilkan *instance mask*. Pada kondisi terdapat beberapa cabai, sistem memilih cabai matang sebagai target dan mempertahankannya menggunakan mekanisme *target lock*. Area target kemudian dibatasi menggunakan *local ROI* agar pengamatan tetap berfokus pada objek yang sama. Dari *instance mask* target, sistem menentukan *cut-point* menggunakan PCA.

Koordinat *cut-point* serta karakteristik *mask* selanjutnya digunakan sebagai masukan bagi sistem kontrol visual.

b. *Visual Servo Controller*

Bagian *visual servo controller* berfungsi mengubah informasi visual menjadi koreksi gerak Kartesian robot. Posisi *cut-point* aktual dibandingkan dengan *desired point* pada koordinat (387, 36) piksel sehingga diperoleh *image error* pada arah u dan v . Error tersebut digunakan bersama *empirical Jacobian* pada persamaan (2.32) untuk menentukan koreksi pada sumbu Y dan Z, sedangkan gerakan pendekatan pada sumbu X ditentukan berdasarkan perubahan karakteristik *instance mask*. Koreksi dari kedua bagian kemudian digabungkan menjadi perintah Kartesian $\Delta p = [\Delta X, \Delta Y, \Delta Z]^T$. Proses koreksi dilakukan hingga *cut-point* berada pada daerah toleransi dan kondisi pendekatan target terpenuhi.

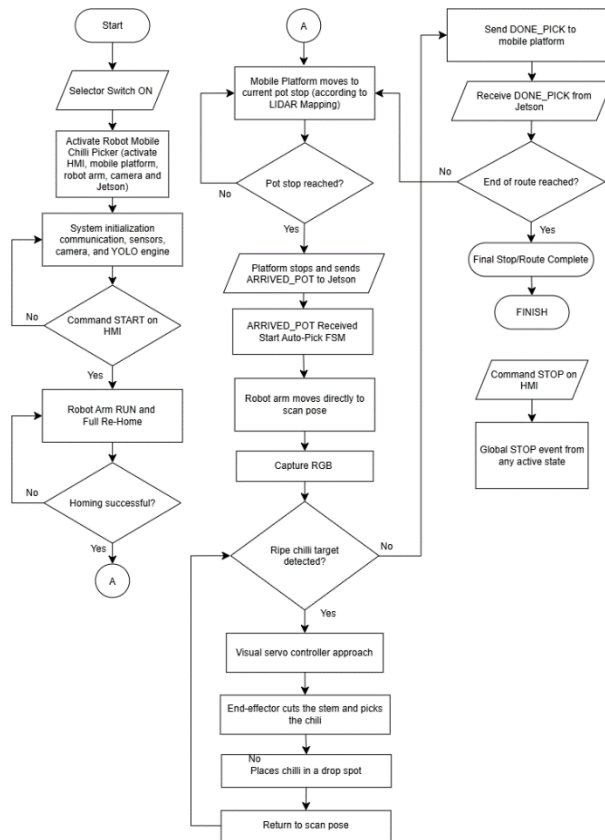
c. Eksekusi Gerak Robot

Bagian eksekusi gerak robot menerima perintah Kartesian ΔX , ΔY , dan ΔZ dari *visual servo controller*. Perintah dikirim dari NVIDIA Jetson Orin Nano menuju STM32 melalui komunikasi UART. STM32 melakukan perhitungan *inverse kinematics* untuk mengubah perintah Kartesian menjadi gerak *joint* pada robot manipulator 5-DOF, kemudian mengendalikan aktuator sesuai hasil perhitungan tersebut. Setelah satu gerakan selesai, STM32 mengirimkan status penyelesaian gerak ke Jetson sehingga kamera memperoleh kondisi target yang baru. Siklus pengamatan, perhitungan koreksi, dan pergerakan robot dilakukan secara bergantian menggunakan mekanisme *step-look* hingga target memenuhi kondisi siap untuk proses pemetikan.

3.2 Flowchart Sistem

3.2.1 Sistem Keseluruhan

Flowchart sistem secara keseluruhan menjelaskan proses kinerja sistem yang terstruktur, dimulai dari tahap awal hingga proses akhir. Flowchart sistem keseluruhan ini ditampilkan pada **Gambar 3.3**.



Gambar 3.3 Flowchart Sistem Keseluruhan

Flowchart di atas menggambarkan alur kerja Chimi-Bot, sistem terintegrasi *mobile platform* dan *robot arm* untuk proses pemetikan cabai otomatis. Proses dimulai ketika *selector switch* diaktifkan, kemudian seluruh perangkat utama seperti *mobile platform*, *robot arm*, kamera, Jetson, serta HMI mulai bekerja. Sistem melakukan inisialisasi komunikasi, sensor, kamera, dan *YOLO engine*. Setelah operator memberikan perintah START melalui HMI, *robot arm* masuk ke kondisi *RUN* dan melakukan proses *full re-home* untuk memastikan posisi awal robot sesuai dengan referensi sistem. Jika proses *homing* belum berhasil, proses tersebut akan diulangi. Setelah *homing* berhasil, sistem melanjutkan ke tahap navigasi *mobile platform*.

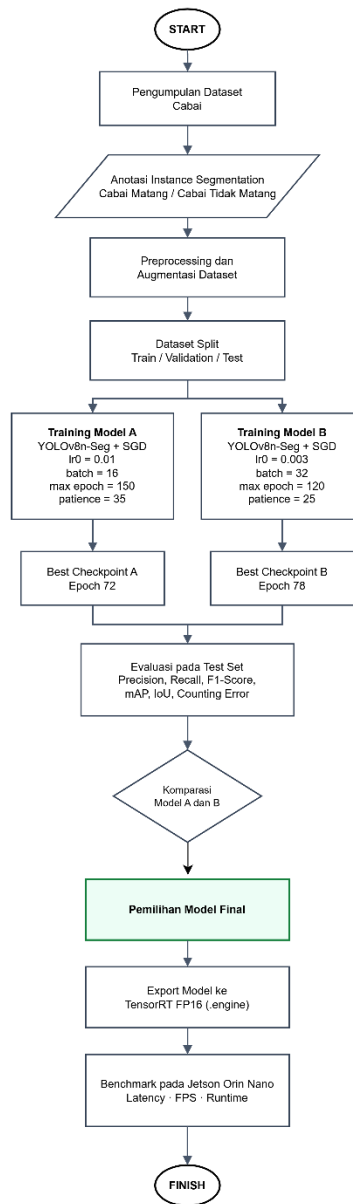
Mobile platform kemudian bergerak menuju setiap pot stop berdasarkan hasil pemetaan LiDAR. Ketika titik pemberhentian belum tercapai, *mobile platform* terus melakukan navigasi hingga mencapai posisi yang telah ditentukan. Setelah sampai, *mobile platform* berhenti dan mengirimkan

status *ARRIVED_POT* ke Jetson sebagai pemicu dimulainya *Auto-Pick FSM*. *Robot arm* selanjutnya bergerak menuju *scan pose*, kemudian kamera mengambil citra RGB untuk mendeteksi keberadaan cabai matang. Jika cabai matang terdeteksi, sistem menjalankan *visual servo controller* untuk mengarahkan *end-effector* menuju posisi *cut-point* target berdasarkan informasi visual. Setelah kondisi posisi pemetikan terpenuhi, *end-effector* melakukan pemotongan dan pengambilan cabai, kemudian hasil pemetikan ditempatkan pada lokasi penampungan sebelum robot kembali ke *scan pose*. Apabila tidak terdapat cabai matang yang dapat diproses, tahapan pemetikan dilewati sehingga sistem dapat melanjutkan ke titik berikutnya.

Setelah proses pada satu pot selesai, Jetson mengirimkan sinyal *DONE_PICK* kepada *mobile platform* sebagai tanda bahwa *robot arm* telah menyelesaikan tugas pada posisi tersebut. *Mobile platform* menerima sinyal tersebut dan memeriksa apakah masih terdapat titik pemberhentian berikutnya pada rute. Jika masih ada, *mobile platform* bergerak menuju *pot stop* selanjutnya dan siklus navigasi–deteksi–*Visual Servoing*–pemetikan diulangi. Jika seluruh rute telah diselesaikan, *mobile platform* berhenti pada posisi akhir dan sistem memasuki kondisi *FINISH*. Selain itu, perintah *STOP* dari HMI berfungsi sebagai *global stop*, sehingga operator dapat menghentikan proses dari kondisi aktif.

3.2.2 Training Model

Flowchart ini menjelaskan aliran untuk menghasilkan model pendeteksian dari tahapan awal. Berikut flowchart pada **Gambar 3.4**.



Gambar 3.4 flowchart pelatihan model

Flowchart **Gambar 3.4** menunjukkan tahapan pelatihan dan pemilihan model YOLOv8n-Seg untuk segmentasi cabai matang dan tidak matang. Proses dimulai dengan pengumpulan citra cabai dan pemberian anotasi *instance segmentation* pada dua kelas, yaitu Cabai_Matang dan Cabai_TidakMatang, menggunakan dataset yang dikelola melalui Roboflow. Dataset kemudian dibagi menjadi 7.629 citra *training*, 441 citra *validation*, dan 535 citra test. Seluruh citra digunakan dengan

ukuran masukan 640×640 piksel. Pada proses *training* juga diterapkan augmentasi ringan berupa perubahan *hue*, saturasi, dan *brightness*, rotasi hingga 8° , translasi 5%, *scaling* 10%, serta *horizontal flip*. Augmentasi seperti *mosaic*, *mixup*, *vertical flip*, dan *copy-paste* tidak digunakan untuk menjaga bentuk dan karakteristik warna cabai agar tetap sesuai dengan kondisi sebenarnya.

