

BAB II

DASAR TEORI

Pada bab mengenai dasar teori ini, akan dibahas mengenai teori yang terkait dengan topik pembahasan dan komponen-komponen yang digunakan dalam pembuatan.

2.1 Tinjauan Pustaka

Pada bagian tinjauan pustaka berisi beberapa referensi penelitian terdahulu yang relevan dengan perancangan sistem deteksi dan segmentasi buah, implementasi model pada NVIDIA Jetson Orin Nano, penentuan titik pemetikan berdasarkan hasil segmentasi, penguncian target, serta penerapan *uncalibrated image-based visual servoing* pada sistem robotik. Tinjauan ini digunakan sebagai dasar dalam menentukan metode YOLOv8n-Seg, pengolahan geometris *instance mask*, *empirical Jacobian*, dan kontrol visual bertahap pada Robot *Mobile Chilli Picker*. Hasil tinjauan pustaka disajikan pada **Tabel 2.1**.

Tabel 2.1 Tinjauan Pustaka

Judul Jurnal	Penulis/ Tahun	Pembahasan	Persamaan dan Perbedaan
<i>Performance Evaluation of YOLOv8 Model Configurations for Instance segmentation of Strawberry Fruit Development Stages in an Open Field Environment</i>	Gamani et al., 2024	Mengevaluasi beberapa konfigurasi YOLOv8 untuk <i>instance segmentation</i> tahap perkembangan buah stroberi.	Persamaan: menggunakan YOLOv8 dan <i>instance segmentation</i> untuk membedakan tingkat kematangan buah. Perbedaan: penelitian tersebut fokus pada evaluasi model, sedangkan penelitian ini meneruskan hasil <i>mask</i> sebagai input <i>target lock</i> , cut point, dan <i>Visual Servoing</i> .
<i>Precision Citrus Segmentation and Stem Picking point Localization</i>	Li et al., 2025	Menggunakan improved YOLOv8n-Seg untuk segmentasi buah dan tangkai jeruk serta geometric	Persamaan: menggunakan YOLOv8n-Seg dan geometri hasil <i>mask</i> untuk menentukan titik

Judul Jurnal	Penulis/ Tahun	Pembahasan	Persamaan dan Perbedaan
<i>Using Improved YOLOv8n-Seg Algorithm</i>		constraints untuk menentukan <i>picking point</i> .	pemetikan. Perbedaan: Li et al. melakukan segmentasi buah dan tangkai jeruk secara langsung, sedangkan penelitian ini menentukan cut point cabai dari geometri <i>mask/PCA</i> .
<i>Fully Uncalibrated image-based visual servoing of 2DOFs Planar Manipulators With a Fixed Camera</i>	Liang et al., 2022	Mengembangkan metode <i>fully uncalibrated image-based visual servoing</i> untuk pengendalian posisi manipulator tanpa memerlukan kalibrasi parameter kamera. Metode menggunakan error posisi fitur pada bidang citra untuk mengendalikan pergerakan robot.	Persamaan: sama-sama menggunakan uncalibrated IBVS, dengan error fitur citra sebagai dasar koreksi pergerakan robot serta mengurangi ketergantungan terhadap parameter kalibrasi kamera. Perbedaan: Liang et al. menggunakan manipulator planar 2-DOF dengan kamera tetap, sedangkan penelitian ini menggunakan manipulator 5-DOF dengan konfigurasi <i>eye-in-hand</i> dan koreksi Kartesian XYZ terhadap <i>cut-point</i> cabai.
<i>A Data-Driven Acceleration-Level Scheme for Image-Based Visual Servoing of Manipulators with Unknown Structure</i>	Wen & Xie, 2024	Mengembangkan IBVS berbasis data untuk manipulator dengan struktur/Jacobian yang tidak diketahui. Hubungan antara gerakan robot dan perubahan fitur citra dipelajari melalui virtual visual Jacobian berdasarkan data gerak dan	Persamaan: sama-sama tidak sepenuhnya mengandalkan model Jacobian analitik dan menggunakan hubungan empiris/data-driven antara gerakan robot dan perubahan fitur citra. Perbedaan: Wen dan Xie mengestimasi <i>virtual visual Jacobian</i> secara <i>online</i> menggunakan RNN dan kontrol tingkat

Judul Jurnal	Penulis/ Tahun	Pembahasan	Persamaan dan Perbedaan
		perubahan koordinat piksel. Metode diuji pada manipulator dengan konfigurasi <i>eye-in-hand</i>	percepatan, sedangkan penelitian ini memperoleh <i>empirical Jacobian</i> melalui <i>probing</i> gerakan Kartesian XYZ dan menggunakannya untuk menghitung koreksi <i>cut-point</i> terhadap <i>desired point</i> .
<i>A Distance Measurement Approach for Large Fruit Picking with Single Camera</i>	Liu et al., 2023	Mengembangkan metode pengukuran jarak buah menggunakan kamera monokular. Jarak objek diperkirakan berdasarkan hubungan antara jumlah piksel yang ditempati area target pada citra dengan perubahan jarak kamera terhadap objek. Model dibangun dari data objek pada berbagai jarak dan kemudian diuji pada beberapa jenis buah berukuran besar. Hasil menunjukkan metode monocular ranging dapat mencapai <i>relative error</i> sekitar 5%.	Persamaan: sama-sama menggunakan perubahan ukuran/luas proyeksi objek dalam satuan piksel sebagai indikator jarak relatif terhadap kamera. Pada penelitian ini, luas yang digunakan berasal dari <i>segmentation mask</i> YOLOv8n-Seg, kemudian perubahan luas <i>mask</i> digunakan untuk menentukan kondisi FAR, MID, dan NEAR selama pendekatan robot. Perbedaan: Liu et al. membangun model regresi untuk memperkirakan jarak absolut buah dari hubungan jumlah piksel dan jarak.

2.2 Computer vision

Computer vision merupakan bidang yang mempelajari bagaimana informasi visual dari citra dapat diolah sehingga karakteristik objek dapat diperoleh secara komputasional. Dalam sistem robot berbasis kamera, citra bukan hanya digunakan untuk mengenali objek, tetapi dapat menjadi sumber informasi mengenai

posisi objek pada bidang citra. Hubungan antara posisi suatu titik di ruang kamera dengan posisi proyeksinya pada citra dapat dijelaskan menggunakan model proyeksi perspektif atau pinhole camera model[13]. Karena koordinat hasil *computer vision* dinyatakan dalam satuan piksel, karakteristik internal kamera perlu diketahui agar hubungan antara koordinat citra dan model geometris kamera dapat dijelaskan dengan benar.[14].

