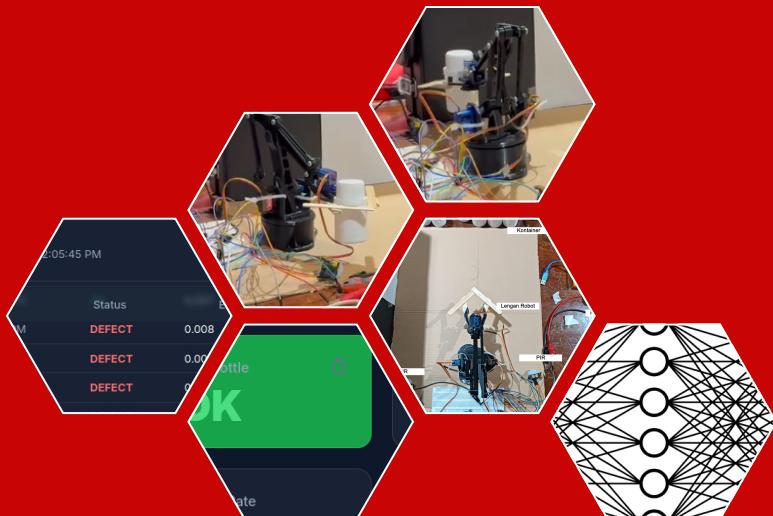


**PROTOTIPE SISTEM DETEKSI DAN PENANGANAN CACAT PADA
KONTAINER KIMIA UNTUK AREA INDUSTRI-TERISOLASI
BERBASIS ARTIFICIAL INTELLIGENCE (AI) DAN LENGAN ROBOT**



**ALRIDHO
H021211006**



**PROGRAM STUDI FISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN
ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2025**

**PROTOTIPE SISTEM DETEKSI DAN PENANGANAN CACAT PADA
KONTAINER KIMIA UNTUK AREA INDUSTRI-TERISOLASI
BERBASIS ARTIFICIAL INTELLIGENCE (AI) DAN LENGAN ROBOT**

**ALRIDHO
H021211006**



**PROGRAM STUDI FISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2025**

**PROTOTIPE SISTEM DETEKSI DAN PENANGANAN CACAT PADA
KONTAINER KIMIA UNTUK AREA INDUSTRI-TERISOLASI
BERBASIS ARTIFICIAL INTELLIGENCE (AI) DAN LENGAN ROBOT**

ALRIDHO
H021211006

Skripsi

sebagai salah satu syarat untuk mencapai gelar sarjana

Program Studi Fisika

pada

**PROGRAM STUDI FISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2025**

SKRIPSI

**PROTOTIPE SISTEM DETEKSI DAN PENANGANAN CACAT PADA
KONTAINER KIMIA UNTUK AREA INDUSTRI-TERISOLASI
BERBASIS ARTIFICIAL INTELLIGENCE (AI) DAN LENGAN ROBOT**

**ALRIDHO
H021211006**



telah dipertahankan di depan Panitia Ujian Sarjana Xxxx pada tanggal bulan
tahun dan dinyatakan telah memenuhi syarat kelulusan
pada

Program Studi Fisika
Departemen Fisika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Hasanuddin
Makassar

Mengesahkan:
Pembimbing tugas akhir,

Prof. Dr. Arifin, M.T
NIP. 19670520 199403 1 002

Mengetahui:
Ketua Program Studi,

Prof. Dr. rer-nat Wira Bahari Nurdin
NIP. 19670923 199003 1 001

PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya menyatakan bahwa, skripsi berjudul "**PROTOTIPE SISTEM DETEKSI DAN PENANGANAN CACAT PADA KONTAINER KIMIA UNTUK AREA INDUSTRI TERISOLASI BERBASIS ARTIFICIAL INTELLIGENCE (AI) DAN LENGAN ROBOT**" adalah benar karya saya dengan arahan dari pembimbing (Prof. Dr. Arifin, M.T). Karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuk apapun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka skripsi ini. Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini adalah karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut berdasarkan aturan yang berlaku.

Dengan ini saya melimpahkan hak cipta (hak ekonomis) dari karya tulis saya berupa skripsi ini kepada Universitas Hasanuddin.

Makassar, 22 Juli 2025

Alridho
H021211006

UCAPAN TERIMA KASIH

Segala puji dan syukur penulis panjatkan kehadirat Allah Subhanahu Wa Ta'ala atas segala berkah, rahmat, dan karunia-Nya yang telah memberikan ilmu pengetahuan, pengalaman, kekuatan, kesabaran, dan kesempatan kepada penulis sehingga mampu menyelesaikan skripsi ini. Penulisan skripsi yang berjudul "**PROTOTIPE SISTEM DETEKSI DAN PENANGANAN CACAT PADA KONTAINER KIMIA UNTUK AREA INDUSTRI-TERISOLASI BERBASIS ARTIFICIAL INTELLIGENCE (AI) DAN LENGAN ROBOT**" merupakan upaya penulis memenuhi salah satu syarat dalam menyelesaikan pendidikan dan memperoleh gelar Sarjana Sains di Departemen Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin. Selain itu, skripsi ini juga diharapkan dapat memberikan manfaat bagi pembaca dan peneliti lain untuk menambah wawasan dalam bidang fisika khususnya elektronika dan instrumentasi.

Proses penyelesaian skripsi ini merupakan suatu rangkaian perjuangan yang cukup panjang bagi penulis. Selama proses penelitian maupun penyusunan skripsi ini, tidak sedikit hambatan maupun kendala yang penulis hadapi. Do'a dan dukungan dari berbagai pihak merupakan hal yang berarti, sehingga penyusunan skripsi ini dapat diselesaikan oleh penulis. Oleh karena itu, dengan tulus dan ikhlas, penulis mengucapkan terima kasih sebanyak-banyaknya.

Penulis menyampaikan penghargaan setinggi-tingginya dan banyak terima kasih kepada Bapak **Prof. Dr. Arifin, M.T** selaku pembimbing saya atas kesedianya telah meluangkan banyak waktu, tenaga dan pikiran dalam memberikan bimbingan dan motivasi kepada Penulis, mulai dari awal penyusunan sampai penyelesaian skripsi ini. Pada kesempatan ini pula, dengan segala kerendahan hati Penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Keluarga tercinta, khususnya kedua orang tua saya, Ayahanda **Muh. Anto** dan Ibunda **Nuraeni**, yang telah memberikan dukungan materiil dan moril serta tak henti mendoakan. Terima kasih juga untuk adik saya, Cinta Kasih Rahmadany, yang turut memberi semangat dalam menyelesaikan perjalanan ini.
2. Dosen penguji kak **Ida Laila, S.Si., M.Si.** dan Ibu **Prof. Dr. Sri Suryani, DEA.** atas kesediaannya untuk meluangkan waktu, menguji, serta memberikan kritik dan saran yang sangat berharga untuk skripsi ini.
3. Bapak dan Ibu **Dosen Pengajar Departemen Fisika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin**, terima kasih atas ilmu dan bimbingannya selama ini. Semoga hasil ajaran Bapak dan Ibu selalu memberikan manfaat bagi setiap orang.
4. Bapak dan Ibu Staf Departemen Fisika khususnya **Pak Syukur**, Ibu **Evi**, dan Kak **Rana** yang selalu membantu penulis selama berada di kampus.
5. Teman-teman seperjuangan di kampus **Ragil, Vivaldo, Salim, Werdi, Adri, Akmal, dan lain-lain** atas kebersamaannya di dalam dan di luar kampus.

6. Seluruh pihak yang telah membantu dan memberikan dukungan yang tidak dapat penulis sebutkan namanya satu per satu.

Penulis berharap semoga segala kebaikan yang diberikan dari berbagai pihak kepada penulis untuk menyelesaikan skripsi ini dapat bernilai ibadah. Akhir kata, penulis memohon maaf atas kesalahan yang disengaja maupun tidak disengaja dalam rangkaian penyusunan skripsi ini. Semoga skripsi ini dapat memberikan manfaat bagi pembaca.

Makassar, 22 Juli 2025

Alridho
H021211006

ABSTRAK

Alridho. **PROTOTIPE SISTEM DETEKSI DAN PENANGANAN CACAT PADA KONTAINER KIMIA UNTUK AREA INDUSTRI-TERISOLASI BERBASIS ARTIFICIAL INTELLIGENCE (AI) DAN LENGAN ROBOT** (dibimbing oleh Prof. Dr. Arifin, M.T.).

Latar Belakang. Inspeksi manual kontainer kimia memiliki tingkat kesalahan tinggi (20%–30%) dan berisiko bagi pekerja. *Artificial Intelligence* (AI) dan robotika dapat meningkatkan efisiensi dan keselamatan. Model deteksi cacat berbasis *autoencoder* efektif karena bisa belajar dari data normal untuk mendeteksi anomali, sangat membantu saat data cacat terbatas. **Tujuan.** Penelitian ini bertujuan: (1) merancang model deteksi objek berbasis YOLO untuk mengenali kontainer, (2) membangun model deteksi cacat menggunakan CVAE, dan (3) mengintegrasikan keduanya dalam sistem lengan robot untuk identifikasi dan penyortiran otomatis. **Metode.** Penelitian menggunakan pendekatan eksperimental dengan prototipe yang terdiri dari lengan robot, Arduino Uno, motor servo, sensor PIR, dan kamera. Kamera menangkap citra kontainer yang kemudian dideteksi YOLO. CVAE menganalisis cacat berdasarkan *reconstruction error*, dengan ambang 0,007183 (dari kurva ROC). Lengan robot lalu menyortir kontainer ke area "cacat" atau "normal". **Hasil.** Model YOLO mencapai mAP@50 dan mAP@50-95 mendekati 1.00. Model CVAE berhasil membedakan kontainer cacat dan normal. Pengujian 25 kali menunjukkan akurasi 100%, dan seluruh proses ditampilkan *real-time* berbasis web. **Kesimpulan.** Prototipe ini berhasil mendemonstrasikan otomasi penuh inspeksi visual di lingkungan industri, dengan akurasi tinggi dan sistem penyortiran robotik yang terintegrasi.

Kata Kunci: *Deteksi cacat; prototipe; lengan robot; YOLO; autoencoder; IoT.*

ABSTRACT

Alridho. **AI-DRIVEN PROTOTYPE FOR DEFECT DETECTION AND HANDLING OF CHEMICAL CONTAINERS IN ISOLATED INDUSTRIAL ENVIRONMENTS WITH A ROBOTIC ARM** (dibimbing oleh Prof. Dr. Arifin, M.T.).

Background. Manual inspection of chemical containers has a high error rate (20%–30%) and poses risks to workers. Artificial Intelligence (AI) and robotics can improve efficiency and safety. Defect detection models based on autoencoders are effective since they can learn from normal data to detect anomalies, which is very useful when defective data are limited. **Objective.** This study aims to: (1) design an object detection model based on YOLO to accurately recognize containers, (2) develop a defect detection model using CVAE, and (3) integrate both models into a robotic arm system for automatic identification and sorting. **Method.** The research uses an experimental approach with a prototype consisting of a robotic arm, Arduino Uno, servo motors, PIR sensor, and camera. The camera captures images of the containers, which are then detected by YOLO. The CVAE analyzes defects based on reconstruction error, with a threshold of 0.007183 (from the ROC curve). The robotic arm then sorts the containers into "defect" or "normal" areas. **Results.** The YOLO model achieved mAP@50 and mAP@50-95 close to 1.00. The CVAE model successfully distinguished defective and normal containers. Testing conducted 25 times showed 100% accuracy, and the entire process was displayed in real-time via a web-based interface. **Conclusion.** This prototype successfully demonstrated full automation of visual inspection in an industrial environment, with high accuracy and an integrated robotic sorting system.