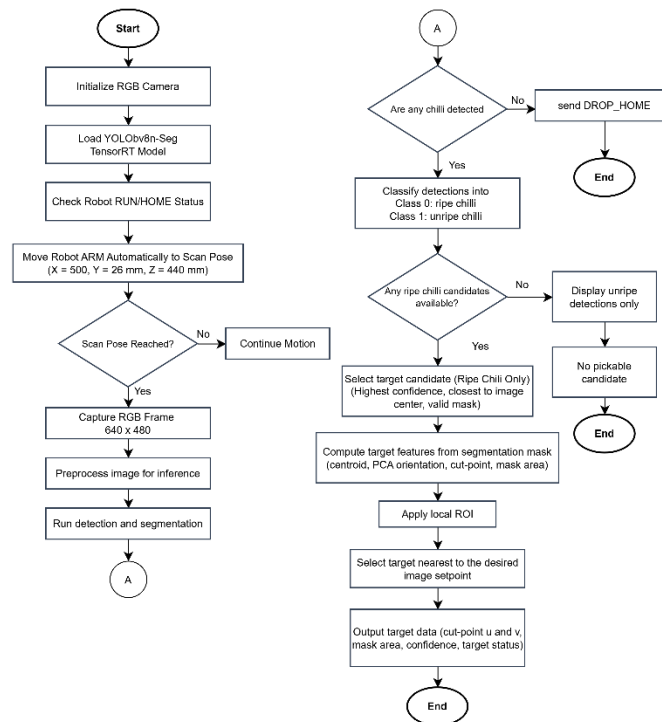
Tahap berikutnya adalah pelatihan dua konfigurasi YOLOv8n-Seg pretrained menggunakan GPU NVIDIA A100-SXM4 40 GB pada Google Colab. Model A menggunakan *optimizer* SGD dengan *learning rate* awal 0,01, *batch size* 16, maksimum 150 *epoch*, dan *early stopping patience* 35. *Training* Model A berhenti pada *epoch* ke-107 karena tidak terjadi peningkatan performa dalam 35 *epoch* terakhir, sedangkan *best checkpoint* yang disimpan oleh Ultralytics diperoleh pada *epoch* 72. Model B menggunakan *optimizer* SGD dengan *learning rate* awal 0,0003, *batch size* 32, maksimum 120 *epoch*, dan *patience* 25. Model ini menggunakan *cosine learning-rate scheduler*, *Automatic Mixed Precision* (AMP), momentum 0,937, *weight decay* 0,0005, dan *seed* 42. *Training* Model B berhenti pada *epoch* ke-103 dan menghasilkan *best checkpoint* pada *epoch* 78. Mekanisme *early stopping* digunakan agar *training* tidak dilanjutkan ketika performa validasi sudah tidak mengalami peningkatan.

Kedua *best checkpoint* selanjutnya dievaluasi menggunakan *test set* yang berisi 535 citra dengan total 952 objek *ground truth*. Evaluasi dilakukan terhadap hasil deteksi dan segmentasi menggunakan parameter *precision*, *recall*, Skor F1, mAP50, mAP50–95, IoU, serta *counting error*, sehingga pemilihan model tidak hanya didasarkan pada *loss* selama *training*. Model dengan performa yang paling sesuai kemudian ditetapkan sebagai model final. Setelah tahap pelatihan dan evaluasi di Google Colab selesai, model final dikonversi ke format TensorRT FP16 (*.engine*) untuk implementasi pada NVIDIA Jetson

Orin Nano. Model hasil optimasi tersebut kemudian diuji pada perangkat Jetson berdasarkan *latency*, FPS, dan *runtime* untuk memastikan model dapat digunakan pada sistem segmentasi cabai secara *real-time*.

3.2.3 Sistem Deteksi Cabai

Flowchart pendeteksian cabai menggambarkan alur kerja sistem dalam melakukan proses deteksi objek cabai pada citra secara terstruktur, mulai dari tahap inialisasi sistem hingga proses identifikasi objek yang terdeteksi. Flowchart dapat dilihat pada **Gambar 3.5**.



Gambar 3.5 Flowchart Sistem Deteksi dan Pemilihan Target Cabai Matang

Proses flowchart **Gambar 3.5**, Flowchart tersebut menjelaskan alur deteksi dan pemilihan target cabai sebelum data diteruskan ke pengendali *Visual Servoing*. Proses dimulai dengan inialisasi kamera RGB dan pemuatan model YOLOv8n-Seg TensorRT. Setelah itu, sistem memeriksa status RUN dan HOME robot. Jika robot siap, *robot arm* bergerak otomatis menuju *scan pose*. Setelah posisi *scan* tercapai, kamera mengambil citra RGB

berukuran 640×480 piksel, kemudian citra diproses melalui tahap *preprocessing* sebelum dilakukan deteksi dan segmentasi.

Hasil deteksi kemudian diperiksa untuk mengetahui keberadaan cabai. Jika tidak ada cabai yang terdeteksi, sistem mengirim perintah *DROP_HOME* agar *robot arm* kembali ke posisi *home* sebelum siklus berakhir atau sebelum AGV bergerak ke titik berikutnya. Hal ini dilakukan karena pada *scan pose* posisi lengan lebih menjulur, sehingga ketika AGV bergerak kondisi mekanik menjadi kurang stabil dan lebih mudah mengalami getaran. Posisi *home* digunakan sebagai konfigurasi yang lebih aman dan stabil selama perpindahan AGV. Jika cabai terdeteksi, hasilnya diklasifikasikan menjadi *Class 0: ripe chilli* dan *Class 1: unripe chilli*. Cabai belum matang hanya ditampilkan sebagai hasil deteksi dan tidak digunakan sebagai target pemetikan. Jika tidak terdapat kandidat cabai matang, sistem menghasilkan kondisi *no pickable candidate*. Sebaliknya, jika terdapat cabai matang, sistem memilih kandidat target dengan mempertimbangkan *confidence*, kedekatan terhadap area target citra, serta validitas *mask* segmentasi.

Setelah kandidat terpilih, sistem menghitung fitur target dari *mask* segmentasi berupa *centroid*, orientasi PCA, *cut-point*, dan luas *mask*. Selanjutnya, *local ROI* digunakan untuk membatasi area pencarian dan membantu menjaga konsistensi target. Dari kandidat yang tersedia, target akhir dipilih berdasarkan posisi yang paling dekat terhadap *desired image setpoint*. Output dari tahap ini berupa koordinat *cut-point*, luas *mask*, nilai *confidence*, dan status target. Data tersebut kemudian digunakan sebagai masukan untuk proses *Visual Servoing* pada tahap berikutnya.

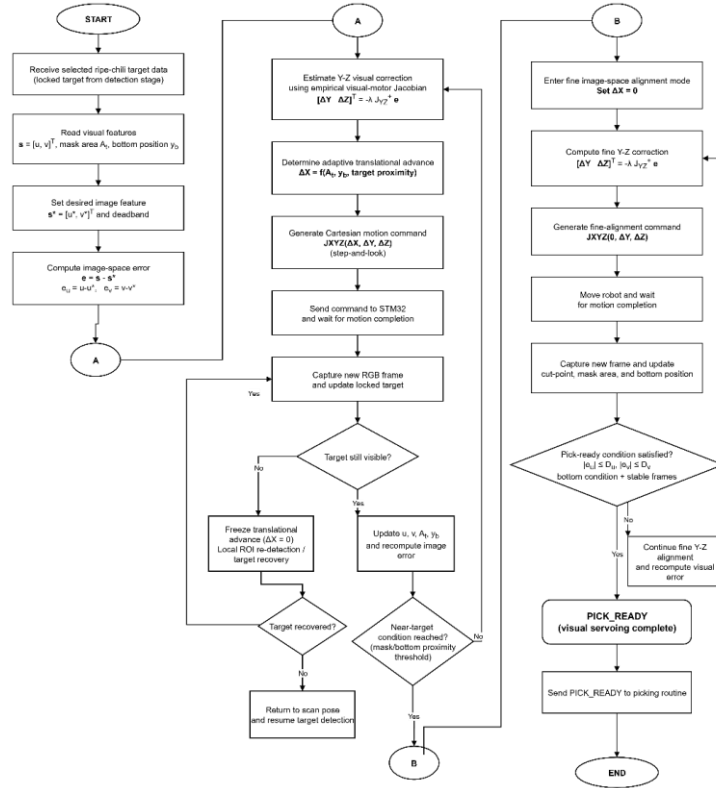
Flowchart tersebut menjelaskan proses deteksi dan pemilihan target sebelum masuk ke tahap *Visual Servoing*. Setelah sistem aktif, kamera RGB dan model YOLOv8n-Seg diinisialisasi, kemudian status RUN dan HOME robot diperiksa. *Robot arm* lalu bergerak otomatis ke *scan pose*. Posisi ini digunakan khusus untuk proses pengamatan tanaman. Setelah *scan pose* tercapai, kamera mengambil citra RGB 640×480 piksel, melakukan

preprocessing, lalu menjalankan deteksi dan segmentasi dengan YOLOv8n-Seg.

Hasil deteksi dibagi menjadi dua kelas, yaitu cabai matang sebagai *pickable target* dan cabai belum matang sebagai *detection only*. Jika tidak terdapat kandidat cabai matang, sistem tidak menjalankan *Visual Servoing*. Robot kemudian dikembalikan ke posisi *home* menggunakan perintah DROP_HOME sebelum AGV bergerak ke titik berikutnya. Hal ini dilakukan karena pada *scan pose* posisi lengan lebih menjulur, sehingga ketika AGV bergerak kondisi mekanik menjadi kurang stabil dan lebih mudah mengalami getaran. Posisi *home* digunakan sebagai konfigurasi yang lebih aman dan stabil selama perpindahan AGV.