2.2.1 *Object detection* dan *Instance segmentation*

Object detection bertujuan menentukan kelas dan lokasi objek menggunakan kotak pembatas (*bounding box*). Metode ini dapat menunjukkan letak objek secara umum, tetapi belum memberikan informasi bentuk objek secara rinci. *Instance segmentation* menambahkan segmentasi pada tingkat piksel sehingga setiap objek pada kelas yang sama memiliki *mask* tersendiri. Informasi tersebut memungkinkan bentuk, luas, titik pusat, kontur, dan orientasi masing-masing objek dianalisis secara terpisah.



Gambar 2.1 Keluaran *Instance segmentation*

2.2.2 YOLOv8-Seg

YOLO (*You Only Look Once*) merupakan keluarga model deteksi satu tahap yang memprediksi objek secara langsung dari citra masukan [9]. YOLOv8-Seg merupakan pengembangan YOLO yang mendukung tugas *instance segmentation*. Keluaran model meliputi kelas objek, nilai *confidence*, *bounding box*, dan *instance mask* untuk setiap objek yang terdeteksi. Huruf **n** pada YOLOv8n-Seg menunjukkan varian *nano*, yaitu varian dengan kebutuhan komputasi yang lebih ringan dibandingkan varian yang lebih besar. Karakteristik tersebut sesuai untuk implementasi pada perangkat *edge* yang harus menjalankan inferensi secara waktu nyata

Secara umum, jaringan YOLOv8-Seg dapat dijelaskan melalui tiga bagian utama, yaitu *backbone*, *neck*, dan *head*. *Backbone* mengekstraksi fitur dari citra, *neck* menggabungkan fitur pada beberapa skala, sedangkan *head* menghasilkan prediksi deteksi dan segmentasi. Pembahasan arsitektur pada bagian ini difokuskan pada fungsi umum setiap bagian tanpa menguraikan struktur setiap lapisan jaringan.

2.2.3 Evaluasi Model Segmentasi

Evaluasi dilakukan dengan membandingkan prediksi model terhadap *ground truth*. Pada deteksi dan *instance segmentation*, sebuah prediksi dinyatakan sesuai apabila kelas cocok dan nilai *Intersection over Union* (IoU) memenuhi ambang evaluasi. IoU menyatakan perbandingan luas irisan terhadap luas gabungan *mask* prediksi dan *ground truth*.

$$IoU = \frac{|M_p \cap M_{gt}|}{|M_p \cup M_{gt}|} \quad (2.1)$$

Confusion matrix merupakan metode evaluasi yang digunakan untuk mengukur kinerja suatu model klasifikasi dengan membandingkan hasil prediksi model terhadap data sebenarnya (*ground truth*). Matriks ini menampilkan empat komponen utama yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). *True Positive* yang dapat dilihat pada **Gambar 2.22** yang menunjukkan jumlah data yang diprediksi positif dan memang benar positif, sedangkan *True Negative* menunjukkan jumlah data yang diprediksi negatif dan memang benar negatif. *False Positive* terjadi ketika model memprediksi positif namun sebenarnya negatif, sementara *False Negative* terjadi ketika model memprediksi negatif padahal sebenarnya positif. Dengan menggunakan *confusion matrix*, berbagai metrik evaluasi seperti *precision*, *recall*, *accuracy*, dan Skor F1 dapat dihitung untuk mengetahui seberapa baik performa model dalam melakukan klasifikasi atau deteksi objek.[15]

True Positive (TP) adalah objek yang terdeteksi dengan kelas dan pasangan *ground truth* yang benar. *False Positive* (FP) adalah prediksi yang tidak memiliki pasangan *ground truth* yang sesuai atau salah kelas, sedangkan *False Negative* (FN) adalah objek *ground truth* (GT) yang tidak berhasil ditemukan. *True Negative* tidak dihitung secara eksplisit karena jumlah piksel atau area latar belakang sangat besar dan bukan unit objek yang dievaluasi.

		AKTUAL	
		Positif	Negatif
PREDIKSI	Positif	<i>True Positive</i> (TP)	<i>False Positive</i> (FP)
	Negatif	<i>False Negative</i> (FN)	<i>True Negative</i> (TN)

Gambar 2.2 *Confusion matrix*[15]

Nilai metrik evaluasi biasanya dihitung berdasarkan hasil dari *confusion matrix*. Metrik evaluasi yang pertama digunakan yaitu *Precision*. Persamaan (2.18) menunjukkan seberapa banyak prediksi positif yang dilakukan oleh model benar-benar merupakan data positif. Selanjutnya terdapat *recall* yang secara sistematis pada persamaan (2.19), yang menunjukkan kemampuan model dalam mendeteksi seluruh data positif yang sebenarnya. Untuk menyeimbangkan nilai *precision* dan *recall* digunakan Skor F1 pada persamaan (2.20), yaitu rata-rata harmonik dari kedua metrik tersebut sehingga dapat memberikan gambaran performa model secara lebih menyeluruh. Berikut rumus untuk tiap metrik evaluasi.

$$Precision = \frac{TP}{TP + FP} \quad (2.2)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.3)$$

$$F1 = 2 \frac{Precision \cdot Recall}{Precision + Recall} \quad (2.4)$$

Selain itu, ada *Average precision* (AP) pada persamaan (2.21) yang merupakan nilai yang diperoleh dari luas area di bawah kurva *Precision–Recall*, yang digunakan untuk mengevaluasi performa model pada satu kelas objek. Semakin tinggi nilai AP, maka semakin baik kemampuan model dalam mendeteksi objek pada kelas tersebut. Sementara itu, *mean Average precision* (mAP) pada persamaan (2.22) merupakan rata-rata dari nilai *Average precision* untuk seluruh kelas objek dalam dataset sehingga dapat menggambarkan performa keseluruhan model deteksi objek.

$$AP = \int_0^1 P(R) dR \quad (2.5)$$

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.6)$$

Keterangan:

$P(R)$ = *Precision* sebagai fungsi *recall*

N = Jumlah kelas objek

AP_i = Nilai *Average precision* pada kelas ke- i

2.3 Visual Servoing

Visual Servoing merupakan metode pengendalian robot yang menggunakan informasi visual dari kamera sebagai umpan balik (*feedback*) di dalam *loop* kontrol. Informasi visual tersebut digunakan untuk menentukan arah koreksi gerak robot sampai keadaan visual saat ini mendekati keadaan yang diinginkan. Pendekatan yang secara umum digunakan ada *Position-Based Visual Servoing* (PBVS), error biasanya dibentuk dari estimasi pose tiga dimensi, sedangkan pada *Image-Based Visual Servoing* (IBVS) error dibentuk langsung dari fitur pada bidang citra.

Pendekatan ini tidak membutuhkan rekonstruksi pose objek secara penuh pada setiap iterasi kontrol. Karena error dikendalikan langsung pada ruang citra, IBVS dikenal lebih toleran terhadap kesalahan kalibrasi walaupun tetap memiliki

permasalahan seperti nonlinieritas pemetaan citra, singularitas, dan kemungkinan target keluar dari *field of view* kamera.