Keywords: *Defect detection; prototype; robotic arm; YOLO; autoencoder; IoT.*

DAFTAR ISI

| | Halaman |
|---|---------|
| HALAMAN JUDUL | i |
| HALAMAN PENGAJUAN | ii |
| HALAMAN PENGESAHAN | iii |
| LEMBAR PERNYATAAN KEASLIAN SKRIPSI | iv |
| UCAPAN TERIMA KASIH | v |
| ABSTRAK | vii |
| ABSTRACT | viii |
| DAFTAR ISI | ix |
| DAFTAR TABEL | x |
| DAFTAR GAMBAR | xi |
| DAFTAR LAMPIRAN | xii |
| BAB I PENDAHULUAN | 1 |
| 1.1 Latar Belakang | 1 |
| 1.2 Tujuan Penelitian | 3 |
| 1.3 Manfaat Penelitian | 3 |
| BAB II METODE PENELITIAN | 4 |
| 2.1 Tempat dan Waktu Penelitian | 4 |
| 2.2 Peralatan Penelitian | 4 |
| 2.3 Metode Kerja | 4 |
| 2.3.1 Perancangan <i>Hardware</i> | 6 |
| 2.3.2 Perancangan <i>Software</i> | 7 |
| 2.3.3 Bagan Alir Sistem Kerja Alat | 9 |
| BAB III HASIL DAN PEMBAHASAN | 11 |
| 3.1 Hasil Perancangan Sistem | 11 |
| 3.2 Hasil Perancangan Model YOLO | 12 |
| 3.2.1 <i>Dataset</i> dan <i>Preprocessing</i> | 12 |
| 3.2.2 Hasil Pelatihan Model | 13 |
| 3.3 Hasil Perancangan Model Deteksi Kecacatan | 16 |
| 3.3.1 Arsitektur Variational Autoencoder | 16 |
| 3.3.2 <i>Dataset</i> dan <i>Preprocessing</i> | 16 |
| 3.3.3 Hasil Pelatihan Model | 17 |
| 3.3.4 Penentuan Ambang Batas Kecacatan | 19 |
| 3.4 Hasil Pengujian Sistem Lengan Robot | 21 |
| BAB IV KESIMPULAN | 26 |
| 4.1 Kesimpulan | 26 |
| 4.2 Saran | 26 |
| DAFTAR PUSTAKA | 26 |
| LAMPIRAN | 29 |

DAFTAR TABEL

| | Halaman |
|---|----------------|
| 1 Distribusi pembagian <i>dataset</i> | 12 |
| 2 Proses <i>training</i> model YOLO | 13 |
| 3 Proses <i>training</i> model model <i>convolutional variational autoencoder</i> . . | 18 |
| 4 Nilai <i>error</i> untuk setiap sampel | 20 |
| 5 Nilai <i>error</i> untuk setiap sampel uji | 24 |

DAFTAR GAMBAR

| | Halaman |
|---|---------|
| 1 Bagan alir penelitian | 5 |
| 2 Rangkaian <i>hardware</i> sistem | 6 |
| 3 Rancang sistem <i>hardware</i> | 7 |
| 4 Diagram alur pelatihan model YOLO | 8 |
| 5 Diagram alur pelatihan model deteksi cacat | 9 |
| 6 Bagan alir sistem kerja alat | 10 |
| 7 Hasil perancangan sistem | 11 |
| 8 Salah satu data <i>train</i> dengan labelnya | 12 |
| 9 Grafik tren mAP: (a) mAP@50, (b) mAP@50-95 | 14 |
| 10 Hasil prediksi YOLO pada data validasi | 15 |
| 11 Arsitektur <i>variational autoencoder</i> | 16 |
| 12 Salah satu data latih model <i>autoencoder</i> | 17 |
| 13 Hasil prediksi kecacatan: (a) Kontainer normal, (b) kontainer cacat | 19 |
| 14 Kurva ROC | 21 |
| 15 Lengan robot menyortir kontainer: (a) normal, (b) cacat | 22 |
| 16 Tampilan hasil prediksi melalui <i>website</i> : (a) kontainer normal, (b) kontainer cacat | 23 |
| 17 Hasil prediksi kecacatan pada 25 sampel | 25 |

DAFTAR LAMPIRAN

| | Halaman |
|---|---------|
| Lampiran 1. Peralatan penelitian | 30 |
| Lampiran 2. Program untuk latih model YOLO | 31 |
| Lampiran 3. Program untuk latih model autoencoder | 32 |
| Lampiran 4. Program kombinasi YOLO dan autoencoder | 36 |
| Lampiran 5. Program lengan robot | 43 |
| Lampiran 6. Dokumentasi pergerakan lengan robot | 45 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Seiring dengan pesatnya perkembangan AI, robotika, dan otomatisasi, penggunaan lengan robot otomatis semakin meluas di sektor manufaktur, logistik, dan layanan (Jhang et al., 2024). Otomatisasi bertujuan untuk menjalankan rangkaian tindakan sesuai dengan proses yang telah ditetapkan tanpa intervensi manusia dengan mengendalikan perangkat mekanis secara otomatis (Oaki et al., 2023). Lengan robot mampu melakukan berbagai tugas seperti perakitan, penanganan, dan pengemasan sehingga dapat mengantikan pekerjaan yang bersifat berulang dan melelahkan bagi manusia (Lin et al., 2024). Dalam industri manufaktur, lengan robot sering dikombinasikan dengan sensor kamera, algoritma visi komputer, dan teknologi otomatisasi guna mendeteksi cacat pada objek dengan tingkat presisi dan efisiensi yang tinggi.

Dalam konteks industri, terutama pada penanganan material berbahaya seperti kontainer kimia, aspek keselamatan menjadi prioritas utama. Inspeksi visual otomatis dengan bantuan robot diperlukan untuk mengurangi risiko paparan zat berbahaya terhadap manusia. Isolasi dalam proses deteksi cacat pada kontainer kimia sangat penting, karena kerusakan atau kontaminasi pada kontainer dapat menimbulkan ancaman keselamatan bagi konsumen. Truong et al. (2024) melaporkan bahwa kesalahan inspeksi manual berkisar 20% hingga 30% yang disebabkan oleh beberapa faktor seperti kelelahan, stres, pencahayaan yang tidak memadai, dan kurangnya pengalaman. Peningkatan penggunaan robot dalam industri juga berdampak positif terhadap produktivitas. Graetz et al. (2018) menemukan bahwa otomatisasi meningkatkan produktivitas tenaga kerja sebesar 0,36% per tahun, serta menurunkan harga produksi. Gihleb et al. (2022) melaporkan bahwa antara 2005 dan 2011, jumlah robot per 1.000 pekerja meningkat sebesar 25%, yang berkontribusi terhadap penurunan 72.658 kasus cedera kerja per tahun dengan potensi penghematan biaya mencapai sekitar Rp27,3 triliun per tahun, atau total sekitar Rp191,2 triliun selama periode tersebut. Oleh karena itu, penerapan metode otomatis yang tepat dalam mendeteksi cacat dan kontaminasi sangat krusial untuk menjamin keselamatan kerja, mencapai skalabilitas sistem, dan meningkatkan efisiensi biaya.

Mayoritas sistem model deteksi cacat saat ini menggunakan metode *supervised learning* seperti algoritma *You Only Look Once* (YOLO). Zhuo dan Zhao (2025) mengusulkan model berbasis YOLOv8 dengan integrasi modul *Multi-Path Convolution Attention* (MPCA) dan *Partial Self-Attention* (PSA) untuk meningkatkan akurasi dan sensitivitas dalam mendeteksi cacat permukaan baja yang telah diuji pada dataset publik *Northeastern University-Defect Detection* (NEU-DET) dan VOC2007. Dong et al. (2025) mengembangkan model YOLO untuk mendeteksi cacat pada komponen *magnet tile motor* kendaraan listrik dengan cepat dan akurat. Liu et al. (2025) juga mengusulkan model YOLO yang dioptimalkan untuk mendeteksi cacat kecil pada ngecoran logam dengan menggunakan modul deteksi objek kecil, fitur re-ekstraksi,

dan *multi-scale attention*, serta diuji menggunakan *dataset* NEU-DET. Semua pendekatan tersebut menggunakan *dataset* berlabel, sehingga memerlukan data cacat yang sudah ditandai secara manual sebagai prasyarat pelatihan.

Sebagai alternatif, algoritma *deep learning* berupa *autoencoder* dapat dimanfaatkan untuk mendeteksi anomali dan cacat permukaan tanpa memerlukan *dataset* berlabel (Tsai et al., 2021). Model *autoencoder*, khususnya *convolutional autoencoder* (CAE), dapat dilatih hanya dengan sampel data bebas cacat (Chen et al., 2021). CAE bekerja dengan meminimalkan kesalahan rekonstruksi sehingga fitur-fitur representatif dari objek dapat teridentifikasi secara optimal. Karena model hanya dilatih menggunakan data normal, setiap sampel yang mengandung cacat akan menunjukkan deviasi signifikan saat proses evaluasi (Liu et al., 2025). Pendekatan ini sangat berguna ketika pengumpulan data sulit atau mahal untuk dikumpulkan.

YOLO juga telah banyak digunakan bersama lengan robot dalam berbagai aplikasi. Wang et al. (2024) mengembangkan sistem kontrol lengan robot berbasis YOLO yang efisien untuk menggenggam komponen logam secara presisi di lingkungan industri yang kompleks. Kato et al. (2022) mengusulkan metode yang menggabungkan YOLO dan *Convolutional Neural Network* (CNN) untuk mendeteksi objek parsial dan memperkirakan kedalaman serta mengontrol lengan robot dengan 4 derajat kebebasan. Kim et al. (2021) menerapkan YOLO dalam metode genggam robot untuk deteksi dan pemilahan sampah secara *real-time*, dengan pembatasan area pasca deteksi objek. Jin et al. (2025) mengembangkan YOLOv8n yang telah ditingkatkan dengan integrasi modul *dilated re-parameterization*, *feature pyramid*, dan *Scylla-IoU loss* yang terbukti meningkatkan akurasi dan adaptabilitas lengan robot dalam memetik apel di kebun yang kompleks.

Sementara itu, *autoencoder* sebagai model jaringan saraf dilatih untuk merekonstruksi data *input* dan digunakan dalam deteksi cacat dengan membandingkan perbedaan signifikan antara *input* normal dan hasil rekonstruksi. Bionda et al. (2022) menggunakan *autoencoder* dengan fungsi *loss* berbasis *Complex Wavelet Structural Similarity* (CW-SSIM) untuk mendeteksi anomali citra industri, yang terbukti efisien dibandingkan model *deep learning* lainnya. Yun et al. (2023) mengembangkan *autoencoder* untuk mendeteksi kerusakan pada lengan robot berdasarkan sinyal suara internal, menggunakan citra spektrogram *Short-Time Fourier Transform* (STFT). Jia et al. (2023) mengusulkan model deteksi anomali dengan arsitektur *encoder-decoder-encoder* (EDE) dan pelatihan dua tahap, yang menggabungkan pendekatan rekonstruksi dan konfrontasi generatif. Kozamernik et al. (2025) mengembangkan *Fuse-Decode autoencoder*, dengan strategi pembelajaran bertahap mulai dari tanpa supervisi, semi-supervisi, dan supervisi campuran, serta menunjukkan kinerja unggul dalam deteksi cacat pada data industri nyata dan *dataset* MVTec Anomaly Detection (AD). Ruediger-Flore et al. (2024) membandingkan klasifikasi biner dan *autoencoder* dalam deteksi anomali citra untuk proses perakitan rangka, khususnya pada kesalahan posisi dan rotasi komponen, dan menyimpulkan bahwa *autoencoder* lebih efektif untuk deteksi anomali halus dan fleksibel dalam kondisi data yang terbatas.