Jika terdapat satu atau lebih cabai matang, sistem menghitung fitur hasil segmentasi berupa *bounding box*, luas *mask*, *centroid*, posisi atas dan bawah objek, *confidence*, serta PCA *cut-point*. Pemilihan target dilakukan berdasarkan kandidat cabai matang yang memiliki *cut-point* paling dekat terhadap *desired image setpoint*, dengan nilai *confidence* sebagai faktor pendukung. Setelah target dipilih, *local ROI* digunakan untuk membatasi area pencarian dan membantu mempertahankan target yang sama. Data akhir berupa koordinat *cut-point* (u,v), luas *mask*, *confidence*, dan status target kemudian diteruskan ke pengendali *Visual Servoing*.

3.2.4 Visual Servoing and Approach Control



Gambar 3.6 Flowchart Visual Servoing and Approach Control

Flowchart *Visual Servoing* menunjukkan proses pengendalian robot setelah target cabai matang berhasil dipilih pada tahap deteksi. Sistem menerima data target yang sudah dikunci, kemudian membaca informasi visual berupa posisi *cut-point* pada sumbu citra u dan v , luas *mask* A_t , serta posisi bagian bawah *mask* y_b

. Posisi *cut-point* dibandingkan dengan *desired image setpoint* untuk mengetahui besar penyimpangan target pada citra. Nilai penyimpangan inilah yang digunakan sebagai acuan utama untuk menentukan arah koreksi gerak robot.

Pada tahap *adaptive Cartesian approach*, robot bergerak mendekati target dengan gerakan translasi pada sumbu X, sementara sumbu Y dan Z juga dikoreksi berdasarkan error posisi target pada citra. Besar langkah X tidak dibuat konstan, tetapi disesuaikan dengan kondisi visual seperti luas *mask* dan posisi bagian bawah objek, sehingga langkah gerak akan berkurang ketika

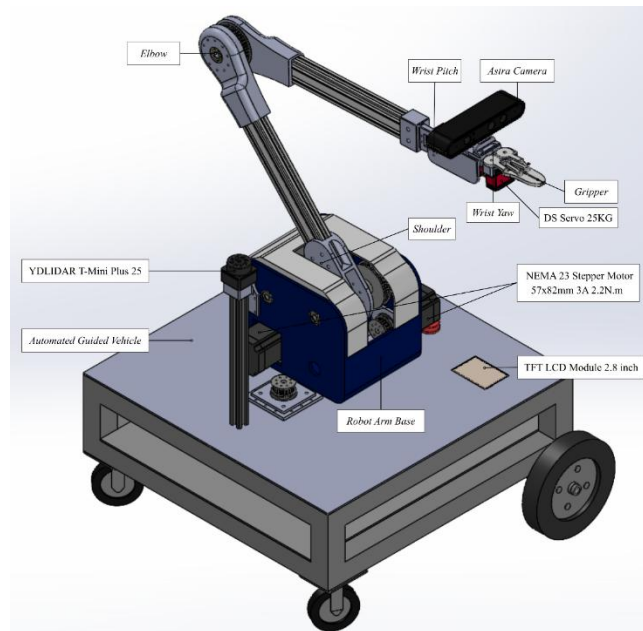
robot semakin dekat dengan target. Koreksi pada sumbu Y dan Z diperoleh menggunakan *empirical visual-motor Jacobian* yang sebelumnya diperoleh dari pengujian gerakan robot. Setelah perintah gerak dikirim ke STM32 dan robot selesai bergerak, kamera kembali mengambil citra baru agar posisi target dapat diperbarui. Proses ini berlangsung berulang dengan konsep *step-and-look*, yaitu setiap gerakan selalu diikuti pengamatan ulang sebelum menentukan koreksi berikutnya.

Selama proses *approach*, sistem terus memeriksa apakah target yang sama masih dapat terdeteksi. Jika target masih terlihat, posisi *cut-point*, luas *mask*, dan posisi bawah *mask* diperbarui untuk menghitung koreksi selanjutnya. Jika target hilang dari citra, gerakan maju dihentikan sementara dan sistem melakukan pencarian ulang pada *local ROI* untuk mencoba menemukan kembali target yang sebelumnya dikunci. Jika target berhasil ditemukan, proses *Visual Servoing* dilanjutkan. Namun, jika target tidak dapat ditemukan kembali dalam batas toleransi yang ditentukan, robot dikembalikan ke *scan pose* agar proses deteksi target dapat dimulai kembali.

Ketika kondisi visual menunjukkan bahwa target telah berada cukup dekat, sistem masuk ke tahap *fine image-space alignment*. Pada tahap ini gerakan sumbu X dihentikan dan koreksi difokuskan pada sumbu Y dan Z untuk memperkecil error posisi *cut-point* terhadap *setpoint*. Setelah setiap koreksi, citra diambil kembali dan kondisi target diperiksa. Sistem akan menetapkan status PICK_READY apabila posisi *cut-point* telah berada di dalam *deadband*, posisi bawah *mask* telah memenuhi kondisi kedekatan target, dan hasil *alignment* stabil selama beberapa frame. Setelah kondisi tersebut terpenuhi, proses *Visual Servoing* dinyatakan selesai dan status PICK_READY diteruskan ke tahap pemetikan.

3.3 Desain 3D Perancangan

Visualisasi desain 3D dari alat tugas akhir ini dapat dilihat pada **Gambar 3.7**. Desain alat ini merupakan hasil gabungan antara rancangan *robot arm* 5-DOF dan *automated guided vehicle* (AGV) sebagai *mobile platform*.

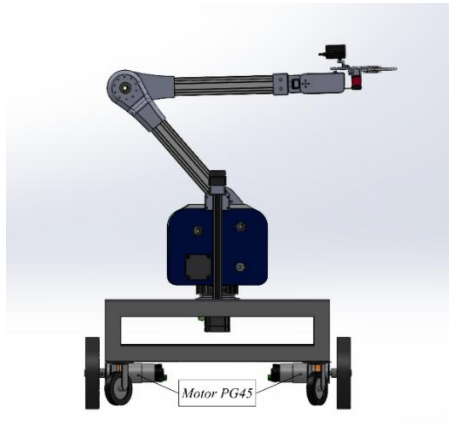


Gambar 3. 7 Desain 3D Robot Mobile Chilli Picker

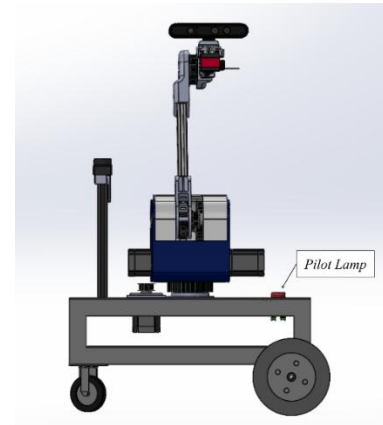
Robot arm memiliki beberapa bagian utama, yaitu *robot arm base*, *shoulder*, *elbow*, *wrist pitch*, dan *wrist yaw*. Pada bagian ujung robot dipasang *gripper* sebagai *end-effector* untuk melakukan proses pemetikan cabai.

Pada bagian *wrist* dipasang Astra Camera dengan konfigurasi *eye-in-hand*, sehingga kamera bergerak mengikuti pergerakan ujung robot dan digunakan sebagai sumber informasi visual untuk mendeteksi serta menentukan posisi target cabai. Penggerak robot menggunakan NEMA 23 *stepper motor* pada beberapa *joint* dan DS Servo 25KG pada mekanisme ujung robot. Sementara itu, pada *mobile platform* terdapat YDLIDAR T-Mini Plus 25 sebagai sensor navigasi serta TFT LCD 2,8 inci sebagai bagian dari antarmuka sistem.

Desain tersebut menunjukkan integrasi antara sistem navigasi *mobile platform*, sistem penglihatan berbasis kamera, dan mekanisme *robot arm* dalam satu perangkat. Pada tugas akhir ini, bagian yang menjadi fokus perancangan adalah *robot arm* 5-DOF beserta sistem kendali dan pengolahan visualnya, sedangkan *mobile platform* digunakan sebagai perangkat pembawa *robot arm* untuk berpindah menuju lokasi tanaman.



Gambar 3.8 Alat Tampak Samping



Gambar 3.9 Alat Tampak Depan

Pada **Gambar 3.8** dan **Gambar 3.9**, diperlihatkan tampak samping dan tampak depan dari *Chilli-Bot* untuk menunjukkan beberapa komponen yang tidak terlihat secara jelas pada tampilan isometrik sebelumnya. Pada tampak samping dapat terlihat motor PG45 yang digunakan sebagai penggerak roda pada *mobile platform*. Posisi motor berada pada bagian bawah rangka sehingga tidak mengganggu area pemasangan *robot arm* serta menjaga susunan mekanik tetap ringkas.

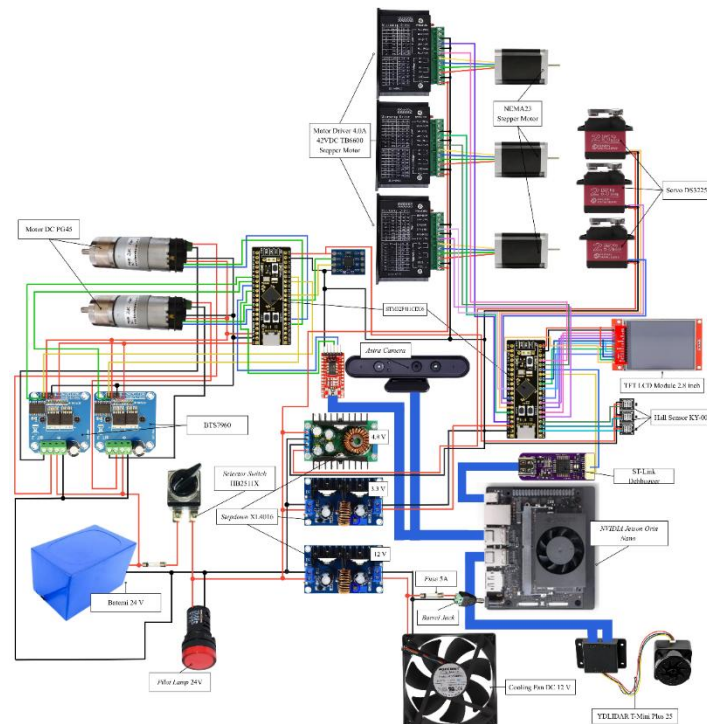
Sementara itu, pada tampak depan dapat terlihat *pilot lamp* yang ditempatkan pada bagian depan *mobile platform*. *Pilot lamp* digunakan sebagai indikator visual untuk menunjukkan kondisi atau status kerja sistem saat alat sedang dioperasikan. Tampilan samping dan depan ini juga memperlihatkan posisi *robot arm* terhadap rangka AGV, sehingga dapat diketahui bahwa *robot arm* dipasang pada bagian tengah platform untuk memberikan ruang gerak yang cukup selama proses pencarian target dan pemetikan cabai.