Pada sistem dengan kamera *eye-in-hand*, kamera dipasang pada bagian manipulator dan bergerak bersama *end-effector*. Konfigurasi ini memungkinkan target diamati kembali setelah robot melakukan gerakan sehingga kesalahan visual dapat diperbarui secara bertahap [3], [4]. Hubungan antara perubahan fitur pada citra dan perubahan gerak robot dapat direpresentasikan melalui *image Jacobian* atau *interaction matrix*. Pada pendekatan yang tidak menggunakan model analitik penuh, hubungan lokal tersebut dapat diperoleh secara empiris dari perubahan fitur yang diamati akibat gerakan robot yang diketahui [19].

2.3.1 Image-Based Visual Servoing (IBVS)

IBVS menggunakan fitur citra sebagai variabel yang dikendalikan. Misalkan posisi fitur visual aktual dinyatakan sebagai vektor

$$\mathbf{s} = \begin{bmatrix} u \\ v \end{bmatrix} \quad (2.7)$$

sedangkan posisi fitur yang diinginkan pada citra dinyatakan sebagai

$$\mathbf{s}^* = \begin{bmatrix} u^* \\ v^* \end{bmatrix} \quad (2.8)$$

Pada sistem ini, u dan v menyatakan koordinat piksel dari fitur visual yang digunakan sebagai titik kontrol. Fitur tersebut dapat berupa titik, pusat objek, sudut, garis, momen citra, maupun fitur lain yang dapat diekstraksi secara konsisten dari citra. Pada penelitian ini, titik kontrol utama berupa *cut-point* hasil pengolahan *mask* segmentasi.

Error citra didefinisikan sebagai selisih antara fitur aktual dan fitur yang diinginkan,

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* = \begin{bmatrix} e_u \\ e_v \end{bmatrix}, \quad (2.9)$$

Besarnya error total dapat dinyatakan menggunakan norma Euclidean,

$$\|\mathbf{e}\|_2 = \sqrt{e_u^2 + e_v^2}. \quad (2.10)$$

Tujuan dasar IBVS adalah menghasilkan gerak kamera atau robot yang menyebabkan error citra menuju nol ($\mathbf{e} \rightarrow \mathbf{0}$). Dalam formulasi IBVS klasik, perubahan fitur citra berhubungan dengan kecepatan kamera melalui *interaction matrix* atau *image Jacobian*,

$$\dot{\mathbf{s}} = \mathbf{L}_s \mathbf{v}_c, \quad (2.11)$$

Dengan $\mathbf{L}_s$ sebagai *interaction matrix* dan $\mathbf{v}_c$ sebagai vektor kecepatan kamera. Persamaan tersebut menjadi dasar bahwa suatu perubahan gerak kamera akan menimbulkan perubahan terukur pada fitur di bidang citra. Pada implementasi diskrit dan *uncalibrated* yang digunakan dalam penelitian, hubungan tersebut tidak dihitung dari model kamera analitik, tetapi didekati melalui Jacobian lokal yang diperoleh dari hasil gerak robot secara empiris.

pemilihan fitur citra sangat memengaruhi perilaku IBVS karena error, interaksi visual-motor, dan konvergensi kontrol semuanya dibentuk berdasarkan fitur yang dipilih. Oleh karena itu, fitur yang digunakan perlu mudah diamati, stabil terhadap perubahan citra, dan tetap berada dalam area pengamatan kamera selama proses servoing.

2.3.2 Fitur Visual dari Mask Segmentasi

Hasil *instance segmentation* dapat dinyatakan sebagai *mask* biner $M(u, v)$. Piksel yang termasuk objek bernilai satu, sedangkan piksel latar bernilai nol.

$$M(u, v) = \begin{cases} 1, & \text{piksel objek} \\ 0, & \text{piksel latar} \end{cases} \quad (2.12)$$

Luas *mask* merupakan jumlah piksel yang termasuk ke dalam objek dan dapat dihitung sebagai:

$$A = \sum_u \sum_v M(u, v) \quad (2.13)$$

Selain luas, titik pusat geometris objek dapat diperoleh melalui momen citra. Momen orde p, q didefinisikan sebagai [13]:

$$m_{pq} = \sum_u \sum_v u^p v^q M(u, v) \quad (2.14)$$

Berdasarkan momen tersebut, koordinat *centroid* dapat dihitung menggunakan:

$$c_u = \frac{m_{10}}{m_{00}}, \quad c_v = \frac{m_{01}}{m_{00}} \quad (2.15)$$

Centroid menggambarkan pusat geometris *mask*, sedangkan kontur merupakan kumpulan titik pada batas luar objek. Titik-titik kontur dapat digunakan untuk menganalisis bentuk dan orientasi dominan suatu *instance*. *Principal Component Analysis* (PCA) merupakan metode transformasi linear yang memperoleh arah dengan variasi terbesar dari sekumpulan data. Jika koordinat titik kontur diperlakukan sebagai data dua dimensi, PCA dapat digunakan untuk memperoleh sumbu utama objek dari *eigenvector* matriks kovarian. Jika titik-titik kontur cabai dinyatakan sebagai $x_i = [u_i, v_i]^T$, pusat data dihitung menggunakan:

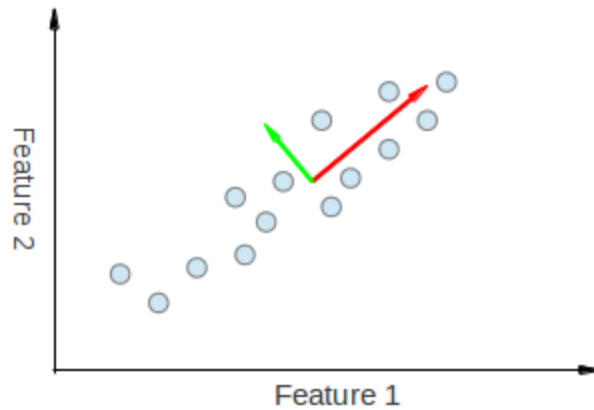
$$c = \frac{1}{N} \sum_{i=1}^N x_i \quad (2.16)$$

Matriks kovarians selanjutnya dihitung sebagai:

$$C = \frac{1}{N} \sum_{i=1}^N (x_i - c) (x_i - c)^T \quad (2.17)$$

Arah utama diperoleh dari *eigenvector* matriks kovarians. *Eigenvector* dengan *eigenvalue* terbesar menjadi sumbu mayor objek.

$$C e_k = \lambda_k e_k, \quad \lambda_1 \geq \lambda_2 \quad (2.18)$$



Gambar 2.3 Ilustrasi arah *eigenvector* pada data dua dimensi

Sumbu utama hasil PCA dapat digunakan sebagai acuan untuk menentukan titik yang bergeser dari *centroid*. Secara umum, titik acuan pada bidang citra dapat dinyatakan sebagai:

$$p_{cut} = c \pm de_1 \quad (2.19)$$

Dengan d merupakan jarak pergeseran dalam satuan piksel dan e_1 merupakan *eigenvector* utama. Besar dan arah pergeseran ditentukan berdasarkan bagian cabai yang akan dijadikan titik potong. Selain itu, luas *mask A* digunakan sebagai informasi perubahan skala objek pada citra, sedangkan posisi batas bawah *mask* dapat dinyatakan sebagai

$$v_b = \max\{v \mid M(u, v) = 1\} \quad (2.20)$$

Nilai v_b menunjukkan seberapa jauh proyeksi objek memanjang ke bagian bawah citra. Nilai ini tidak langsung dianggap sebagai jarak dalam satuan milimeter, tetapi digunakan sebagai salah satu petunjuk kedekatan dan kondisi visual objek.