Kebutuhan untuk meningkatkan efisiensi dan keselamatan dalam proses inspeksi industri mendorong pemanfaatan AI dan robotika. Deteksi cacat otomatis, terutama dalam penanganan material berbahaya, sangat penting untuk menjamin kualitas produk dan keselamatan manusia. *Autoencoder* merupakan algoritma pembelajaran mesin yang efektif karena kemampuannya untuk belajar dari data normal dan mengidentifikasi anomalis secara mandiri. Dengan demikian, penelitian ini bertujuan untuk mengeksplorasi dan mengoptimalkan penerapan *autoencoder* dalam mendeteksi cacat permukaan, sehingga sehingga dapat mendukung pengembangan manufaktur cerdas yang efisien dan berkelanjutan.

1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Merancang dan melatih model deteksi objek berbasis YOLO yang mampu mengenali kontainer kimia secara akurat dalam berbagai lingkungan.
2. Membangun model deteksi kecacatan menggunakan algoritma *convolutional autoencoder* yang efektif dalam membedakan antara kontainer cacat dan tidak.
3. Mengintegrasikan model deteksi objek dan deteksi kecacatan ke dalam sistem berbasis lengan robot untuk proses identifikasi dan penyortiran otomatis.

1.3 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini antara lain:

1. Memberikan kontribusi terhadap pengembangan sistem deteksi objek yang cepat dan akurat di lingkungan industri, khususnya untuk inspeksi visual kontainer kimia dalam kondisi nyata yang bervariasi.
2. Menyediakan solusi yang efektif dalam mendeteksi kecacatan secara otomatis, sehingga dapat menggantikan metode inspeksi manual yang memakan waktu dan rentan terhadap kesalahan manusia.
3. Mendorong otomatisasi penuh dalam proses identifikasi dan penyortiran kontainer kimia, sehingga meningkatkan efisiensi produksi, menurunkan biaya operasional, dan meminimalkan risiko kesalahan klasifikasi dalam sistem manufaktur.

BAB II

METODE PENELITIAN

2.1 Tempat dan Waktu Penelitian

Penelitian ini dilaksanakan dari bulan Februari 2025 hingga Juni 2025, bertempat di Laboratorium Elektronika dan Instrumentasi, Departemen Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin, Makassar.

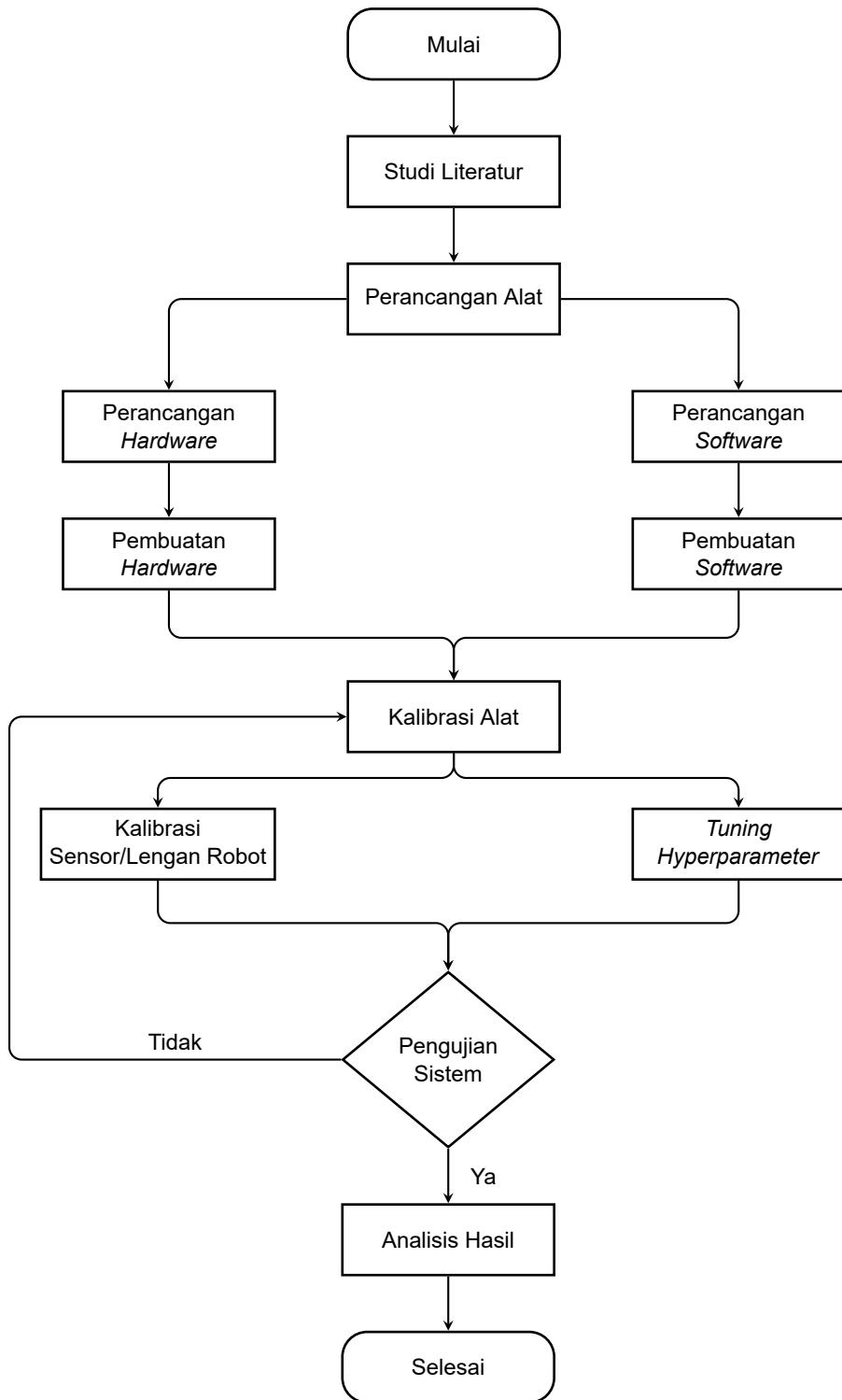
2.2 Peralatan Penelitian

Peralatan yang digunakan pada penelitian ini adalah sebagai berikut:

1. Arduino Uno. berfungsi sebagai mikrokontroler utama yang mengendalikan motor servo pada lengan robot dan menerima sinyal dari sensor.
2. Motor *servo*, digunakan sebagai aktuator untuk menggerakkan bagian-bagian lengan robot sesuai perintah dari Arduino.
3. Lengan robot EEZYbotARM MK1, merupakan struktur mekanik tempat pemasangan motor *servo* dan berperan sebagai sistem penggerak robotik.
4. *Power supply* 5V, berfungsi menyediakan tegangan stabil untuk pengoperasian motor *servo*.
5. *Sensor Passive Infrared Receiver* (PIR) HC-SR501, digunakan untuk mendeteksi pergerakan dan membantu menghitung jumlah kontainer cacat dan maupun tidak cacat yang melintas.
6. Kamera, digunakan untuk mengambil citra kontainer kimia, yang akan diproses oleh model deteksi objek (YOLO) dan deteksi kecacatan (*autoencoder*).
7. Laptop/komputer, digunakan untuk memuat program ke Arduino Uno, serta menjalankan model deteksi objek berbasis YOLO dan *autoencoder* untuk analisis visual.
8. Kabel *jumper*, berfungsi menghubungkan berbagai komponen elektronik seperti sensor dan aktuator ke papan rangkaian dan Arduino.
9. Papan rangkaian, berfungsi untuk menyediakan jalur koneksi antar komponen.

2.3 Metode Kerja

Penelitian ini terdiri dari beberapa tahapan, sebagaimana ditunjukkan pada Gambar 1. Fokus penelitian ini adalah pada perancangan dan pembuatan prototipe sistem deteksi cacat secara otomatis.

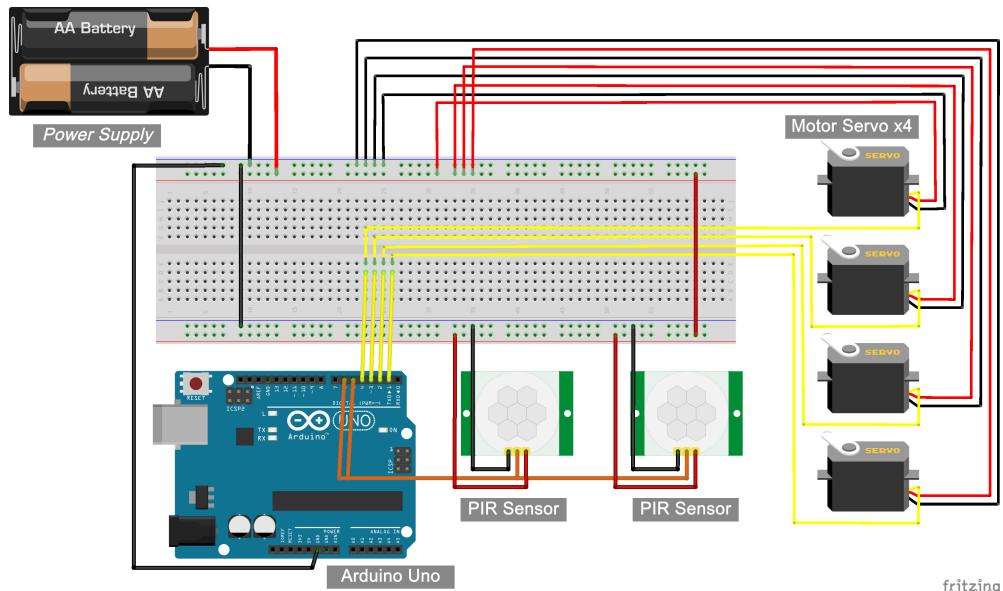


Gambar 1. Bagan alir penelitian

Tahapan awal dimulai dengan kajian literatur mendalam mengenai teknologi robotik, algoritma *autoencoder* untuk deteksi anomali visual, serta metode deteksi objek seperti YOLO. Kajian ini bertujuan untuk memperoleh pemahaman komprehensif terkait format *dataset* dan perancangan model deteksi cacat pada kontainer kimia. Setelah pemahaman awal diperoleh, dilakukan perancangan sistem yang mencakup komponen perangkat keras (lengan robot dan sensor) dan perangkat lunak berupa algoritma *machine learning*. Selanjutnya sistem robot dikalibrasi agar dapat bekerja secara optimal, termasuk proses *tuning hyperparameter* pada model *machine learning* agar berfungsi secara optimal. Setelah sistem dapat beroperasi dengan baik, dilakukan proses pengambilan data sebagai langkah awal dalam pengujian dan validasi model.

2.3.1 Perancangan Hardware

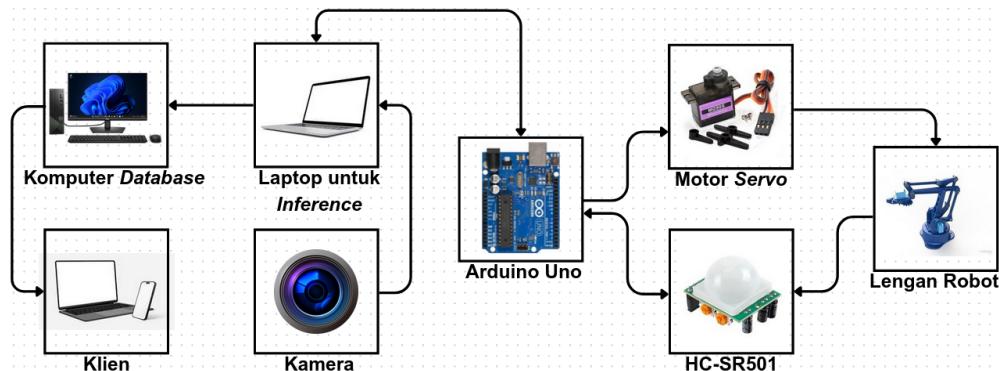
Penelitian ini dimulai dengan tahap perancangan *hardware*. Komponen *hardware* meliputi kamera untuk mengambil citra kontainer kimia, laptop sebagai pusat pemrosesan algoritma, dan sensor PIR untuk menghitung jumlah kontainer cacat dan tidak cacat. Rangkaian *hardware* dapat dilihat pada Gambar 2.