3.4 Cara Pembuatan

a. Perancangan Elektrikal Keseluruhan

Rangkaian elektrikal keseluruhan pada sistem robot pemanen cabai menggambarkan hubungan dan konektivitas antar komponen utama yang membentuk sistem, mulai dari sumber catu daya, unit pemrosesan, perangkat sensor, hingga aktuator dan perangkat antarmuka pengguna. Rangkaian ini

menunjukkan bagaimana aliran daya listrik dan jalur komunikasi data terdistribusi ke setiap komponen sehingga seluruh perangkat dapat bekerja secara terintegrasi dalam melakukan proses pengambilan citra, pengolahan data, pengendalian aktuator, serta interaksi dengan pengguna.



Gambar 3.10 Rangkaian Elektrikal Keseluruhan Sistem

Berdasarkan **Gambar 3.10**, sumber daya utama sistem berasal dari baterai 24 V. Tegangan dari baterai dialirkan melalui jalur pengaman dan *selector switch* sebagai sakelar utama untuk mengaktifkan atau memutus catu daya sistem. Pada jalur tersebut juga dipasang *pilot lamp* 24 V sebagai indikator bahwa sistem sedang memperoleh daya. Tegangan 24 V kemudian didistribusikan ke beberapa bagian sesuai kebutuhan masing-masing komponen. Perangkat yang membutuhkan tegangan lebih rendah menggunakan modul *step-down* XL4016, yang menghasilkan beberapa level tegangan, yaitu 12 V, 4,8 V, dan 3,3 V. Jalur 12 V digunakan antara lain untuk mencatu NVIDIA Jetson Orin Nano melalui *barrel jack* dan fuse 5 A, serta untuk *cooling fan* 12 V. Fuse 5 A pada jalur Jetson berfungsi sebagai proteksi apabila terjadi arus berlebih pada rangkaian komputasi utama.

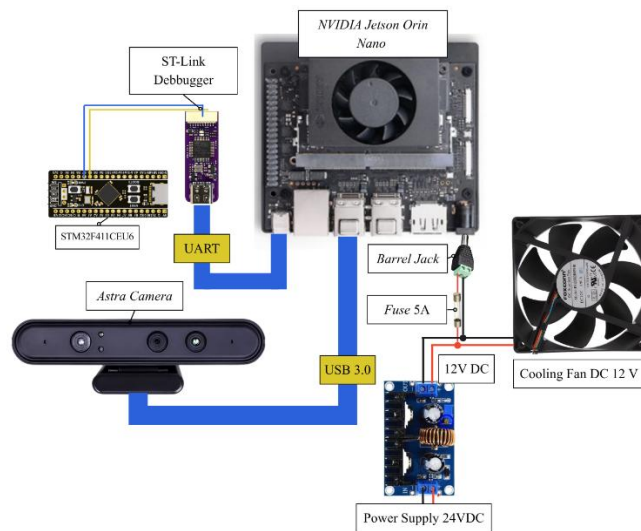
Sistem penggerak terdiri dari bagian *mobile platform* dan *robot arm*. Pada *mobile platform*, dua motor DC PG45 dikendalikan menggunakan dua modul BTS7960. Mikrokontroler STM32F411CEU6 memberikan sinyal kendali kepada *motor driver* untuk mengatur arah dan kecepatan putaran motor sehingga *mobile platform* dapat bergerak sesuai perintah navigasi. Pada *robot arm*, tiga NEMA 23 *stepper motor* digerakkan menggunakan tiga *stepper motor driver* TB6600 4 A, sedangkan bagian *wrist* dan *gripper* menggunakan servo DS3225. Mikrokontroler STM32 menghasilkan sinyal kendali ke masing-masing *driver* dan servo berdasarkan perintah posisi yang diterima. Sistem *robot arm* juga dilengkapi Hall Sensor KY-003 sebagai sensor referensi posisi atau *homing*, serta TFT LCD 2,8 inci untuk menampilkan informasi kondisi sistem.

Pada bagian pemrosesan utama digunakan NVIDIA Jetson Orin Nano sebagai komputer yang menjalankan pengolahan citra dan algoritma kendali tingkat tinggi. Astra Camera terhubung ke Jetson dengan USB 3.0 dengan tegangan 5V untuk memperoleh citra RGB sebagai masukan YOLOv8n-Seg serta *Visual Servoing*. Pada implementasi ini informasi kedalaman kamera tidak digunakan sebagai masukan utama kendali, karena posisi target ditentukan berdasarkan informasi citra dua dimensi, *cut-point*, dan hasil segmentasi. Jetson kemudian memberikan perintah gerak kepada STM32 untuk diterjemahkan menjadi aktuasi pada *robot arm*. Selain kamera, YDLIDAR T-Mini Plus 25 terhubung ke sistem untuk mendukung proses pemetaan dan navigasi *mobile platform*, sedangkan ST-Link Debugger digunakan untuk proses pemrograman dan *debugging* mikrokontroler STM32.

b. Skema Rangkaian Elektrikal Sistem Pengolahan Citra

Skema rangkaian elektrikal yang digunakan pada sistem pengolahan citra ditampilkan pada **Gambar 3.11**. Rangkaian ini menunjukkan hubungan antara sumber daya, kamera sebagai perangkat masukan, NVIDIA Jetson Orin Nano sebagai unit pemrosesan utama, serta jalur komunikasi dengan mikrokontroler STM32. Secara umum, rangkaian dibagi menjadi empat

bagian, yaitu blok catu daya (*power*), blok masukan (*input*), blok kendali (*controller*), dan blok komunikasi keluaran (*output*).



Gambar 3.11 Skema Rangkaian Elektrikal Sistem Pengolahan Citra

Blok *power* menunjukkan suplai daya untuk **NVIDIA Jetson Orin Nano**. Sumber tegangan berasal dari sistem catu daya *Chilli-Bot* yang kemudian diturunkan menjadi **12 VDC** menggunakan modul *step-down*. Tegangan 12 V tersebut diteruskan melalui **fuse 5 A** sebagai pengaman arus lebih, kemudian masuk ke Jetson melalui *barrel jack*. Blok *input* terdiri dari **Astra Camera** yang terhubung langsung ke Jetson melalui antarmuka USB. Kamera digunakan untuk memperoleh citra RGB yang selanjutnya diproses pada Jetson untuk proses deteksi dan segmentasi cabai.

Pada blok *controller*, NVIDIA Jetson Orin Nano berfungsi sebagai unit pemrosesan utama yang menjalankan akuisisi citra, inferensi **YOLOv8n-Seg**, penentuan *cut-point*, serta algoritma *Visual Servoing*. Hasil pengolahan tersebut kemudian diteruskan melalui komunikasi serial USB kepada **STM32** sebagai sistem kendali robot. STM32 menerima perintah gerak dari Jetson dan mengirimkan kembali informasi kondisi robot, seperti posisi hasil *forward kinematics* dan status sistem. Rangkaian aktuator, *motor driver*, servo, serta

penggerak *mobile platform* berada pada bagian sistem robot dan tidak menjadi fokus perancangan elektrikal pada tugas akhir ini.

Tabel 3. 2 Konfigurasi I/O

Komponen/Komunikasi	I/O NVIDIA Jetson Orin Nano	I/O STM32
Catu daya Jetson	<i>Barrel jack</i> 12 VDC	–
Astra Camera	Port USB	–
Komunikasi Jetson–STM32	Port USB / Serial	USB Serial
UART/Serial Device	/dev/ttyACM0	USB CDC/Serial
<i>Cooling fan</i>	Catu daya 12 V	–

3.5 Teknik Fabrikasi

Tahapan dalam perancangan dan pembuatan sistem visi untuk deteksi, segmentasi, dan penentuan posisi target cabai merah matang menggunakan NVIDIA Jetson Orin Nano yang kemudian diimplementasikan pada robot lengan, terdiri dari beberapa bagian utama sebagai berikut.

3.5.1 Alat dan Bahan

Sistem pada tugas akhir ini membutuhkan berbagai peralatan dan bahan yang mendukung proses akuisisi citra, deteksi dan segmentasi objek, penentuan titik target, serta proses *Visual Servoing* secara *real-time*. Alat dan bahan yang digunakan ditunjukkan pada **Tabel 3.3**.

Tabel 3.3 Alat dan Bahan

Alat dan bahan	Fungsi
Baterai 24V	Sumber catu daya utama sistem robot <i>Single-board computer</i> sebagai unit komputasi utama untuk menjalankan pemrosesan citra, model YOLOv8n-Seg, penentuan titik target, serta sistem <i>Visual Servoing</i> .
NVIDIA Jetson Orin Nano	
Orbbec Astra Pro Camera	Kamera <i>eye-in-hand</i> untuk memperoleh citra RGB objek cabai sebagai masukan sistem deteksi dan segmentasi.