2.3.3 Uncalibrated dan Model-Free IBVS

Pada IBVS konvensional, hubungan antara perubahan fitur citra dan gerak robot biasanya dihitung menggunakan *interaction matrix* yang diturunkan dari model perspektif kamera. Perhitungan tersebut dapat

memerlukan parameter intrinsik kamera, kedalaman fitur, transformasi kamera terhadap robot, dan model kinematika yang cukup akurat. Pada kondisi lapangan, parameter tersebut dapat berubah atau sulit diperoleh dengan presisi tinggi.

Uncalibrated IBVS mengurangi ketergantungan tersebut dengan memperkirakan hubungan visual-motor dari data atau respons gerak yang diamati. Feature Jacobian dapat diperkirakan secara lokal untuk kontrol robot tertutup tanpa mengandalkan kalibrasi visual secara penuh. Salah satu permasalahan utama pada *uncalibrated image-based visual servoing* adalah memperoleh estimasi *image Jacobian* yang cukup baik untuk memetakan perubahan fitur citra terhadap gerak robot.

Istilah *model-free* pada konteks penelitian ini berarti pengendali visual tidak menggunakan model analitik lengkap kamera-robot untuk menghitung koreksi gerak. Pengendali memanfaatkan hubungan input-output yang diperoleh dari perubahan fitur citra setelah robot diberi gerak Cartesian tertentu. Namun, Karena hubungan visual-motor yang digunakan merupakan aproksimasi lokal, gerak yang terlalu besar dapat membuat kondisi aktual terlalu jauh dari daerah tempat Jacobian diperoleh. Selain itu, target dapat bergerak mendekati tepi citra atau bahkan keluar dari *field of view* (FOV). Oleh karena itu, IBVS memerlukan *image-space constraints* atau *visibility constraints* agar fitur yang dikendalikan tetap berada pada area citra yang dapat diamati.

Dengan $u_{min}, u_{max}, v_{min}, v_{max}$ sebagai batas citra dan m_u, m_v sebagai margin pengaman. Jika fitur mendekati batas FOV, pengendali dapat membatasi gerak yang berpotensi mendorong target keluar citra dan memprioritaskan koreksi yang membawa target kembali ke daerah aman.

Pada penelitian ini, konsep constraint tersebut diterapkan secara sederhana dalam bentuk *boundary guard*, *region of interest*, dan pembatasan besar langkah gerak. Implementasi tersebut bukan *Model Predictive Control* atau

optimasi constrained yang kompleks, tetapi tetap memiliki tujuan akademik yang sama, yaitu menjaga *visibility* target selama servoing

2.3.4 Empirical Local Image Jacobian

Image Jacobian menyatakan sensitivitas perubahan fitur citra terhadap perubahan gerak robot. Pada pendekatan analitik, Jacobian diturunkan dari model proyeksi kamera dan geometri sistem. Pada pendekatan empiris, elemen Jacobian diperoleh langsung dari respons citra akibat gerakan robot yang diketahui. Untuk koreksi pada sumbu Y dan Z, fitur citra dituliskan sebagai persamaan (u, v) . Sedangkan koreksi Cartesian yang digunakan adalah $\Delta \mathbf{q}_{YZ}$. Pada daerah gerak yang relatif kecil, hubungan keduanya dapat dilinierkan menjadi

$$\Delta \mathbf{s} \approx \mathbf{J}_{YZ} \Delta \mathbf{q}_{YZ}, \quad (2.21)$$

$$\mathbf{J}_{YZ} = \begin{bmatrix} \frac{\partial u}{\partial Y} & \frac{\partial u}{\partial Z} \\ \frac{\partial v}{\partial Y} & \frac{\partial v}{\partial Z} \end{bmatrix} \quad (2.22)$$

Kedua kolom tersebut kemudian disusun menjadi $\mathbf{J}_{YZ} = [\mathbf{J}_Y \ \mathbf{J}_Z]$. Estimasi Jacobian secara lokal dapat digunakan untuk *uncalibrated closed-loop control*. Disebut *local* karena nilai Jacobian merupakan pendekatan linier yang paling representatif di sekitar konfigurasi tempat data probe diperoleh. Ketika posisi kamera berubah jauh, hubungan citra dan gerak dapat berubah sehingga akurasi Jacobian juga dapat menurun.

Selain nilai elemen Jacobian, kondisi numeriknya perlu diperhatikan. Salah satu ukuran yang umum digunakan adalah *condition number*. Jika *singular value* terbesar dan terkecil dari Jacobian adalah σ_{max} dan σ_{min} , maka nilai κ yang relatif kecil menunjukkan Jacobian memiliki kondisi numerik yang baik. Sebaliknya, nilai yang sangat besar menunjukkan matriks mendekati kondisi singular atau *ill-conditioned*. Pada kondisi tersebut, perubahan kecil pada error atau *noise* citra dapat menghasilkan koreksi gerak yang terlalu besar ketika Jacobian diinverskan. Pada penelitian Chaumette juga

menekankan pentingnya memilih fitur dengan interaksi yang memiliki kondisi numerik baik untuk meningkatkan kestabilan dan ketahanan kontrol terhadap *noise*.

Dengan demikian, pada penelitian ini Jacobian empiris tidak hanya dipandang sebagai matriks konversi piksel-ke-gerak, tetapi sebagai pendekatan lokal yang perlu diperoleh pada kondisi pengamatan yang representatif dan diverifikasi kualitas numeriknya sebelum digunakan sebagai dasar *control law*.