Gambar 2. Rangkaian *hardware* sistem

Alur kerja sistem dimulai saat kamera menangkap citra kontainer yang berada pada area pengambilan gambar. Model YOLO digunakan untuk mendeteksi objek kontainer. Setelah objek dikenali, citra dikirim ke model deteksi kecacatan berbasis *convolutional variational autoencoder* (CVAE) untuk menentukan keberadaan cacat. Berdasarkan hasil analisis tersebut, Arduino mengirimkan sinyal ke motor servo untuk menggerakkan lengan robot dalam memindahkan kontainer sesuai hasil klasifikasi. Dua sensor PIR yang dipasang pada masing-masing wadah (cacat dan tidak

cacat) digunakan untuk mencatat jumlah kontainer. Data dari sensor dikirim *server* dan ditampilkan secara *real-time*. Diagram keterkaitan antar komponen disajikan pada Gambar 3.

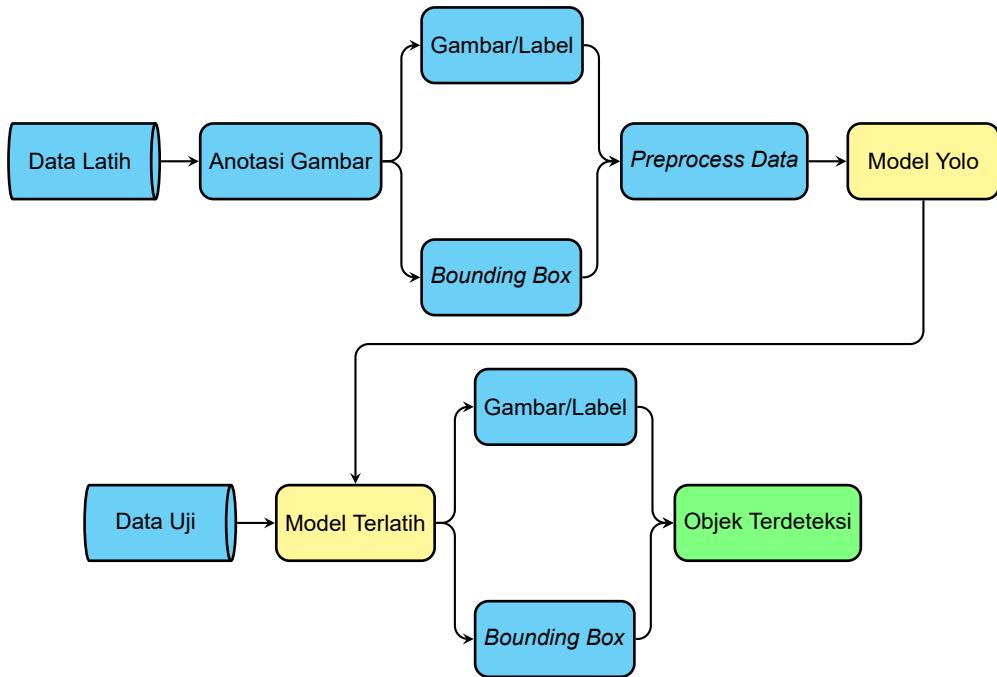


Gambar 3. Rancang sistem *hardware*

2.3.2 Perancangan Software

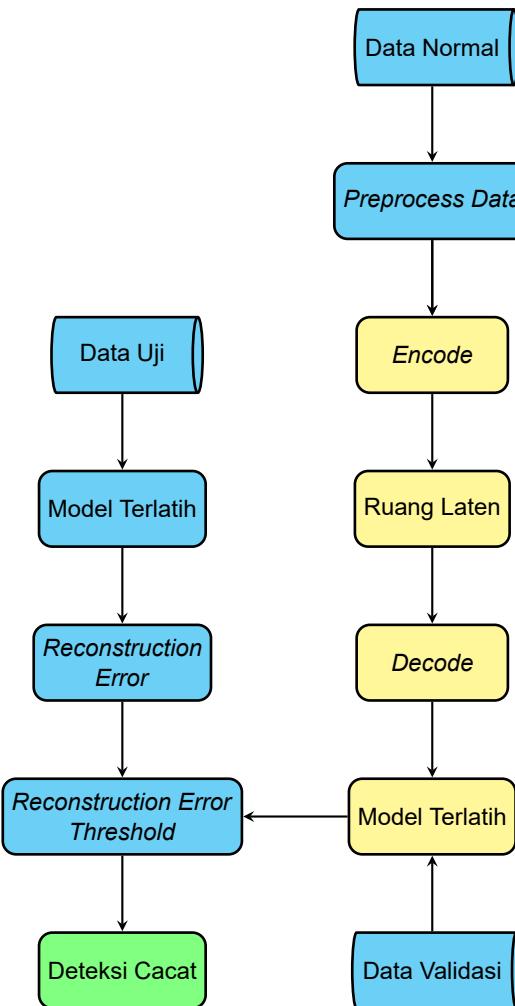
Perangkat lunak dalam penelitian ini mencakup: model deteksi objek berbasis YOLO untuk mengidentifikasi kontainer kimia, kedua adalah model deteksi cacat menggunakan CVAE, ketiga adalah algoritma kontrol pergerakan lengan robot, dan yang keempat adalah integrasi modul *Internet of Things* (IoT) untuk menampilkan data secara *real-time* melalui *website*.

Proses diawali dengan pengumpulan citra kontainer kimia dari berbagai kondisi dan sudut pandang menggunakan kamera. *Dataset* yang diperoleh kemudian dianotasi secara manual dengan label dan *bounding box* format YOLO. Model dilatih untuk mendeteksi kontainer secara akurat dan cepat. Evaluasi dilakukan menggunakan metrik *precision*, *recall*, dan *mean Average Precision* (mAP), guna memastikan model memiliki kemampuan generalisasi yang baik. Alur proses dapat dilihat pada Gambar 4.



Gambar 4. Diagram alur pelatihan model YOLO

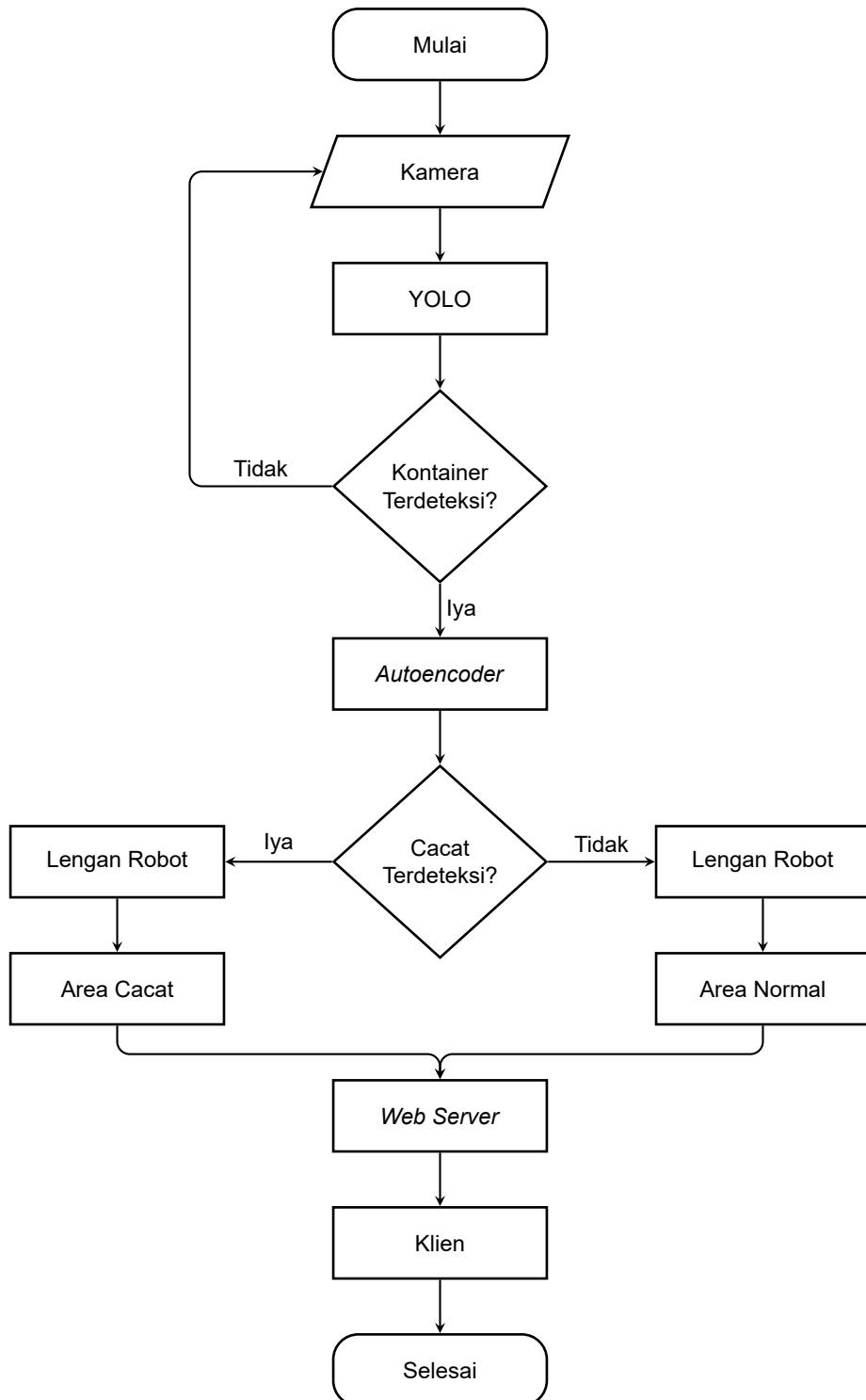
Selanjutnya, model deteksi cacat menggunakan algoritma CVAE dikembangkan dengan *dataset* serupa namun tanpa *bounding box*, karena model bersifat *unsupervised*. Data melalui tahap *preprocessing* (ubah ukuran, normalisasi, dan augmentasi) untuk meningkatkan keragaman data. Model terdiri dari *encoder* (untuk mengestraksi fitur dan menghasilkan representasi laten) dan *decoder* (untuk merekonstruksi citra). Evaluasi dilakukan dengan mengukur *reconstruction error*, guna membedakan citra normal dan cacat. Ambang batas ditentukan dari distribusi *error* pada data validasi. Proses ini ditunjukkan pada Gambar 5.



Gambar 5. Diagram alur pelatihan model deteksi cacat

2.3.3 Bagan Alir Sistem Kerja Alat

Alur kerja sistem dimulai dari pengambilan citra oleh kamera, pendekripsi kontainer menggunakan YOLO dan klasifikasi kecacatan oleh *autoencoder*. Berdasarkan hasil tersebut, lengan robot memindahkan kontainer ke wadah yang sesuai. Data hasil klasifikasi dikirim ke *web server* untuk ditampilkan secara *real-time*. Alur kerja sistem secara keseluruhan ditunjukkan pada Gambar 6.