<i>Stepdown</i> XL4016	Tegangan dari 24V menjadi 12V
<i>Cooling fan</i> 12V	Pendingin untuk menjaga suhu operasi Jetson Orin Nano.
Fuse 10A	Pengaman dari arus berlebih untuk melindungi Jetson Orin Nano.
STM32F411CEU6	Mikrokontroler untuk mengendalikan aktuator dan menerima data dari Jetson

3.5.2 Perancangan *Image Processing*

Perancangan sistem *image processing* dengan YOLOv8n pada alat tugas akhir ini meliputi beberapa tahapan proses yang masing-masing dijelaskan dibawah ini:

a. Akuisisi dan Persiapan Dataset

Dataset yang digunakan merupakan citra cabai merah besar yang diperoleh dari pengambilan gambar secara langsung. Pengambilan data dilakukan dengan memberikan variasi kondisi agar model dapat mengenali objek ketika digunakan pada kondisi sebenarnya.

Variasi data meliputi perbedaan posisi cabai, orientasi objek, jarak terhadap kamera, kondisi pencahayaan, serta jumlah cabai yang terdapat pada satu citra. Dataset juga mencakup cabai dengan tingkat kematangan berbeda, yaitu cabai matang dan cabai tidak matang.

Data awal yang dikumpulkan dan dimasukkan ke dalam workspace Roboflow berjumlah 3.519 citra. Contoh citra dataset yang digunakan dapat dilihat pada **Gambar 3.12**.

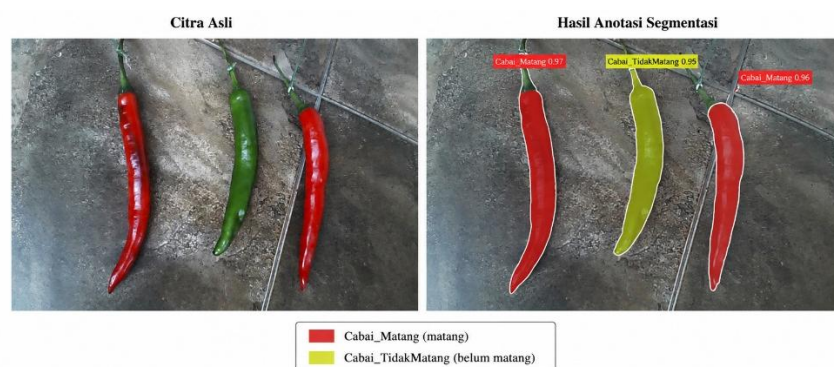


Gambar 3. 12 Sampel Data

b. Anotasi Dataset

Anotasi dataset merupakan proses pemberian label pada objek yang terdapat di dalam citra. Karena model yang digunakan adalah YOLOv8n-Seg, proses anotasi dilakukan menggunakan metode *instance segmentation*, yaitu dengan memberikan area *mask* yang mengikuti bentuk masing-masing objek cabai.

Proses anotasi dilakukan menggunakan platform Roboflow. Setiap objek cabai diberi *mask* sesuai dengan batas bentuk objek dan kemudian diberikan label berdasarkan tingkat kematangannya. Contoh proses anotasi dapat dilihat pada **Gambar 3.13**.



Gambar 3.13 Proses Anotasi

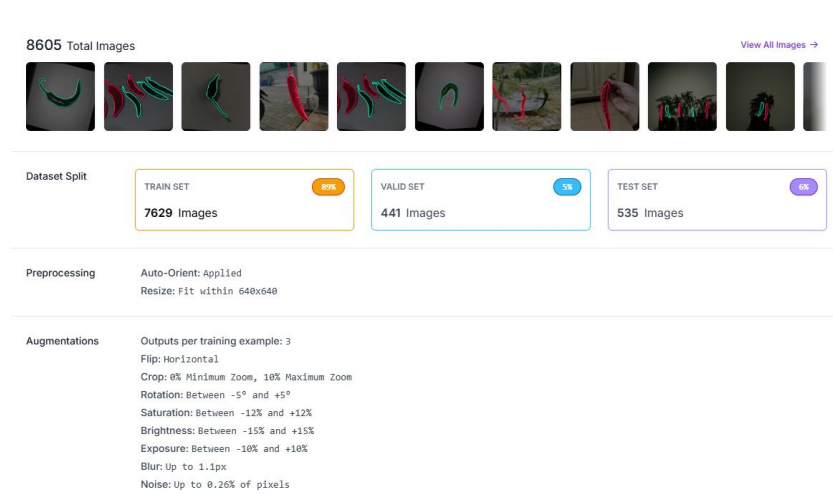
Pada penelitian ini digunakan dua kelas objek, rincian hasil anotasi tersebut pada **Tabel 3.4**.

Tabel 3.4 Jumlah Anotasi Gambar

Kelas	Jumlah
Cabai Matang	4.366
Cabai Tidak Matang	4.469

c. *Dataset split*

Dataset yang telah melalui proses anotasi kemudian dibagi menjadi tiga bagian, yaitu data pelatihan (*training set*), data validasi (*validation set*), dan data pengujian (*testing set*). Pembagian dataset ini bertujuan agar model dapat dilatih menggunakan sebagian besar data, serta diuji menggunakan data yang belum pernah dilihat sebelumnya sehingga performa model dapat dievaluasi secara objektif.



Gambar 3. 14 Tampilan *Dataset split* Roboflow

Pada tugas akhir ini, dataset dibagi dengan rasio 89% untuk data pelatihan, 5% untuk data validasi, dan 6% untuk data pengujian. Data pelatihan digunakan untuk melatih model dalam mengenali pola dan karakteristik objek cabai pada citra. Data validasi digunakan untuk memantau kinerja model selama proses pelatihan serta membantu dalam melakukan penyesuaian parameter model. Sementara itu, data pengujian digunakan untuk mengevaluasi kemampuan model dalam mendeteksi

objek pada data baru yang tidak digunakan selama proses pelatihan. Pembagian dataset tersebut dapat dilihat pada **Tabel 3.5**.

Tabel 3.5 Pembagian Dataset

Total Data	Training set (89%)	Validation set (5%)	Testing set (6%)
13.605	7629 Images	441 Images	535 Images

d. *Preprocessing* Data

Tahap *preprocessing* dilakukan untuk menyeragamkan data sebelum digunakan sebagai masukan pada model YOLOv8n-Seg. *Preprocessing* dilakukan melalui Roboflow dengan konfigurasi *Auto-orient* dan *Resize*.

1) *Auto-orient*

Fitur *Auto-orient* digunakan untuk menyesuaikan orientasi citra berdasarkan informasi orientasi yang terdapat pada metadata gambar. Proses ini dilakukan agar seluruh citra memiliki orientasi yang sesuai sebelum diproses lebih lanjut.

2) *Resize* Citra

Citra pada dataset disesuaikan dengan ukuran masukan model menggunakan metode Fit within 640×640 pixels. Metode ini menyesuaikan ukuran citra sehingga dapat dimasukkan ke dalam dimensi maksimum 640×640 piksel dengan tetap mempertahankan proporsi citra.

Penggunaan ukuran input 640 piksel dipilih karena memberikan keseimbangan antara detail informasi objek dan kebutuhan komputasi pada proses pelatihan serta inferensi model.

3) *Normalisasi Nilai Piksel*

Pada saat citra digunakan sebagai input jaringan YOLOv8n-Seg, nilai piksel citra yang berada pada rentang 0–255 diskalakan menjadi rentang 0–1. Proses ini dilakukan dengan membagi nilai

piksel dengan 255 sehingga data citra berada pada skala yang lebih beragam sebelum diproses oleh jaringan.

d. *Data augmentation*

Data augmentation digunakan untuk meningkatkan variasi citra pada dataset tanpa melakukan pengambilan data baru. Proses ini menghasilkan variasi dari citra yang telah tersedia sehingga model memperoleh lebih banyak kombinasi kondisi selama proses pelatihan.

Pada dataset ini, konfigurasi augmentasi diterapkan melalui Roboflow dengan menghasilkan hingga 3 output untuk setiap contoh data *training*. Konfigurasi augmentasi yang digunakan ditunjukkan pada **Tabel 3.6**.

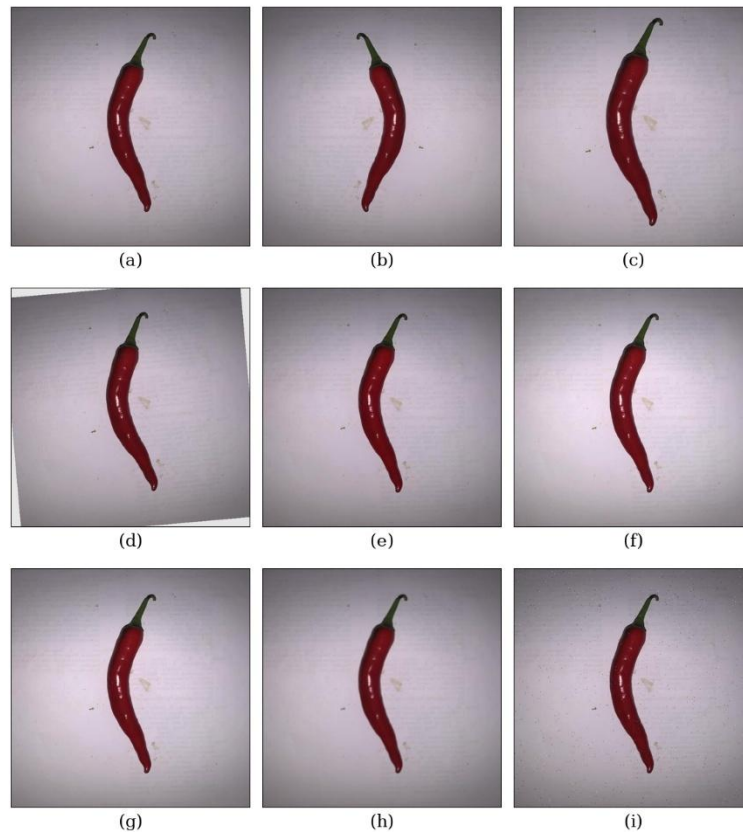
Tabel 3.6 Konfigurasi *Data augmentation*

Augmentasi	Konfigurasi
Flip	Horizontal
Crop	Minimum Zoom 0%, Maximum Zoom 16%
Rotation	-5° hingga $+5^{\circ}$
Saturation	-12% hingga $+12\%$
Brightness	-15% hingga $+15\%$
Exposure	-10% hingga $+10\%$
Blur	Hingga 1,1 px
Noise	Hingga 0,26% piksel
Outputs per <i>training</i> example	3

Variasi rotasi digunakan untuk mensimulasikan perubahan orientasi cabai terhadap kamera. Perubahan *brightness*, *exposure*, dan *saturation* digunakan untuk memberikan variasi terhadap kondisi pencahayaan dan warna objek. Sementara itu, *blur* dan *noise* digunakan untuk memberikan variasi kualitas citra yang mungkin muncul ketika sistem digunakan pada kondisi sebenarnya.