2.3.5 Control law

Setelah error citra dan Jacobian lokal diperoleh, koreksi lateral Y-Z dapat dihitung menggunakan *control law* IBVS. Hubungan lokal jacobian empiris seperti di persamaan (). Setiap kolom Jacobian dapat diestimasi dengan memberikan gerakan kecil pada satu arah kemudian mengukur perubahan fitur citra:

$$J(:,j) \approx \frac{s(q + \Delta q_j) - s(q)}{\Delta q_j} \quad (2.23)$$

Setelah hubungan lokal diperoleh, perubahan gerak yang diperlukan untuk mengurangi error dapat diperoleh dari invers hubungan tersebut. Karena Jacobian dapat berbentuk tidak ideal atau terpengaruh *noise*, digunakan *Moore-Penrose pseudoinverse*, sehingga *control law* dituliskan sebagai

$$\Delta q_{YZ} = -\lambda J_{YZ}^{\dagger} e \quad (2.24)$$

Dengan $J_{YZ}^{\dagger}$ merupakan *pseudoinverse* Jacobian, λ sebagai gain control, dan tanda negatif menunjukkan bahwa gerak diberikan ke arah yang mengurangi error merupakan faktor pengali koreksi. Penggunaan *pseudoinverse* merupakan bentuk umum dalam IBVS karena memungkinkan solusi diperoleh walaupun pemetaan visual tidak selalu memiliki invers biasa yang ideal. Untuk mencegah robot terus melakukan koreksi akibat perubahan kecil

atau *noise*, digunakan daerah toleransi (*deadband*). Kondisi penyelarasan dapat dinyatakan sebagai:

$$|e_u| \leq \varepsilon_u, \quad |e_v| \leq \varepsilon_v \quad (2.25)$$

Metode *step-and-look* atau *look-and-move* menjalankan pengamatan dan pergerakan secara bergantian. Kamera memperoleh citra, fitur dan kesalahan dihitung, robot melakukan satu langkah koreksi, kemudian robot berhenti dan citra baru diambil untuk menghitung kesalahan berikutnya [17], [18]. Pola ini sesuai untuk sistem yang memerlukan pembaruan pengamatan setelah setiap perpindahan serta mengurangi pengaruh *motion blur* ketika aktuator bergerak.

Gain λ menentukan seberapa besar respons gerak terhadap error. Gain yang terlalu besar dapat mempercepat gerakan tetapi meningkatkan risiko *overshoot*, sedangkan gain yang terlalu kecil membuat konvergensi lambat. Untuk memperoleh respons yang lebih agresif ketika error masih besar dan lebih halus ketika mendekati *setpoint*, gain dapat diubah sebagai fungsi besar error. Salah satu bentuk *gain scheduling* yang sederhana adalah

$$\lambda(\|\mathbf{e}\|) = \lambda_{min} + (\lambda_{max} - \lambda_{min}) \min\left(\frac{\|\mathbf{e}\|}{e_g}, 1\right). \quad (2.26)$$

Dengan λ_{min} sebagai gain minimum di sekitar *setpoint*, λ_{max} sebagai gain maksimum, dan e_g sebagai skala error untuk transisi gain. Penggunaan *adaptive control gain* pada IBVS dapat memperbaiki laju dan domain konvergensi dibanding gain tetap. Dalam penelitian ini, istilah yang lebih tepat adalah *gain scheduling*, karena gain diubah berdasarkan besar error yang terukur dan bukan diestimasi menggunakan suatu hukum adaptasi parameter dinamis.

Selain gain, besar gerak setiap iterasi perlu dibatasi untuk menghindari gerakan Cartesian yang berlebihan. Pembatasan tersebut dapat dinyatakan dengan fungsi saturasi

$$\Delta q_i^{cmd} = \text{clip}(\Delta q_i, -\Delta q_{i,max}, \Delta q_{i,max}). \quad (2.27)$$

Dengan demikian, *control law* pada penelitian ini dapat dipahami sebagai kombinasi *pseudoinverse* Jacobian, *deadband*, *gain scheduling*, dan pembatasan langkah. Koreksi dihitung kembali setelah robot menyelesaikan satu gerakan dan citra baru diperoleh, sehingga proses berlangsung secara iteratif sampai error berada dalam toleransi yang ditentukan.

2.3.6 *Active monocular range*

. Kamera monokular menghasilkan citra dua dimensi sehingga satu *frame* tidak dapat memberikan jarak metrik objek secara langsung. Pada penelitian ini, informasi jarak longitudinal diperoleh dengan memanfaatkan perubahan ukuran proyeksi target setelah kamera bergerak maju dengan perpindahan yang diketahui. Pendekatan ini digunakan sebagai *active monocular range*, yaitu pemanfaatan gerakan kamera untuk memperoleh informasi tambahan mengenai kedekatan target.

Pada tahap awal pendekatan, gerak sumbu X masih ditentukan menggunakan informasi visual dari *instance mask*, terutama luas *mask* dan posisi batas bawah *mask*. Estimasi *active monocular range* digunakan terutama ketika target telah berada lebih dekat dengan kamera dan *mask* target sudah cukup besar serta stabil. Pada kondisi tersebut, perubahan skala *mask* akibat gerakan X yang diketahui digunakan untuk memperkirakan sisa jarak longitudinal sebelum proses pemetikan. Dengan demikian, metode ini berfungsi sebagai informasi pendekatan akhir dan bukan sebagai sensor kedalaman utama selama seluruh proses gerak.

Untuk target yang sama dengan perubahan orientasi yang relatif kecil, luas proyeksi objek pada citra meningkat ketika jarak kamera terhadap target berkurang. Secara ideal, hubungan luas *mask* A terhadap jarak Z dapat dituliskan sebagai:

$$A \propto \frac{1}{Z^2} \quad (2.28)$$

Jika luas *mask* sebelum gerakan adalah A_1 pada jarak Z_1 dan setelah kamera bergerak mendekati target luasnya menjadi A_2 pada jarak Z_2 , maka perubahan skala dapat dinyatakan sebagai:

$$r_A = \sqrt{\frac{A_2}{A_1}} = \frac{Z_1}{Z_2} \quad (2.29)$$

Apabila kamera bergerak menuju target sejauh d , maka hubungan jarak sebelum dan sesudah gerakan adalah:

$$Z_1 = Z_2 + d \quad (2.30)$$

Dengan mensubstitusikan Persamaan (2.38) ke Persamaan (2.37), estimasi jarak target setelah gerakan dapat dihitung sebagai:

$$Z_2 = \frac{d}{\sqrt{\frac{A_2}{A_1}} - 1} \quad (2.31)$$

dengan:

A_1 = luas *mask* target sebelum gerakan kamera (piksel²).

A_2 = luas *mask* target setelah gerakan kamera (piksel²).

Z_1 = jarak kamera terhadap target sebelum gerakan (mm).

Z_2 = estimasi jarak kamera terhadap target setelah gerakan (mm).