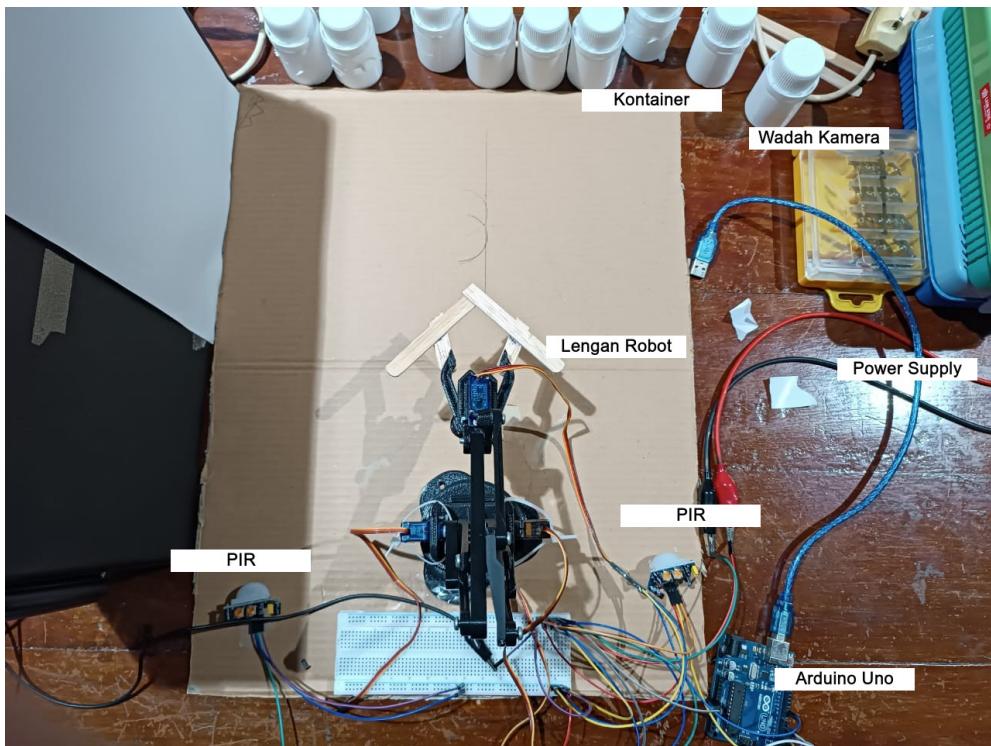
Gambar 6. Bagan alir sistem kerja alat

BAB III

HASIL DAN PEMBAHASAN

3.1 Hasil Perancangan Sistem

Sistem deteksi kecacatan kontainer yang dikembangkan pada penelitian ini dirancang untuk bekerja secara otomatis dengan menggunakan kamera sebagai sensor utama dan lengan robot sebagai aktuator. Proses diawali oleh model YOLO yang mendeteksi keberadaan kontainer dari citra hasil tangkapan kamera. Jika objek kontainer berhasil terdeteksi, citra tersebut dianalisis oleh model *autoencoder* untuk mengklasifikasikan apakah kontainer mengalami kecacatan. Berdasarkan hasil klasifikasi, mikrokontroler mengendalikan motor *servo* untuk menggerakkan lengan robot guna menyortir kontainer ke dalam kategori cacat atau tidak cacat. Secara paralel, sensor PIR digunakan untuk menghitung jumlah total kontainer pada masing-masing kategori. Data ini kemudian dikirim (melalui metode POST) ke *web server* untuk divisualisasikan pada sisi klien melalui tampilan *website*. Desain sistem yang telah dirancang diilustrasikan pada Gambar 7.



Gambar 7. Hasil perancangan sistem

3.2 Hasil Perancangan Model YOLO

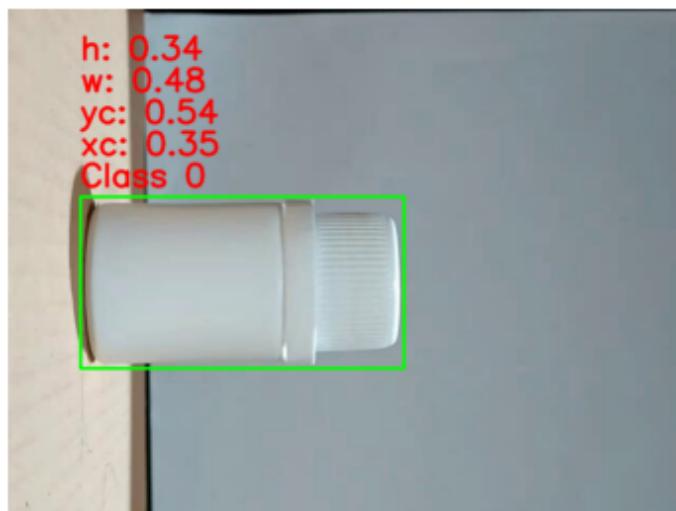
3.2.1 Dataset dan Preprocessing

Tahap awal penelitian adalah pengumpulan data primer berupa citra kontainer kimia menggunakan kamera. Proses akuisisi ini dilakukan untuk membangun sebuah *dataset* kustom yang mampu merepresentasikan objek target secara akurat. Total citra mentah yang berhasil dikumpulkan berjumlah 463 citra yang diambil dari berbagai posisi dan sudut pandang untuk memastikan model dapat mengenali objek dalam berbagai kondisi. *Dataset* kemudian dibagi menjadi 395 citra untuk data latih dan 68 citra data validasi. Distribusi pembagian dataset disajikan pada Tabel 1.

Tabel 1. Distribusi pembagian *dataset*

| Kategori | Jumlah Gambar | Persentase |
|---------------|---------------|------------|
| Data Latih | 395 | 85,3% |
| Data Validasi | 68 | 14,7% |
| Total Data | 463 | 100% |

Setelah akuisisi data, dilakukan anotasi citra untuk memberikan *bounding box* dan label kelas pada setiap objek yang terdeteksi di dalam citra. Proses ini penting karena arsitektur YOLO dirancang untuk secara simultan memprediksi letak objek dan kelasnya, sehingga memerlukan data berlabel (Ragab et al., 2024; Hussain, 2024). Sebanyak 463 citra kontainer telah dianotasi secara manual. Contoh hasil anotasi dapat dilihat pada Gambar 8.



Gambar 8. Salah satu data *train* dengan labelnya

3.2.2 Hasil Pelatihan Model

Dalam penelitian ini, metrik utama yang digunakan untuk mengevaluasi performa model YOLO adalah mAP. Metrik ini penting untuk menilai seberapa akurat model dalam mendeteksi objek sehingga lengan robot dapat melakukan penyortiran secara tepat. Nilai mAP dihitung berdasarkan rata-rata *Average Precision* (AP), yang merupakan gabungan antara *precision* dan *recall* untuk setiap kelas (Padilla et al., 2021).

Untuk menentukan apakah sebuah prediksi tergolong benar (*True Positive*) atau salah (*False Positive*), digunakan metrik *Intersection over Union* (IoU). IoU yaitu rasio antara *bounding box* prediksi ($BB_{predict}$) dengan *bounding box* sebenarnya (BB_{ground}), dengan nilai mendekati 1 menunjukkan prediksi yang sangat akurat (Kaur dan Singh, 2023). Semakin mendekati 1, berarti prediksi semakin akurat dan sesuai dengan objek sebenarnya. IoU dihitung menggunakan persamaan:

$$\text{IoU} = \frac{|BB_{predict} \cap BB_{ground}|}{|BB_{predict} \cup BB_{ground}|} \quad (1)$$

Selanjutnya, *precision* mengukur proporsi prediksi positif yang benar (*True Positive*) terhadap seluruh prediksi positif ($TP + False Positive$), sedangkan *recall* atau *true positive rate* mengukur seberapa banyak objek yang benar-benar terdeteksi (Casas et al., 2023). *Precision* dan *recall* dapat dihitung menggunakan persamaan:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

Karena model YOLO dalam penelitian ini hanya memprediksi satu kelas yaitu kontainer kimia, maka nilai mAP setara dengan nilai AP. Nilai AP dihitung sebagai rata-rata *precision* di seluruh rentang nilai *recall* (0 hingga 1) (Cao et al., 2024), sebagaimana dirumuskan dalam Persamaan 3, di mana P adalah *precision* dan r adalah *recall*.

$$AP = \int_0^1 P(r) dr \quad (3)$$

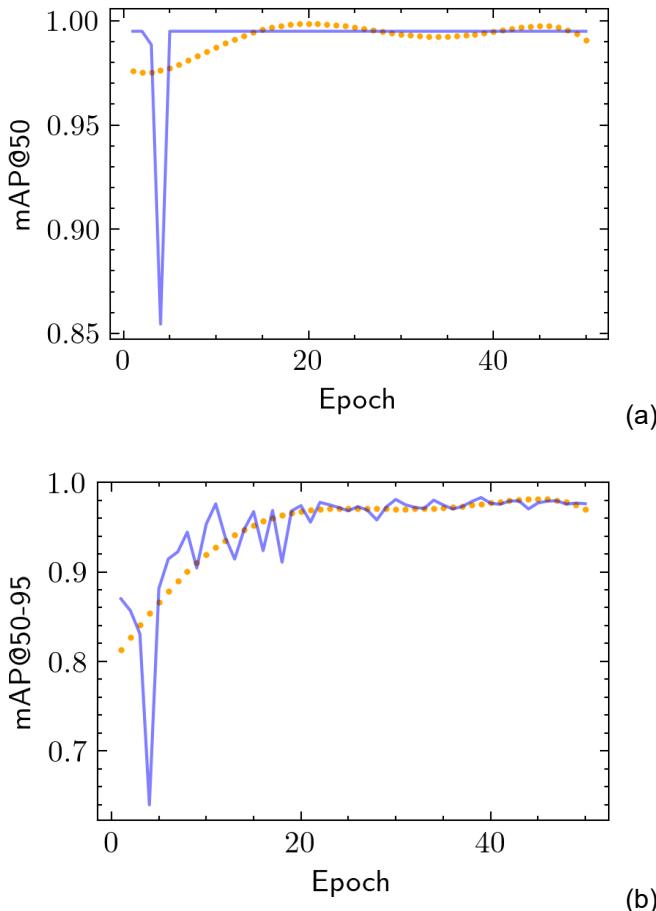
Proses pelatihan model YOLO dilakukan di platform *Google Colaboratory* selama 50 *epoch* menggunakan *batch size* 16 dan *optimizer* Adam. Ringkasan hasil pelatihan setiap 10 *epoch* disajikan pada Tabel 2.