Horizontal flip dan *crop* digunakan untuk menambah variasi posisi serta skala objek di dalam citra. Dengan augmentasi tersebut, jumlah

data pada versi dataset yang digunakan untuk pelatihan meningkat dari dataset awal 3.519 citra menjadi 8.605 citra.



Gambar 3.15 Contoh hasil augmentasi data pada citra cabai: (a) citra asli, (b) *horizontal flip*, (c) *crop*, (d) *rotation*, (e) *saturation*, (f) *brightness*, (g) *exposure*, (h) *blur*, dan (i) *noise*.

e. *Training Model*

Pelatihan model dilakukan menggunakan arsitektur YOLOv8n-Seg. Model ini dipilih karena memiliki ukuran jaringan yang relatif ringan sehingga sesuai untuk diterapkan pada perangkat komputasi *embedded* pada sistem robot, namun tetap memiliki kemampuan deteksi dan segmentasi objek.

Selama proses pelatihan, model mempelajari karakteristik visual kedua kelas cabai berdasarkan citra dan *segmentation mask* yang telah disiapkan. Proses pembelajaran dilakukan dengan

memperbarui bobot jaringan berdasarkan nilai *loss* yang diperoleh pada setiap iterasi. *Training* dilakukan di Google Colab menggunakan GPU A100.

Dalam penelitian ini dilakukan dua konfigurasi pelatihan, yaitu Model A sebagai konfigurasi awal dan Model B sebagai hasil penyesuaian parameter pelatihan. Penggunaan dua konfigurasi dilakukan untuk mengetahui pengaruh perubahan parameter terhadap performa model.

Tabel 3.7 Konfigurasi Pelatihan Model YOLOv8n-Seg

Parameter	Model A	Model B
<i>Model</i>	YOLOv8n-Seg	YOLOv8n-Seg
<i>Optimizer</i>	SGD	SGD
<i>Input size</i>	640×640	640×640
<i>Initial learning rate (lr_0)</i>	0,010	0,0003
<i>Batch size</i>	16	32
<i>Target epoch</i>	150	120
<i>Patience</i>	35	25
<i>Seed</i>	42	42

Berikut penjelasan tiap parameter:

1) *Epoch*

Epoch merupakan satu siklus ketika seluruh data *training* telah diproses oleh model. Pada setiap *epoch*, bobot jaringan diperbarui berdasarkan nilai kesalahan yang diperoleh selama proses pelatihan. Model A dirancang dengan maksimum 150 *epoch*, sedangkan Model B menggunakan maksimum 120 *epoch*. Jumlah tersebut merupakan batas maksimum pelatihan dan tidak selalu berarti seluruh *epoch* harus dijalankan karena digunakan mekanisme *early stopping*.

2) *Batch size*

Batch size merupakan jumlah citra yang diproses sebelum dilakukan pembaruan bobot jaringan. Model A menggunakan *batch size* 16, sedangkan Model B menggunakan *batch size* 32.

Perubahan *batch size* dilakukan untuk memperoleh proses pembelajaran yang lebih stabil sekaligus menyesuaikan penggunaan sumber daya komputasi selama pelatihan.

3) Input Size

Ukuran masukan yang digunakan pada proses pelatihan adalah 640×640 piksel. Ukuran ini disesuaikan dengan konfigurasi *preprocessing* dataset dan ukuran input YOLOv8n-Seg.

4) Optimizer

Optimizer yang digunakan pada kedua model adalah *Stochastic Gradient Descent* (SGD). *Optimizer* berfungsi untuk memperbarui parameter jaringan berdasarkan gradien yang diperoleh selama proses pelatihan.

SGD digunakan secara konsisten pada kedua konfigurasi sehingga pengaruh perubahan beberapa *hyperparameter*, seperti *learning rate* dan *batch size*, dapat diamati.

5) Learning rate

Learning rate menentukan besarnya perubahan bobot jaringan pada setiap proses pembaruan parameter. Model A menggunakan nilai awal *learning rate* sebesar 0,010, sedangkan pada Model B nilainya diturunkan menjadi 0,0003. Penyesuaian tersebut dilakukan untuk memperoleh pembaruan bobot yang lebih terkontrol selama proses pelatihan. Analisis pengaruh perubahan *learning rate* terhadap nilai *loss* dan performa model selanjutnya dibahas pada BAB IV.

6) Early stopping

Untuk mencegah proses pelatihan terus berjalan ketika tidak lagi menghasilkan peningkatan performa yang berarti, digunakan mekanisme *early stopping*. Parameter *patience* menentukan jumlah *epoch* yang dapat dilewati tanpa adanya peningkatan sebelum proses pelatihan dihentikan secara otomatis.

Model A menggunakan *patience* = 35, sedangkan Model B menggunakan *patience* = 25. Dengan mekanisme tersebut, model tidak harus menyelesaikan seluruh target *epoch* apabila performa validasi tidak mengalami peningkatan.

7) Testing Data

Tahap pengujian model dilakukan menggunakan 535 citra *test set* yang terdiri atas sampel cabai matang dan cabai tidak matang dengan variasi kondisi pencahayaan, posisi objek, serta tingkat keterhalangan oleh daun. Seluruh citra pada *test set* tidak digunakan pada proses pelatihan maupun validasi, sehingga dapat digunakan untuk mengevaluasi kemampuan model dalam mengenali objek pada data yang belum pernah dilihat sebelumnya. Pengujian terhadap seluruh 535 citra digunakan dalam perhitungan metrik kinerja model. Namun, karena jumlah data pengujian cukup banyak, 10 citra dipilih sebagai sampel representatif untuk ditampilkan pada laporan. Sampel tersebut mencakup kondisi pencahayaan luar ruangan dan dalam ruangan, objek yang sebagian tertutup daun, serta variasi posisi dan orientasi cabai. Data pengujian di capture di kamera Astra dan dapat dilihat pada tabel 3.8.

Tabel 3.8 Data Pengujian

No	Data Pengujian	Keterangan
1		Data Pengujian A
2		Data Pengujian B

3		Data Pengujian C
4		Data Pengujian D
5		Data Pengujian E
6		Data Pengujian F
7		Data Pengujian G
8		Data Pengujian H
9		Data Pengujian I

10		Data Pengujian J
----	--	------------------

3.5.3 Perancangan Optimasi Inferensi Model

Setelah proses pelatihan dan pemilihan model selesai, tahap selanjutnya adalah melakukan optimasi inferensi agar model dapat dijalankan secara *real-time* pada NVIDIA Jetson Orin Nano. Model yang digunakan pada sistem merupakan YOLOv8n-Seg yang menghasilkan informasi kelas objek, *confidence score*, *bounding box*, dan *instance mask* dari setiap objek cabai yang terdeteksi.

Model hasil pelatihan awalnya disimpan dalam format PyTorch dengan ekstensi .pt. Agar proses inferensi dapat memanfaatkan akselerasi GPU NVIDIA secara lebih optimal, model dikonversi terlebih dahulu ke format ONNX, kemudian dikompilasi menjadi TensorRT *Engine* dengan presisi FP16. *Engine* TensorRT selanjutnya digunakan pada program C++ yang menjalankan proses deteksi, segmentasi, pengolahan fitur visual, dan *Visual Servoing* pada Jetson Orin Nano.

Alur optimai model yang digunakan dapat dinyatakan sebagai berikut:

1) Konversi Model PyTorch ke ONNX

Model YOLOv8n-Seg hasil pelatihan disimpan dalam format PyTorch dengan nama best.pt. Model tersebut terlebih dahulu dikonversi ke format Open Neural Network Exchange (ONNX). Format ONNX digunakan sebagai format perantara agar struktur jaringan dapat diproses oleh TensorRT.

```
from ultralytics import YOLO
model = YOLO("best.pt")
```

```
model.export(  
    format="onnx",  
    imgsz=416,  
    dynamic=False,  
    simplify=True  
)
```

Hasil proses tersebut berupa file `best.onnx`. Pada implementasi sistem, ukuran input model ditetapkan sebesar 416×416 piksel. Penggunaan ukuran input tetap (*static input shape*) mempermudah proses optimasi pada TensorRT karena ukuran tensor masukan telah diketahui pada saat *engine* dibangun.

2) Konversi ONNX menjadi TensorRT *Engine*

File ONNX selanjutnya dikonversi menjadi TensorRT *Engine* menggunakan utilitas `trtexec` yang tersedia pada NVIDIA TensorRT. Pada sistem yang digunakan, proses konversi dilakukan pada NVIDIA Jetson Orin Nano dengan TensorRT. Contoh perintah konversi adalah sebagai berikut.

```
/usr/src/TensorRT/bin/trtexec \  
--onnx=best.onnx \  
--saveEngine=best_trt10.engine \  
--FP16 \  
--memPoolSize=workspace:2048
```

Perintah `--onnx` menentukan model ONNX yang akan dikonversi, sedangkan `--saveEngine` menentukan nama file TensorRT *Engine* yang dihasilkan. Opsi `--FP16` digunakan untuk mengaktifkan komputasi *half precision* FP16 pada layer yang mendukungnya. Parameter *workspace* memberikan ruang memori yang dapat digunakan TensorRT selama proses optimasi *engine*. Hasil konversi berupa file `best_trt10.engine` yang kemudian digunakan langsung oleh program C++ pada Jetson Orin Nano.