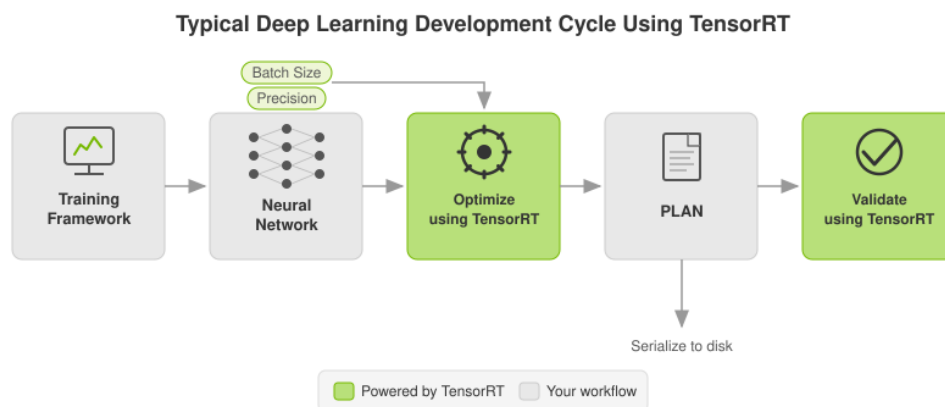
d = besar perpindahan kamera menuju target yang diketahui dari gerak robot (mm).

r_A = rasio perubahan skala proyeksi berdasarkan luas *mask*.

Nilai Z_2 pada Persamaan (2.39) tidak digunakan sebagai pengukuran kedalaman absolut yang berdiri sendiri. Pada sistem, nilai tersebut digunakan sebagai perkiraan jarak longitudinal ketika kondisi visual target telah memadai. Estimasi yang masih dianggap valid dapat dipertahankan selama pendekatan dan disesuaikan berdasarkan perpindahan X yang telah dilakukan.

2.4 Optimasi Inferensi Model

Inferensi pada perangkat *edge* perlu memperhatikan waktu proses dan penggunaan sumber daya. NVIDIA TensorRT merupakan perangkat lunak untuk mengoptimalkan dan menjalankan jaringan *deep learning* pada GPU NVIDIA. Model yang telah dilatih dapat dikonversi menjadi *engine* TensorRT sehingga eksekusi inferensi dapat dilakukan lebih efisien [20]. Salah satu optimasi yang digunakan adalah presisi FP16. FP16 menggunakan representasi bilangan 16 bit, sedangkan FP32 menggunakan 32 bit. Penggunaan FP16 dapat mengurangi kebutuhan memori dan mempercepat operasi pada perangkat yang mendukung Tensor Core. Setelah konversi, hasil inferensi tetap perlu dibandingkan untuk memastikan keluaran model masih sesuai. Dapat dilihat di **Gambar 2.4**.



Gambar 2.4 Alur pengembangan dan optimasi model menggunakan TensorRT

2.5 NVIDIA Jetson Orin Nano

NVIDIA Jetson Orin Nano Super Developer Kit merupakan perangkat *edge computing* yang dikembangkan untuk aplikasi robotika, visi komputer, dan pemrosesan AI secara langsung pada perangkat. Perangkat ini menggunakan arsitektur GPU NVIDIA Ampere yang dilengkapi dengan 1024 CUDA cores dan 32 Tensor cores, sehingga mampu menjalankan pemrosesan paralel untuk inferensi model *deep learning*. Perangkat ini memiliki performa AI hingga 67 INT8 TOPS, sehingga sesuai untuk menjalankan algoritma deteksi objek, segmentasi citra, dan pengolahan data visual secara *real-time* pada sistem tertanam [27]. Gambar modul dan pemetaan visual dari seluruh komponen utama NVIDIA Jetson Orin Nano Super Developer Kit 8GB terlihat pada **Gambar 2.5**[16]



Gambar 2.5 Board Diagram NVIDIA Jetson Orin Nano Super Developer Kit 8GB[17]

Dari **Gambar 2.5**, NVIDIA Jetson Orin Nano Super Developer Kit 8GB menggunakan CPU 6-core Arm Cortex-A78AE v8.2 64-bit dan GPU NVIDIA Ampere. Perangkat ini dilengkapi dengan RAM 8 GB LPDDR5 128-bit dengan bandwidth memori hingga 102 GB/s. Untuk penyimpanan, perangkat ini mendukung penggunaan kartu microSD serta penyimpanan eksternal berbasis NVMe SSD melalui slot M.2. NVIDIA Jetson Orin Nano Super Developer Kit 8GB juga memiliki beberapa antarmuka I/O, seperti empat port USB 3.2 Type-A, port USB-C, Gigabit Ethernet, DisplayPort, konektor kamera MIPI CSI, 40-pin expansion header, serta konektivitas nirkabel melalui modul M.2 Key-E. Spesifikasi detailnya terdapat pada **Tabel 2.2**.

Tabel 2.2 Spesifikasi Jetson

Spesifikasi	Keterangan
GPU	NVIDIA Ampere, 1024 CUDA Core, 32 Tensor Core
CPU	6-core Arm Cortex-A78AE 64-bit
Memori	8 GB LPDDR5
Penyimpanan	Dukungan microSD dan NVMe
Performa AI	Hingga 67 INT8 TOPS pada konfigurasi yang mendukung
Daya	7 W sampai 25 W

2.6 Orbbec Astra Pro

Kamera AstraPro merupakan kamera sensor 3D yang menggunakan teknologi *structured light* untuk menghasilkan citra warna (RGB) sekaligus peta kedalaman (*depth* mAP) secara simultan. Sensor ini memungkinkan sistem untuk

tidak hanya mendeteksi objek, tetapi juga mengetahui jarak absolut objek tersebut dari lensa kamera. Dengan resolusi tinggi dan latensi rendah, kamera ini sangat efektif digunakan dalam aplikasi estimasi posisi 3D dan navigasi robotika dalam ruangan[18]. Berikut gambar kamera pada **Gambar 2.6**. [18]



Gambar 2.6 AstraPro RGB Camera[18].

Spesifikasi teknis kamera memberikan gambaran mengenai kemampuan sensor dalam menangkap data visual dan kedalaman, seperti jarak deteksi, resolusi citra, *field of view* (FOV), serta kecepatan pemrosesan data. Parameter-parameter ini sangat penting dalam sistem yang memerlukan pemrosesan visual secara *real-time*. Selain itu, dukungan output point cloud dan sinkronisasi RGB–Depth juga mempermudah proses konversi koordinat dari citra 2D ke koordinat 3D yang dibutuhkan dalam proses penentuan posisi target. Spesifikasi utama dari kamera yang digunakan dalam penelitian ini ditunjukkan pada **Tabel 2.3**.