Tabel 2. Proses *training* model YOLO

| Epoch | Train/Box Loss | Train/Class Loss | Train/DFL Loss | Val/Box Loss | Val/Class Loss | Val/DFL Loss |
|-------|----------------|------------------|----------------|--------------|----------------|--------------|
| 0 | 0,5946 | 1,9374 | 0,9706 | 0,4709 | 2,5029 | 0,8990 |
| 10 | 0,4333 | 0,4356 | 0,8713 | 0,3052 | 0,3758 | 0,8262 |
| 20 | 0,3286 | 0,2935 | 0,8510 | 0,3342 | 0,2105 | 0,8448 |
| 30 | 0,2821 | 0,2328 | 0,8390 | 0,2418 | 0,1566 | 0,8244 |
| 40 | 0,2267 | 0,1776 | 0,8010 | 0,2209 | 0,1386 | 0,8201 |
| 50 | 0,2016 | 0,1546 | 0,7993 | 0,2003 | 0,1163 | 0,8156 |

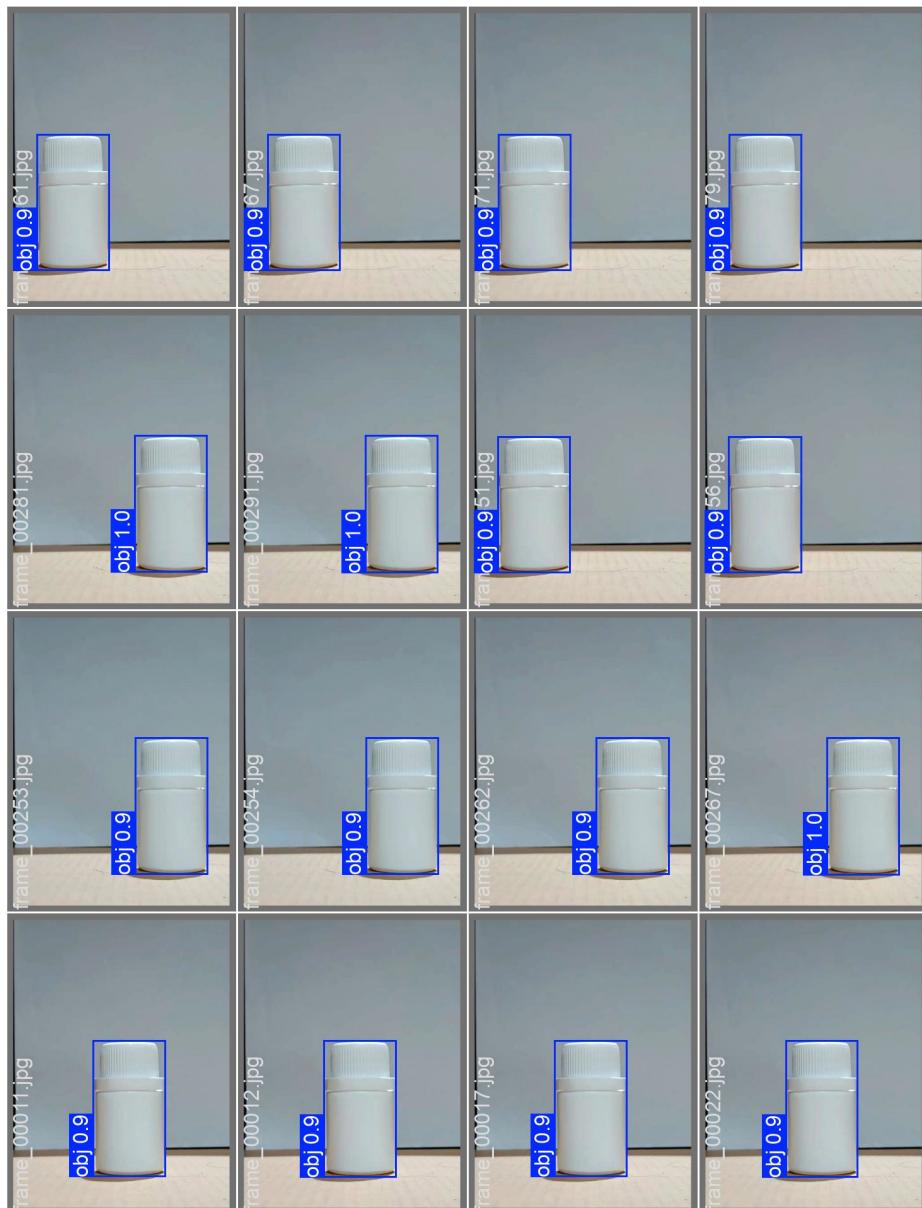
Nilai *loss* pada data latih dan validasi secara umum menunjukkan tren menurun, mengindikasikan bahwa model berhasil menyesuaikan parameter internalnya dengan data. Pada *Train/Box Loss*, terjadi penurunan signifikan dari 0,5946 pada *epoch* ke-0 menjadi 0,2016 pada *epoch* ke-50, yang menunjukkan peningkatan akurasi deteksi posisi objek. Penurunan signifikan juga terlihat pada *Train/Class Loss* dan *Train/DFL Loss*, serta pada seluruh komponen *loss* di data validasi. Penurunan konsisten ini menunjukkan bahwa model memiliki kemampuan generalisasi yang baik pada data validasi.

Meskipun demikian, nilai *loss* saja tidak cukup untuk menilai keseluruhan performa model. Oleh karena itu, digunakan pula metrik mAP, seperti mAP@50 dan mAP@50-95, untuk mengevaluasi ketepatan deteksi: mAP@50 berarti evaluasi dilakukan dengan ambang IoU 50% yang cenderung lebih longgar, sedangkan mAP@50-95 adalah rata-rata mAP pada ambang IoU dari 50% hingga 95% dengan interval 5%, sehingga lebih ketat. Hasil pengukuran mAP pada tiap *epoch* dapat dilihat pada Gambar 9.



Gambar 9. Grafik tren mAP: (a) mAP@50, (b) mAP@50-95

Grafik (a) menunjukkan bahwa model dengan cepat mencapai nilai mAP tinggi pada ambang 50%, bahkan dalam 10-15 epoch pertama. Grafik (b) menunjukkan tren yang lebih lambat dan bertahap, mengindikasikan proses penyempurnaan presisi dan akurasi posisi *bounding box*. Secara keseluruhan, nilai mAP yang tinggi pada kedua metrik diakhir pelatihan menunjukkan bahwa model yang dihasilkan tidak hanya mampu mendeteksi objek, tetapi juga mampu menentukan batas objek secara akurat. Contoh hasil prediksi model YOLO terhadap data validasi disajikan pada Gambar 10.

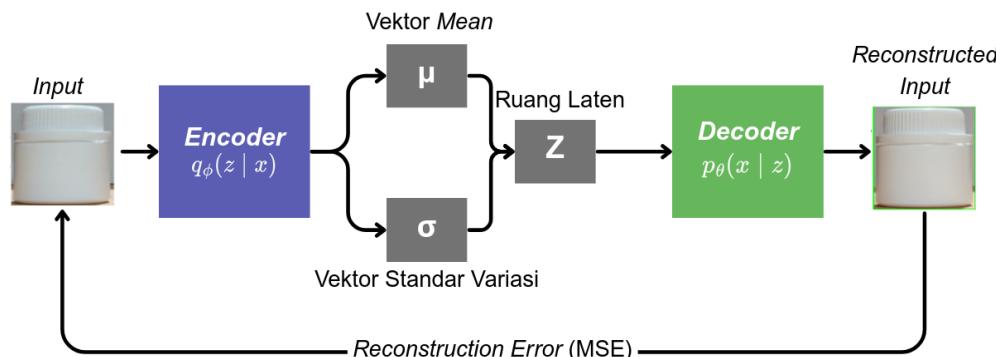


Gambar 10. Hasil prediksi YOLO pada data validasi

3.3 Hasil Perancangan Model Deteksi Kecacatan

3.3.1 Arsitektur Variational Autoencoder

Variational autoencoder (VAE) merupakan pengembangan dari *autoencoder*, yaitu arsitektur *neural network* yang berfungsi untuk mengekstraksi fitur laten dari data. Perbedaan utama VAE dibandingkan *autoencoder* konvensional adalah pendekatannya yang berbasis Bayesian, di mana vektor laten (z) dipaksa mengikuti distribusi probabilitas terstruktur, biasanya distribusi Gaussian. Secara operasional, bagian *encoder* VAE ($q_\phi(z | x)$) tidak hanya mengompresi data masukan (x), tetapi juga memetakannya ke dalam parameter statistik: yaitu vektor rata-rata (*mean*) dan vektor varians (*log-variens*). Kemudian, sebuah vektor laten (z) diambil dari distribusi dan diteruskan ke *decoder* ($p_\theta(x | z)$) untuk merekonstruksi kembali data masukan (Mansouri et al., 2021). Arsitektur *variational autoencoder* yang digunakan pada penelitian ini dapat dilihat pada Gambar 11.



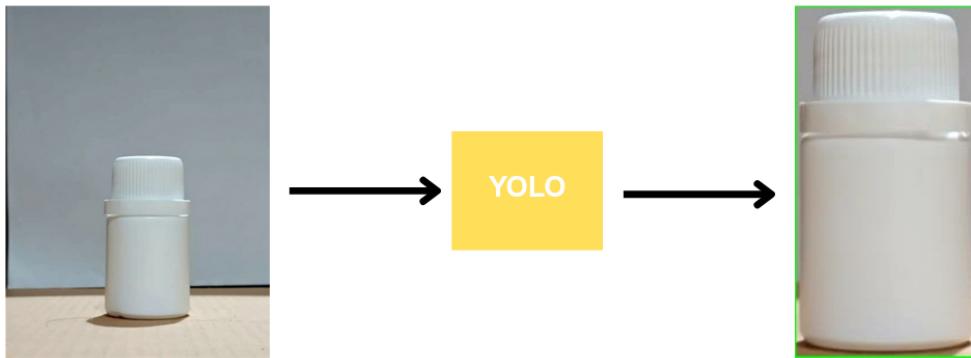
Gambar 11. Arsitektur *variational autoencoder*

Arsitektur VAE dirancang untuk memproses citra berukuran 128×128 dengan tiga kanal warna (RGB). Bagian *encoder* terdiri atas lima lapisan *convolutional* berurutan, masing-masing diikuti fungsi aktivasi ReLU yang bertugas menurunkan resolusi fitur spasial dan mengekstraksi fitur penting hingga menghasilkan representasi akhir berdimensi (512, 4, 4). Representasi ini kemudian diproyeksikan menjadi dua vektor statistik, yaitu vektor *mean* dan vektor *log-variens*, yang bersama-sama membentuk distribusi Gaussian multivariat sebagai ruang laten. Setelah proses pengambilan sampel dari ruang laten, vektor z diteruskan ke *decoder* yang memproyeksikan kembali representasi tersebut melalui lapisan linear dan beberapa lapisan *transpose convolutional* (dekonvolusi) untuk mengembalikan ukuran citra ke dimensi semula. Fungsi aktivasi sigmoid pada lapisan *output* memastikan nilai piksel berada pada rentang [0, 1], sehingga citra hasil rekonstruksi dapat diinterpretasikan sebagai citra valid.

3.3.2 Dataset dan Preprocessing

Tahap ini diawali dengan pengumpulan data primer berupa citra kontainer kimia menggunakan kamera. Tujuannya adalah membangun *dataset* kustom yang dapat merepresentasikan objek target secara spesifik. Total citra mentah yang berhasil di-

kumpulkan berjumlah 3005 citra, yang seluruhnya digunakan sebagai data latih. Sebelum digunakan untuk pelatihan *autoencoder*, seluruh citra diproses terlebih dahulu menggunakan model YOLO untuk mendeteksi serta memotong (*cropping*) area kontainer secara otomatis. Hal ini bertujuan agar hanya bagian penting dari kontainer yang diproses lebih lanjut, sehingga model *autoencoder* dapat secara optimal mempelajari karakteristik visual kontainer. Contoh salah satu citra data latih ditunjukkan pada Gambar 12.



Gambar 12. Salah satu data latih model *autoencoder*

Karena *autoencoder* termasuk dalam kategori *unsupervised learning*, maka data tidak memerlukan label atau anotasi manual. *Dataset* ini hanya terdiri dari citra kontainer dalam kondisi baik (tanpa kerusakan/cacat visual), sehingga model *autoencoder* dapat mempelajari distribusi visual normal secara optimal. Dengan demikian, saat digunakan pada data yang mengandung cacat model akan menunjukkan perbedaan yang mencolok antara citra asli dan hasil rekonstruksi yang menjadi dasar untuk mendeteksi anomali.