Karena model ONNX yang digunakan memiliki ukuran input statis $1 \times 3 \times 416 \times 416$, proses pembuatan *engine* tidak memerlukan pengaturan *dynamic input shape*. Hal ini juga mengurangi kebutuhan pengelolaan profil optimasi TensorRT ketika model dijalankan.

TensorRT *Engine* dibangun menggunakan presisi FP16 (*16-bit floating point*). Dibandingkan FP32, representasi FP16 menggunakan ukuran data yang lebih kecil sehingga dapat mengurangi kebutuhan bandwidth memori dan meningkatkan kemampuan komputasi paralel pada GPU NVIDIA dan Tensor Core yang mendukung pemrosesan *half precision*. Penggunaan FP16 tidak mengubah arsitektur dasar YOLOv8n-Seg, tetapi mengubah presisi numerik yang digunakan pada operasi jaringan yang mendukung FP16. TensorRT secara otomatis melakukan optimasi terhadap *layer*, penggunaan kernel CUDA, pengelolaan memori, dan urutan eksekusi jaringan untuk memperoleh waktu inferensi yang lebih rendah.

3) Implementasi TensorRT pada Program C++

TensorRT *Engine* yang telah dihasilkan selanjutnya dimuat oleh program C++ pada saat sistem dijalankan. *Engine* tidak dimuat ulang pada setiap frame, melainkan diinisialisasi satu kali ketika aplikasi dimulai. Model TensorRT dijalankan menggunakan file *best_trt10.engine*. Sebelum memasuki TensorRT *Engine*, citra melalui tahap *preprocessing*, yaitu penyesuaian ukuran citra menjadi 416×416 piksel, konversi format data, normalisasi nilai piksel, serta penyusunan data ke format tensor masukan model.

Normalisasi nilai piksel dilakukan dari rentang 0 – 255 menjadi 0 – 1 menggunakan:

$$I_n = \frac{I}{255}$$

dengan:

I = nilai piksel citra asli, dan

I_n = nilai piksel setelah normalisasi.

Setelah proses inferensi selesai, keluaran TensorRT diproses kembali untuk memperoleh kelas objek, *confidence score*, *bounding box*, dan informasi *mask* segmentasi.

Dengan metode tersebut, sebagian besar perhitungan jaringan dilakukan langsung pada GPU Jetson sehingga CPU dapat digunakan untuk proses lain seperti *target tracking*, perhitungan *cut-point*, *Visual Servoing*, GUI, dan komunikasi UART dengan STM32.

3.5.4 Perancangan *Hardware*

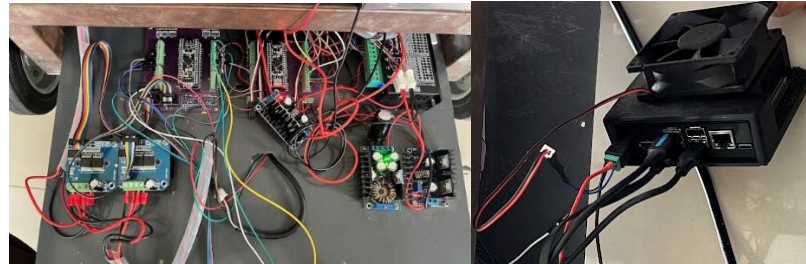
Perancangan *hardware* dilakukan melalui proses perakitan mekanik robot manipulator 5-DOF, pemasangan sistem elektrik, pemasangan kamera RGB Orbbec Astra Pro, serta integrasi perangkat kendali dan HMI. Seluruh komponen dirakit pada platform Chimi-Bot sehingga sistem persepsi, pengendali gerak, dan aktuator robot dapat bekerja sebagai satu kesatuan. Hasil perakitan keseluruhan sistem ditunjukkan pada **Gambar 3.16**



Gambar 3.16 Hasil Perakitan *Hardware*

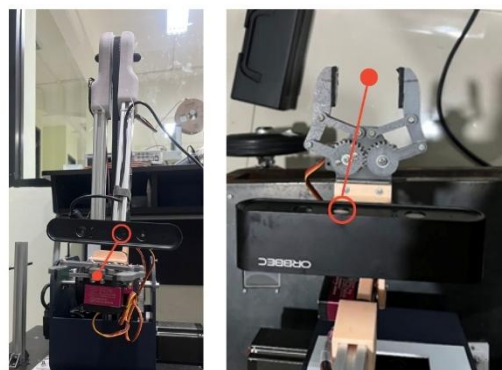
Pada perakitan elektrik, komponen pengendali dan distribusi daya ditempatkan pada bagian dasar Chimi-Bot. NVIDIA Jetson Orin Nano digunakan sebagai unit pemrosesan citra dan *visual servo controller*, sedangkan STM32F411CEU6 digunakan sebagai pengendali gerak robot manipulator. Jetson dilengkapi sistem pendingin aktif untuk membantu menjaga temperatur perangkat selama proses inferensi dan pengoperasian

sistem. Komunikasi antara Jetson dan STM32 dilakukan melalui komunikasi serial UART. Hasil perakitan elektrikal dan sistem pendingin ditunjukkan pada **Gambar 3.17**.



Gambar 3. 17 Hasil Perakitan Elektrikal

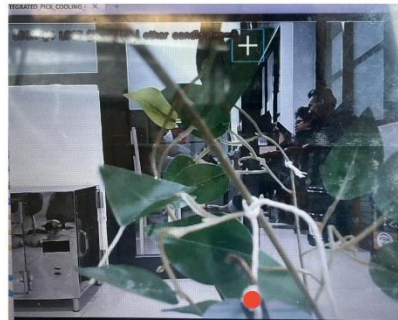
Kamera RGB Orbbec Astra Pro dipasang dengan konfigurasi *eye-in-hand* secara rigid pada bagian *wrist yaw* robot. Pemasangan rigid dilakukan agar posisi dan orientasi kamera terhadap bagian ujung robot tetap konstan selama proses *Visual Servoing*. Berdasarkan konfigurasi mekanik yang digunakan, posisi pusat lensa RGB terhadap TCP memiliki *offset* sebesar 110 mm pada arah longitudinal, -15 mm pada arah lateral, dan +45 mm pada arah vertikal. Konfigurasi pemasangan kamera dan *offset* terhadap TCP ditunjukkan pada **Gambar 3.18**.



Gambar 3.18 Offset kamera dari *gripper*

Selain mempertahankan posisi relatif kamera terhadap TCP, orientasi kamera juga dibuat tetap menghadap lurus ke arah depan selama pergerakan Kartesian robot. Pada setiap perintah perpindahan X, Y, dan Z, orientasi *wrist yaw* dipertahankan sehingga perpindahan robot ke arah lateral tidak

menyebabkan kamera ikut berputar mengikuti arah gerak. Dengan konfigurasi tersebut, arah pandang kamera tetap konsisten dan hubungan antara perubahan posisi robot dengan perpindahan fitur pada bidang citra dapat dipertahankan selama proses *Visual Servoing*. Posisi *gripper* yang terlihat dari bidang pandang kamera ditunjukkan pada **Gambar 3.19**.



Gambar 3.19 Posisi *Gripper* di frame kamera

Posisi relatif *gripper* terhadap bidang pandang kamera selanjutnya digunakan sebagai acuan dalam menentukan *desired image point*. Penentuan titik tersebut dilakukan dengan mengamati posisi proyeksi area kerja TCP/*gripper* pada citra ketika robot berada pada konfigurasi yang digunakan untuk melakukan pemetikan. Karena pusat lensa kamera tidak berimpit dengan TCP, posisi acuan pemetikan tidak berada tepat pada pusat citra, melainkan mengalami pergeseran sesuai konfigurasi pemasangan kamera dan *gripper*.

Pada sistem yang digunakan, titik acuan tersebut ditetapkan pada koordinat $(u^*, v^*) = (387, 66)$ piksel. Titik ini merepresentasikan posisi pada bidang citra yang harus dicapai oleh *cut-point* cabai sebelum robot melanjutkan proses pemetikan.

HMI pada Chimi-Bot digunakan sebagai antarmuka utama untuk menjalankan sistem robot. HMI menampilkan status sistem dan menyediakan perintah operasi seperti *start* dan *stop*. Informasi status dari pengendali robot digunakan untuk memastikan sistem berada pada kondisi siap sebelum proses

pemindaian dan pemetikan dijalankan. Tampilan HMI yang digunakan pada Chimi-Bot ditunjukkan pada **Gambar 3.20**.



Gambar 3.20 HMI pada Chimi-Bot

3.5.5 Perancangan Komunikasi Jetson-STM32

Perancangan komunikasi Jetson–STM32 bertujuan untuk menghubungkan NVIDIA Jetson Orin Nano sebagai unit pemrosesan citra dan pengendali tingkat tinggi dengan STM32F411CEU6 sebagai pengendali gerak robot. Jetson menghasilkan informasi target dan koreksi gerak berdasarkan citra kamera, sedangkan STM32 menjalankan perintah gerak pada robot manipulator melalui perhitungan kinematika dan pengendalian aktuator.

Komunikasi dilakukan melalui antarmuka USB serial. Pada Jetson, perangkat STM32 dikenali sebagai `/dev/ttyACM0`. Sebelum sistem *Auto Pick* dijalankan, program terlebih dahulu membuka koneksi serial dan memeriksa status robot. Jetson mengirimkan perintah gerak Kartesian dan perintah operasi, sedangkan STM32 mengirimkan kembali data posisi hasil *forward kinematics*, nilai *joint*, serta status *RUN* dan *HOMED*. Data status tersebut digunakan sebagai kondisi pengaman agar proses pengambilan citra berikutnya dilakukan setelah pergerakan robot selesai.