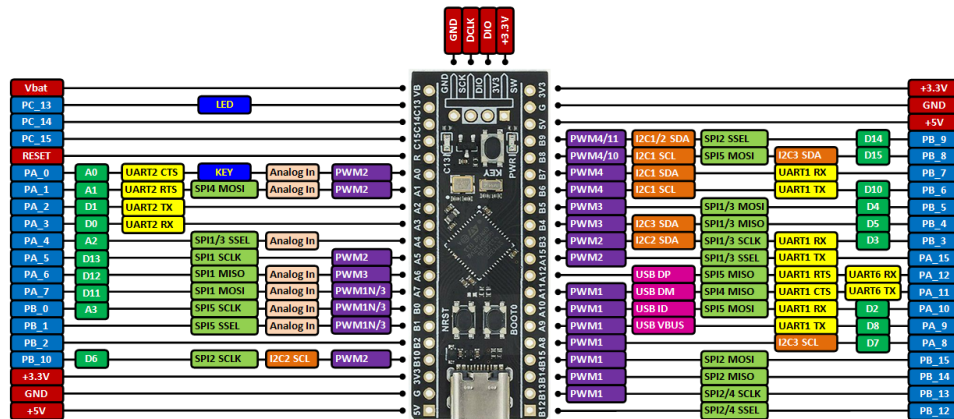
Tabel 2.3 Spesifikasi Astra RGB Camera

Spesifikasi	Keterangan
Resolusi RGB	Hingga 1280×720 pada 30 FPS
Field of View	Sekitar 60° horizontal, $49,5^\circ$ vertikal, dan 73° diagonal
Antarmuka	USB 2.0
Dimensi	Sekitar $165 \times 30 \times 40$ mm
Keluaran sensor	Citra RGB dan depth
Tegangan Kerja	5V DC
Arus Kerja	$< 0,5$ A

2.7 STM32F411CEU6

STM32F411CEU6, atau dikenal sebagai "*Black Pill*", adalah mikrokontroler berbasis ARM Cortex-M4 dengan kecepatan hingga 100 MHz yang memiliki

efisiensi daya tinggi. Perangkat ini sering digunakan sebagai pengontrol tingkat rendah (*low-level controller*) untuk mengelola aktuator dan membaca sensor tambahan dalam sistem robotika. Memori *flash* dan RAM yang cukup besar memungkinkan eksekusi program kontrol yang kompleks dengan komunikasi serial yang stabil[19].



Gambar 2.7 Pinout STM32F411CEU6[19]

Dari **Gambar 2.7** terlihat diagram *pinout* STM32F411CEU6 yang berfungsi sebagai panduan untuk memahami konektivitas serta fungsi masing-masing pin pada mikrokontroler. Diagram ini membantu dalam proses perancangan rangkaian dengan menunjukkan konfigurasi GPIO yang dapat digunakan sebagai input maupun output digital, serta pin yang mendukung fungsi PWM, ADC, dan komunikasi serial seperti UART, SPI, dan I2C. Pada sistem tugas akhir ini, STM32F411CEU6 digunakan untuk mengendalikan aktuator motor serta menangani komunikasi data dengan sistem utama. Spesifikasi utama dari mikrokontroler STM32F411CEU6 ditunjukkan pada **Tabel 2.4**.

Tabel 2.4 Spesifikasi STM32F411CEU6

Spesifikasi	Keterangan
CPU	ARM Cortex-M4 32-bit, 100 MHz
Tegangan Operasi	3.3 V
Arus Operasi	±30–50 mA
Flash Memory	512 KB
SRAM	128 KB
GPIO	50 pin
PWM	12 pin

Spesifikasi	Keterangan
ADC	16 <i>channel</i> (12 bit)
Komunikasi	UART, SPI, I2C

2.8 Stepdown Converter XL4016

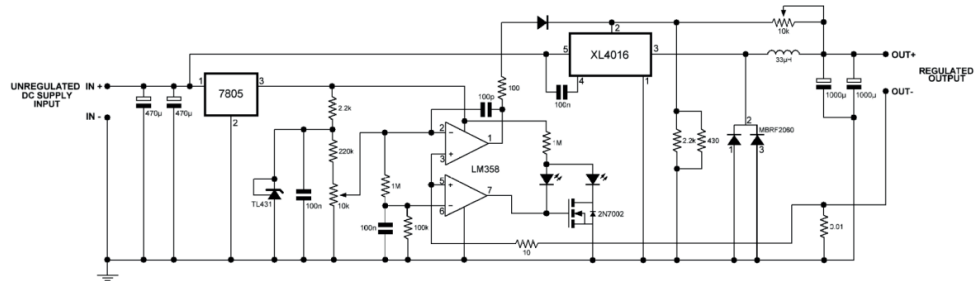
Stepdown XL4016 adalah modul regulator tegangan DC-to-DC jenis *buck converter* yang mampu menurunkan tegangan *input* sekitar 4V hingga 40V, sedangkan tegangan *output* dapat diatur pada kisaran 1,25 V hingga 36 V melalui potensiometer yang tersedia pada modul. Modul ini mampu menangani arus keluaran hingga sekitar 8–10 A dengan daya maksimum hingga sekitar 300 W. Kemampuan ini berperan penting dalam mendistribusikan daya dari baterai utama ke komponen elektronik sensitif seperti Raspberry Pi dan mikrokontroler. Efisiensi konversi yang dapat mencapai sekitar 96%, membantu menjaga suhu operasional tetap rendah selama penggunaan jangka panjang[20]. Gambar modul dapat dilihat pada **Gambar 2.8**.



Gambar 2.8 Modul Stepdown XL4016[20]

Komponen utama modul merupakan regulator *switching* sebagai pengatur tegangan utama berfrekuensi tinggi. Selain itu, terdapat regulator 7805 yang menyediakan tegangan 5V untuk rangkaian pendukung seperti LM358 sebagai komparator dalam sistem umpan balik tegangan. Komponen TL431 berperan sebagai referensi tegangan presisi yang untuk mengontrol kestabilan tegangan keluaran. Dioda berkecepatan tinggi sebagai penyearah dalam proses *switching* untuk menjaga efisiensi konversi daya. Induktor berperan dalam proses penyimpanan dan transfer energi. Sementara itu, kapasitor pada sisi input dan

output digunakan sebagai filter untuk mengurangi riak tegangan serta menjaga kestabilan tegangan keluaran modul. Gambar skematik dapat dilihat pada **Gambar 2.9**.



Gambar 2.9 Skematik Modul Stepdown XL4016[20]

Dengan efisiensi konversi yang tinggi, modul ini juga membantu menjaga suhu operasional tetap rendah selama penggunaan dalam waktu yang lama. Spesifikasi lengkap modul *Stepdown* XL4016 dapat dilihat pada **Tabel 2.5**.