3.3.3 Hasil Pelatihan Model

Fungsi *loss* pada VAE didasarkan pada prinsip *Evidence Lower Bound* (ELBO) yang bertujuan mengoptimalkan model probabilistik dengan mengatasi kendala distribusi posterior yang tidak dapat dihitung secara langsung. Fungsi *loss* VAE terdiri dari dua komponen utama: (1) *reconstruction loss*, yang mengukur selisih antara citra asli (x) dan citra hasil rekonstruksi dari vektor laten z , dan (2) *regularization term* berupa divergensi Kullback-Leibler (KL) yang mengukur sejauh mana distribusi posterior hasil *encoder* menyimpang dari distribusi Gaussian standar (Wei et al., 2020). Fungsi *loss* VAE dirumuskan sebagai berikut:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) \quad (4)$$

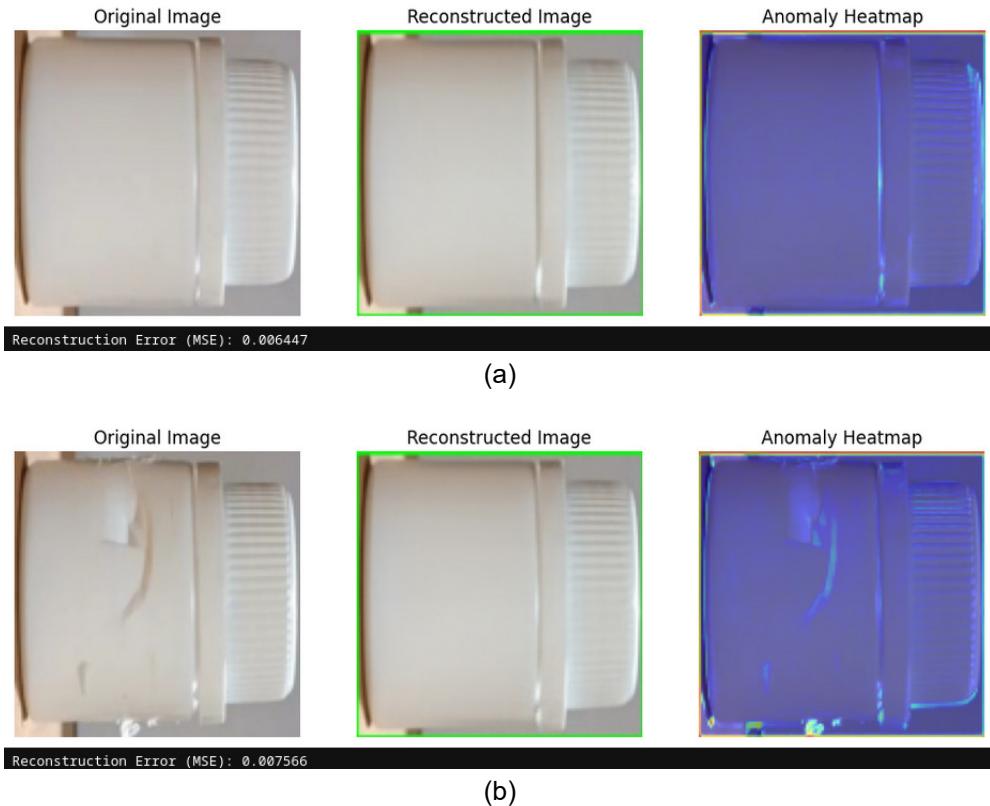
Model CVAE dilatih menggunakan komputer dengan GPU NVIDIA GeForce RTX 3050 8GB VRAM untuk memanfaatkan kemampuan komputasi paralel CUDA.

Proses pelatihan dilakukan selama 100 epoch dengan *batch size* 64 dan *optimizer* Adam. Nilai *loss* setiap 10 epoch ditampilkan pada Tabel 3.

Tabel 3. Proses *training* model model *convolutional variational autoencoder*

| Epoch | Loss |
|-------|----------|
| 1 | 837,7615 |
| 10 | 30,5440 |
| 20 | 22,6941 |
| 30 | 18,5220 |
| 40 | 15,0172 |
| 50 | 11,9098 |
| 60 | 9,8593 |
| 70 | 8,7772 |
| 80 | 7,3923 |
| 90 | 7,1070 |
| 100 | 6,8338 |

Penurunan nilai *loss* selama proses pelatihan menunjukkan bahwa model CVAE secara bertahap berhasil mempelajari pola visual dari kontainer normal. Dengan semakin kecilnya nilai *loss*, model menunjukkan kemampuan yang semakin baik dalam merekonstruksi citra tanpa cacat, sehingga mampu membedakan citra cacat secara signifikan. Hasil rekonstruksi ditampilkan pada Gambar 13.



Gambar 13. Hasil prediksi kecacatan: (a) Kontainer normal, (b) kontainer cacat

Dari gambar di atas, diketahui bahwa nilai rekonstruksi *error* untuk kontainer normal adalah 0,006447, sedangkan untuk kontainer cacat sebesar 0,007566. Selisih ini menunjukkan bahwa model mampu membedakan citra cacat dari normal normal berdasarkan tingkat kesalahan rekonstruksi.

3.3.4 Penentuan Ambang Batas Kecacatan

Penentuan ambang batas merupakan langkah penting dalam memastikan sistem deteksi cacat dapat secara akurat membedakan kontainer cacat dan tidak cacat. Tanpa ambang batas yang tepat, sistem beresiko menghasilkan banyak kesalahan klasifikasi yang tinggi. Dalam penelitian ini, digunakan metrik *Mean Squared Error* (MSE) untuk mengukur perbedaan antara citra asli dan hasil rekonstruksi.

MSE menghitung rata-rata selisih kuadrat antara piksel citra asli (*input*) dengan piksel citra hasil rekonstruksi (*output*). Semakin kecil nilai MSE, semakin mirip citra hasil rekonstruksi dengan citra aslinya, yang berarti model berhasil meminimalkan distorsi (Najjar, 2024). MSE dirumuskan sebagai berikut:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2 \quad (5)$$

Pengujian dilakukan terhadap 20 citra, terdiri dari 10 kontainer normal dan 10 gambar kontainer cacat. Dari pengujian ini diperoleh distribusi nilai MSE masing-masing citra untuk dianalisis lebih lanjut. Hasil perhitungan MSE pada 20 gambar uji tersebut dirangkum pada Tabel 4.

Tabel 4. Nilai error untuk setiap sampel

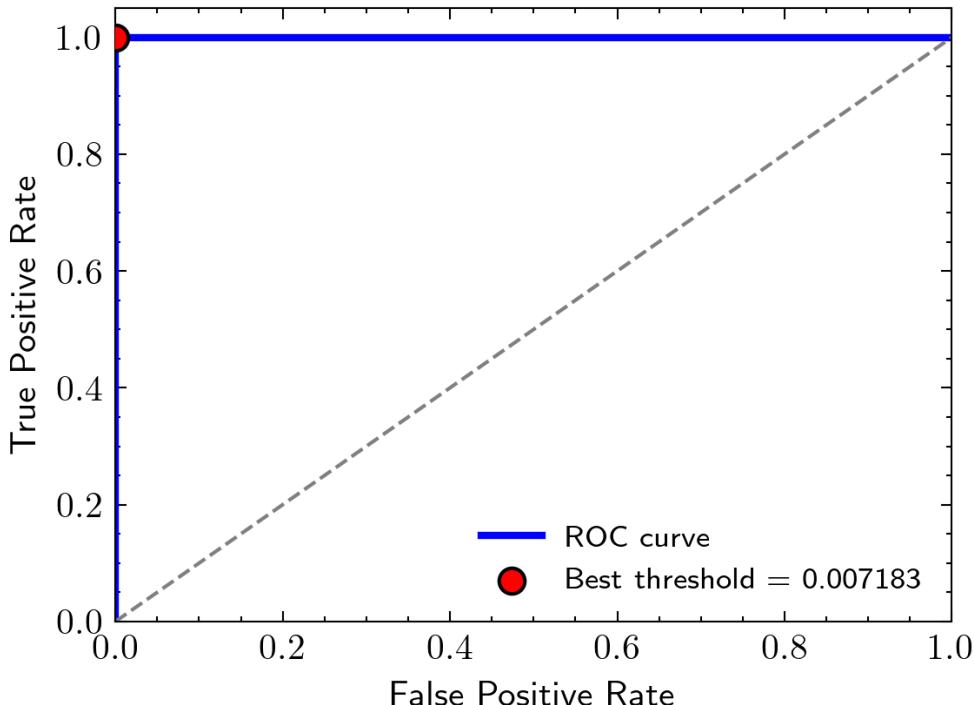
| No | Error | Kategori |
|----|----------|----------|
| 1 | 0,006737 | Normal |
| 2 | 0,006828 | Normal |
| 3 | 0,007091 | Normal |
| 4 | 0,006997 | Normal |
| 5 | 0,006610 | Normal |
| 6 | 0,006795 | Normal |
| 7 | 0,006872 | Normal |
| 8 | 0,006725 | Normal |
| 9 | 0,006873 | Normal |
| 10 | 0,006817 | Normal |
| 11 | 0,007432 | Cacat |
| 12 | 0,007455 | Cacat |
| 13 | 0,008369 | Cacat |
| 14 | 0,007183 | Cacat |
| 15 | 0,007653 | Cacat |
| 16 | 0,008856 | Cacat |
| 17 | 0,007531 | Cacat |
| 18 | 0,007624 | Cacat |
| 19 | 0,007457 | Cacat |
| 20 | 0,008424 | Cacat |

Dari tabel di atas, nilai MSE untuk kontainer normal berkisar 0,0067 sedangkan nilai untuk kontainer cacat secara konsisten lebih tinggi. Hal ini menunjukkan bahwa perbedaan rekonstruksi pada citra cacat lebih besar, menandakan kegagalan model untuk merekonstruksi fitur yang tidak dikenalnya. Untuk menentukan ambang batas optimal, digunakan metode *Receiver Operating Characteristic* (ROC) yang memplot *true positive rate* (*recall*) terhadap *false positive rate* ($1 - specificity$) (Nahm, 2022). Nilai ambang terbaik ditentukan dengan memaksimalkan statistik Youden J yang dirumuskan sebagai:

$$J = True\ Positive\ Rate - False\ Positive\ Rate \quad (6)$$

Kurva ROC dari hasil pengujian ditampilkan pada Gambar 14 digunakan sebagai metode analisis untuk mengevaluasi performa model deteksi cacat pada kontainer kimia dalam penelitian ini. Kurva ROC dibuat dengan memplot nilai $1 - specificity$ (*false positive rate*) pada sumbu-x dan *recall* (*true positive rate*) pada sumbu-y untuk setiap nilai ambang batas yang diuji. *Specificity* sendiri adalah ukuran seberapa baik

model dalam mengenali data negatif secara benar. Nilai ambang batas terbaik ditentukan dengan memaksimalkan statistik Youden J, yang dirumuskan pada Persamaan 6.