Pola komunikasi dibuat berurutan sesuai prinsip *step-look*. Setelah satu perintah gerak dikirim, sistem menunggu pembaruan kondisi robot sebelum kamera mengambil citra baru. Dengan metode tersebut, perubahan posisi *cut-point* yang diamati pada citra dapat dikaitkan dengan gerak robot yang baru saja dilakukan dan digunakan kembali pada iterasi kontrol berikutnya.

Tabel 3. 9 Konfigurasi Komunikasi Jetson–STM32

Bagian	Konfigurasi	Fungsi
Antarmuka	USB Serial / USB CDC	Pertukaran data antara Jetson dan STM32
Perangkat pada Jetson	/dev/ttyACM0	Port komunikasi <i>robot arm</i>
Jetson ke STM32	Perintah gerak Kartesian dan kontrol operasi	Menjalankan perpindahan robot sesuai hasil kontrol visual
STM32 ke Jetson	Pose XYZ, <i>joint</i> , RUN, HOMED, dan status gerak	Umpan balik kondisi robot untuk proses <i>step-look</i>

3.5.6 Perancangan *Target selection*, *Target lock*, dan *Cut-point*

Tahap ini dirancang untuk mengubah keluaran YOLOv8n-Seg menjadi satu target aktif yang dapat digunakan oleh pengendali *Visual Servoing*. Hasil inferensi berupa kelas objek, *confidence score*, *bounding box*, dan *instance mask*. Cabai matang diperlakukan sebagai kandidat pemetikan, sedangkan cabai belum matang tetap dapat terdeteksi tetapi tidak dipilih sebagai target pemetikan.

a. Ekstraksi fitur *instance mask*

Setiap *mask* cabai matang diolah untuk memperoleh luas *mask*, *centroid*, kontur, batas atas dan bawah objek, serta orientasi utama objek.

b. Penentuan *cut-point*.

Titik-titik kontur dianalisis menggunakan Principal Component Analysis (PCA) untuk memperoleh sumbu utama cabai. Arah sumbu tersebut digunakan sebagai acuan untuk menentukan titik kontrol pada bagian atas objek yang merepresentasikan *cut-point* pada koordinat citra.

c. Pemilihan target

Apabila terdapat lebih dari satu cabai matang, kandidat dipilih berdasarkan kedekatan *cut-point* terhadap *desired point* pada citra dengan *confidence score* sebagai informasi pendukung.

d. Penguncian target

Setelah satu target dipilih, sistem mempertahankan target yang sama menggunakan informasi posisi *mask*, *centroid*, dan area pencarian lokal (*local ROI*). Mekanisme ini mencegah sistem berpindah ke cabai lain ketika terdapat beberapa kandidat dalam satu bidang pandang.

Informasi akhir yang diteruskan ke pengendali berupa koordinat *cut-point* (u, v), luas *mask*, *confidence score*, dan status target. Seluruh informasi tersebut tetap berada pada ruang citra dua dimensi sehingga proses ini tidak membutuhkan konversi posisi target menjadi koordinat tiga dimensi dari kanal *depth* kamera.

3.5.7 Perancangan *Visual Servoing* dan *Approach Control*

Perancangan *Visual Servoing* digunakan untuk mengarahkan *cut-point* target menuju *desired point* pada citra. Metode yang diterapkan adalah *uncalibrated image-based visual servoing* berbasis *empirical local image Jacobian*. Hubungan antara gerak Kartesian robot dan perubahan koordinat *cut-point* tidak dihitung menggunakan model kamera analitik, tetapi diperoleh melalui respons citra terhadap gerakan robot yang diketahui.

a. Pengambilan *empirical Jacobian*

Robot diberi gerakan kecil pada sumbu Y dan Z secara terpisah. Perubahan posisi *cut-point* sebelum dan sesudah gerakan dicatat sehingga sensitivitas perubahan piksel terhadap perpindahan Kartesian dapat dibentuk menjadi matriks Jacobian lokal.

b. Perhitungan koreksi lateral.

Error koordinat *cut-point* terhadap *desired point* dihitung pada setiap iterasi. Nilai error kemudian dipetakan menjadi koreksi gerak Y–Z menggunakan *pseudoinverse* dari Jacobian empiris. Besar langkah dibatasi untuk menghindari gerak yang terlalu besar ketika target berada dekat tepi citra atau ketika nilai error dipengaruhi *noise*.

c. Penerapan *deadband*

Koreksi Y–Z dihentikan ketika *cut-point* telah berada di dalam area toleransi di sekitar *desired point*. Pada implementasi akhir digunakan toleransi sekitar ± 25 piksel untuk mencegah robot terus melakukan koreksi kecil akibat variasi hasil segmentasi.

d. Pendekatan longitudinal sumbu X

Pergerakan maju dilakukan secara bertahap dengan menggunakan luas *instance mask* sebagai indikator kedekatan relatif. Sistem menggunakan langkah yang lebih besar ketika target masih jauh dan memperkecil langkah ketika target semakin dekat. Setelah setiap gerakan X, sistem kembali mengambil citra dan melakukan koreksi Y–Z apabila diperlukan.

Seluruh proses dijalankan dengan pola *step-look*, yaitu kamera mengambil citra, sistem menghitung fitur dan error, robot melakukan satu langkah gerak, kemudian berhenti sebelum citra berikutnya diambil. Kondisi siap pemetikan ditentukan ketika posisi *cut-point* telah memenuhi toleransi dan informasi visual menunjukkan target telah berada pada kondisi pendekatan yang ditentukan. Kanal *depth* Astra Pro tidak digunakan sebagai masukan utama pada proses ini.

3.5.8 Perancangan Program Auto Pick dan Finite State Machine

Program *Auto Pick* dirancang agar proses pencarian target hingga pemetikan dapat berlangsung secara berurutan tanpa menjalankan setiap perintah secara manual. Urutan proses dibentuk menggunakan *Finite State Machine* (FSM), sehingga setiap tahap hanya dijalankan apabila kondisi pada tahap sebelumnya telah terpenuhi.

Tabel 3. 10 Tahapan Finite State Machine Auto Pick

State	Fungsi Utama
WAIT / READY	Menunggu perintah Start Auto dan memastikan kamera, model, serta robot siap digunakan.
SCAN	Menggerakkan <i>robot arm</i> menuju <i>scan pose</i> untuk memperoleh bidang pandang tanaman.
DETECT / LOCK	Mendeteksi cabai matang, menentukan <i>cut-point</i> , memilih kandidat, dan mengunci satu target aktif.
APPROACH	Melakukan pendekatan sumbu X secara bertahap berdasarkan perubahan informasi visual.
ALIGN	Melakukan koreksi lateral Y–Z menggunakan <i>empirical Jacobian</i> sampai <i>cut-point</i> berada di dalam <i>deadband</i> .

State	Fungsi Utama
PICK READY	Memastikan kondisi visual target memenuhi syarat sebelum urutan pemetikan dijalankan.
PICK / DROP	Menjalankan urutan gerak pemetikan dan memindahkan hasil pemetikan ke lokasi penampungan.
RETURN / DONE	Mengembalikan robot ke posisi aman dan mengirim status selesai untuk melanjutkan siklus sistem.

Pada kondisi target tidak ditemukan, sistem tidak memaksakan proses *Visual Servoing* dan robot dikembalikan ke posisi aman. Jika target yang telah dikunci sementara hilang dari citra, proses pencarian kembali dilakukan pada area sekitar target terakhir sebelum sistem menentukan tindakan berikutnya. Selain itu, perintah *Stop* pada antarmuka digunakan sebagai penghentian global ketika sistem berada pada salah satu state aktif.

Setelah kondisi *PICK READY* tercapai, Jetson mengirimkan perintah urutan pemetikan kepada sistem gerak robot. Setelah proses pemetikan dan peletakan selesai, robot kembali ke posisi yang telah ditentukan. Pada sistem terintegrasi dengan *mobile platform*, status *DONE_PICK* digunakan sebagai penanda bahwa siklus pada satu titik tanaman telah selesai sehingga platform dapat melanjutkan ke titik berikutnya

3.6 Metode Pengujian dan Analisa

Proses pengujian dan analisa pada perancangan sistem persepsi dan *Visual Servoing* pada Robot Mobile Chilli Picker bertujuan untuk mengetahui kinerja setiap bagian sistem, mulai dari model YOLOv8n-Seg, implementasi model pada NVIDIA Jetson Orin Nano, penguncian target dan penentuan *cut-point*, hingga kemampuan pengendali visual dalam mengarahkan robot menuju kondisi pemetikan. Metode pengujian dan analisa yang digunakan dalam tugas akhir ini dijelaskan sebagai berikut:

1. Analisa *training* Model A dan Model B YOLOv8n-Seg.
2. Pengujian kedua model pada *test set* menggunakan *confusion matrix*, *precision*, *recall*, Skor F1, mAP, dan PR Curve.

3. Pengujian implementasi TensorRT FP16 pada NVIDIA Jetson Orin Nano berdasarkan *latency*, FPS, *runtime*, dan pengaruh pendinginan.
4. Pengujian *target selection* dan *target lock*.
5. Pengujian pengaruh orientasi cabai terhadap penentuan *cut-point* berbasis PCA.
6. Pengujian *Visual Servoing* lateral Y-Z berdasarkan perubahan *image error* terhadap *desired point*.
7. Pengujian pendekatan longitudinal X dengan metode *step-look* dan luas *instance mask*.
8. Pengujian integrasi pemetikan otomatis dari deteksi sampai *pick*, *drop*, dan kembali ke *scan pose*.