Tabel 2.5 Spesifikasi Stepdown XL4016

Spesifikasi	Keterangan
IC Regulator	XL4016
Tegangan Input	4 – 40 VDC
Tegangan Output	1,25 – 35 VDC
Arus Output Maksimal	10 A
Frekuensi Switching	180 kHz
Efisiensi Konversi	Hingga 96%
Daya Maksimal	300 W
Tegangan Input	4 – 40 VDC

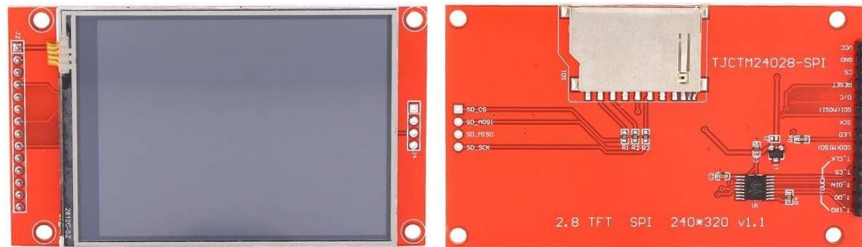
Persamaan efisiensi pada persamaan (2.16) digunakan untuk mengetahui seberapa besar daya masukan yang berhasil dikonversi menjadi daya keluaran.

$$\eta = \frac{V_{out} \times I_{out}}{V_{in} \times I_{in}} \times 100\% \quad (2.16)$$

Pada persamaan tersebut, η menyatakan efisiensi sistem dalam satuan persen. P_{out} merupakan daya listrik yang dihasilkan pada sisi keluaran konverter yang diperoleh dari hasil perkalian antara V_{out} (tegangan keluaran) dan I_{out} (arus keluaran), sedangkan P_{in} merupakan daya listrik yang masuk ke dalam sistem sebelum proses konversi terjadi.

2.9 TFT LCD Module

TFT LCD atau *Thin Film Transistor Liquid Crystal Display* merupakan jenis layar LCD yang menggunakan teknologi transistor film tipis untuk menghasilkan tampilan gambar dengan kualitas warna yang baik dan respons yang cepat. TFT LCD 2.8 inch dapat digunakan sebagai antarmuka pengguna grafis atau *Graphical User Interface* (GUI) untuk menampilkan informasi dalam bentuk teks, gambar, tombol, maupun indikator visual. Modul ini juga dilengkapi dengan fitur layar sentuh sehingga pengguna dapat memberikan input perintah secara langsung melalui tampilan layar. Gambar TFT LCD *Touchscreen* 2.8 inch dapat dilihat pada **Gambar 2.10**.



Gambar 2.10 TFT LCD *Touchscreen* 2.8 inch

Komunikasi antara TFT LCD dan mikrokontroler menggunakan protokol SPI (*Serial Peripheral Interface*). SPI merupakan komunikasi serial sinkron yang menggunakan jalur SCK sebagai *clock*, MOSI (*Master Out Slave In*) untuk mengirim data dari mikrokontroler ke layar, MISO (*Master In Slave Out*) untuk membaca data dari modul, serta CS (*Chip Select*) sebagai pemilih perangkat.. Berikut spesifikasi lebih jelas pada **Tabel 2.6**.

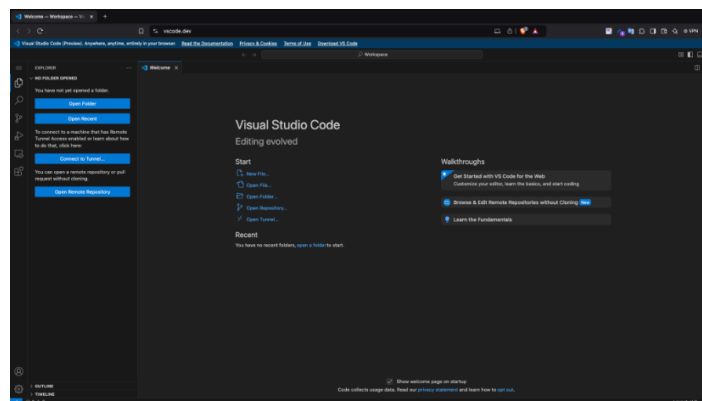
Tabel 2.6 TFT LCD *Touchscreen* 2.8 inch

Spesifikasi	Keterangan
Ukuran Layar	2.8 inci
Resolusi	240 × 320 piksel
Jenis Touchscreen	<i>Resistive Touch</i>
Driver Display	ILI9341
<i>Controller Touchscreen</i>	XPT2046
Tegangan Operasi	3.3 V / 5 V
Komunikasi	SPI
Warna Tampilan	RGB 565

2.10 Perangkat Lunak

2.10.1 Visual Studio Code

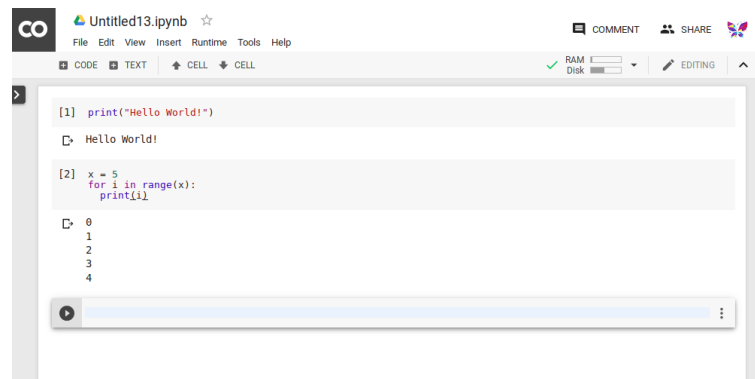
Visual Studio Code adalah editor kode sumber yang ringan namun bertenaga dengan dukungan ekstensif untuk berbagai bahasa pemrograman, termasuk Python dan C++. Fitur seperti *IntelliSense* dan *debugging* terintegrasi mempercepat proses pengembangan perangkat lunak pada proyek robotika. Fleksibilitasnya melalui ekstensi memungkinkan pengembang untuk mengelola kode dan melakukan kontrol versi secara efisien. Terlihat tampilannya pada **Gambar 2.11.[15]**



Gambar 2.11 Tampilan Visual Studio Code[15]

2.10.2 Google Colab

Google Colab merupakan platform komputasi berbasis cloud yang dikembangkan oleh Google untuk menjalankan kode pemrograman Python secara interaktif melalui browser tanpa memerlukan instalasi perangkat lunak tambahan pada komputer lokal. Google Colab menyediakan lingkungan kerja berbasis notebook yang mendukung berbagai library machine learning dan *deep learning* seperti PyTorch dan TensorFlow, serta dilengkapi dengan dukungan akselerasi perangkat keras seperti GPU dan TPU. Berikut tampilannya pada **Gambar 2.12.[21]**



Gambar 2.12 Tampilan Google Colab[21]

2.10.3 Roboflow

Roboflow adalah platform manajemen data yang menyederhanakan proses penandaan (*labeling*), augmentasi, dan manajemen *dataset* untuk penglihatan komputer. Platform ini memungkinkan konversi format *dataset* secara cepat agar sesuai dengan kebutuhan pelatihan model YOLOv8. Penggunaan Roboflow membantu meningkatkan kualitas data latih melalui teknik augmentasi citra yang terstandarisasi[22]. Berikut tampilan Roboflow pada **Gambar 2.13**.



Gambar 2.13 Tampilan proses anotasi dataset di Roboflow[23]