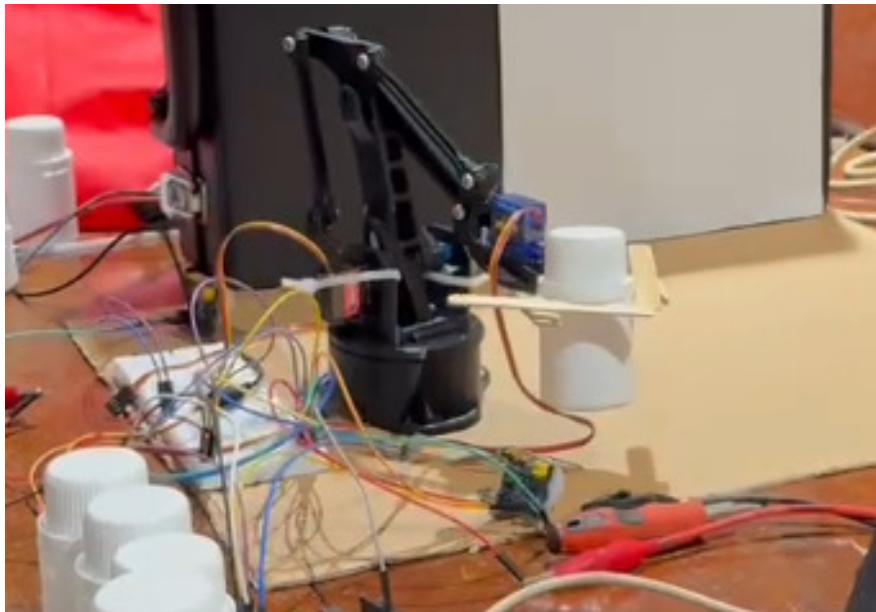


Gambar 14. Kurva ROC

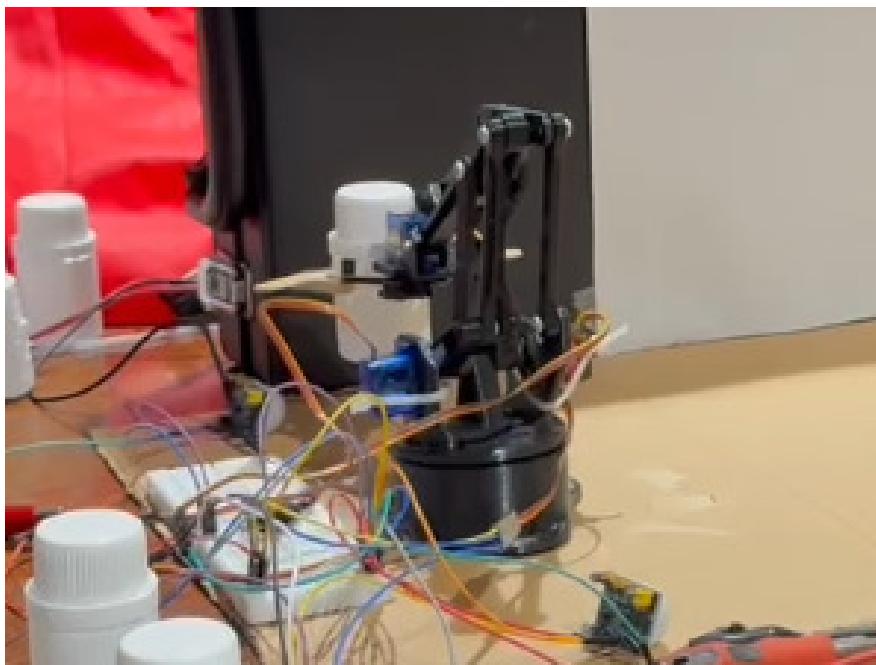
Kurva ROC menunjukkan bahwa model memiliki performa yang sangat baik, dengan garis mendekati sudut kiri atas. Berdasarkan kurva, diperoleh nilai ambang optimal ditunjukkan oleh garis biru yang mendekati titik sudut kiri atas. Berdasarkan perhitungan, ambang batas optimal sebesar 0,007183, ditandai dengan lingkaran merah pada grafik. Nilai ini digunakan sebagai batas akhir dalam klasifikasi kontainer cacat dan tidak cacat, guna memaksimalkan sensitivitas sekaligus meminimalkan kesalahan deteksi positif.

3.4 Hasil Pengujian Sistem Lengan Robot

Setelah model deteksi terlatih dan nilai ambang batas optimal ditentukan, tahap selanjutnya adalah integrasi sistem dengan lengan robot yang dikendalikan oleh mikrokontroler Arduino. Berdasarkan hasil klasifikasi dari model, lengan robot akan secara otomatis memindahkan kontainer kimia. Jika objek teridentifikasi sebagai cacat, maka lengan robot akan mengarahkan ke sisi kiri, sedangkan jika dinyatakan normal akan diarahkan ke sisi kanan . Gambar 15 mengilustrasikan aksi lengan robot ketika sedang menyortir objek yang teridentifikasi normal dan cacat.



(a)

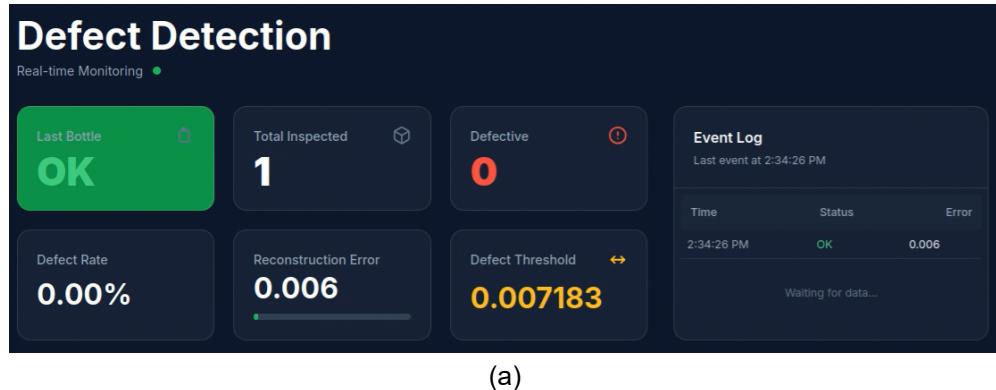


(b)

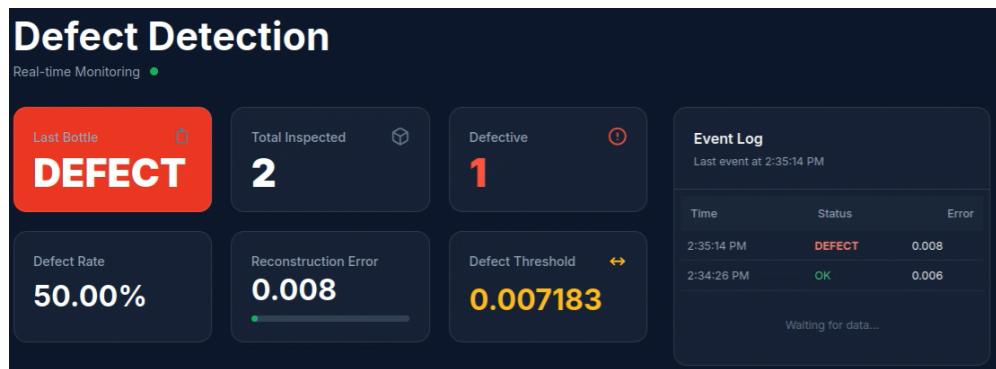
Gambar 15. Lengan robot menyortir kontainer: (a) normal, (b) cacat

Setelah lengan robot melepaskan kontainer di area penyortiran yang telah ditentukan, sebuah sensor PIR yang terpasang di lokasi tersebut akan mendeteksi

keberadaan objek. Deteksi dari sensor PIR ini berfungsi sebagai sinyal konfirmasi bahwa satu siklus penyortiran telah selesai. Sinyal ini kemudian memicu pengiriman data hasil klasifikasi, yang mencakup citra kontainer beserta label statusnya, ke *web server*. Untuk menampilkan informasi ini secara *real-time* kepada klien, sistem manfaatkan protokol *WebSocket*. Tampilan *website* dan hasil prediksi dapat dilihat pada Gambar 16.



(a)



(b)

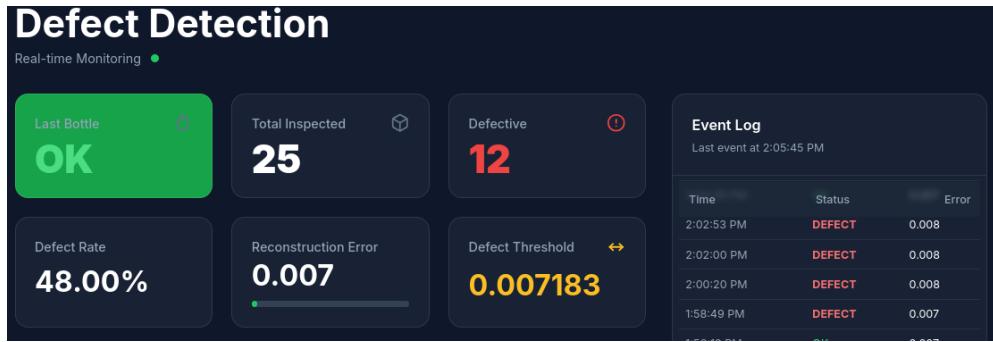
Gambar 16. Tampilan hasil prediksi melalui *website*: (a) kontainer normal, (b) kontainer cacat

Untuk menguji keandalan dan konsistensi performa sistem, dilakukan pengujian sebanyak 25 kali menggunakan sampel kontainer dengan kondisi acak (normal dan cacat). Hasil kuantitatif dari setiap pengujian ini dirangkum secara pada Tabel 5.

Tabel 5. Nilai *error* untuk setiap sampel uji

| No | Error | Kategori (Ambang Batas < 0,007183) | Label Asli |
|------------------------|----------|------------------------------------|-------------|
| 1 | 0,006798 | Normal | Normal |
| 2 | 0,008298 | Cacat | Cacat |
| 3 | 0,006854 | Normal | Normal |
| 4 | 0,007066 | Normal | Normal |
| 5 | 0,006923 | Normal | Normal |
| 6 | 0,007607 | Cacat | Cacat |
| 7 | 0,007261 | Cacat | Cacat |
| 8 | 0,007393 | Cacat | Cacat |
| 9 | 0,006842 | Normal | Normal |
| 10 | 0,008099 | Cacat | Cacat |
| 11 | 0,006945 | Normal | Normal |
| 12 | 0,007381 | Cacat | Cacat |
| 13 | 0,006722 | Normal | Normal |
| 14 | 0,006865 | Normal | Normal |
| 15 | 0,007397 | Cacat | Cacat |
| 16 | 0,007099 | Normal | Normal |
| 17 | 0,006790 | Normal | Normal |
| 18 | 0,006760 | Normal | Normal |
| 19 | 0,007383 | Cacat | Cacat |
| 20 | 0,007693 | Cacat | Cacat |
| 21 | 0,008071 | Cacat | Cacat |
| 22 | 0,007977 | Cacat | Cacat |
| 23 | 0,006885 | Normal | Normal |
| 24 | 0,008007 | Cacat | Cacat |
| 25 | 0,006847 | Normal | Normal |
| Tingkat Akurasi | | | 100% |

Hasil klasifikasi ditentukan berdasarkan perbandingan antara *error* rekonstruksi dengan ambang batas 0,007183. Sampel dengan *error* lebih rendah dikategorikan sebagai "Normal", sedangkan yang lebih tinggi dikategorikan sebagai "Cacat". Dari Tabel 5 terlihat bahwa seluruh prediksi model sesuai dengan kondisi sebenarnya (label asli), menghasilkan tingkat akurasi 100%. Contohnya, sampel no. 2 memiliki nilai *error* yang tinggi (0,008298) terklarifikasi dengan benar "Cacat". Sebaliknya sampel no. 1, dengan *error* yang rendah (0,006798) teklarifikasi dengan benar sebagai "Normal". Hasil prediksi akhir pada website ditunjukkan pada Gambar 17.



Gambar 17. Hasil prediksi kecacatan pada 25 sampel

BAB IV KESIMPULAN

4.1 Kesimpulan

Berdasarkan hasil perancangan, pengujian, dan implementasi sistem diperoleh kesimpulan sebagai berikut:

1. Model deteksi objek berbasis YOLO telah berhasil dirancang dan dilatih untuk mengenali kontainer kimia dengan akurasi dan presisi lokalisasi yang tinggi. Evaluasi menggunakan metrik mAP, menunjukkan performa yang sangat baik dalam mendeteksi objek dan menentukan posisi *bounding box* secara akurat.
2. Model deteksi cacatan berbasis CVAE terbukti efektif dalam mengidentifikasi anomali visual pada permukaan kontainer. Model dilatih hanya pada citra kontainer tanpa cacat dan berhasil merekonstruksi citra normal dengan tingkat error rendah. Sementara itu, kontainer cacat menghasilkan error rekonstruksi yang signifikan lebih tinggi. Ambang batas optimal sebesar 0,007183 ditentukan melalui analisis kurva ROC dan menghasilkan akurasi 100% pada 25 sampel uji.
3. Sistem yang dirancang berhasil mengintegrasikan seluruh proses secara otomatis: mulai dari akuisisi citra oleh kamera, deteksi objek oleh YOLO, dan klasifikasi cacat oleh CVAE, hingga penyortiran fisik kontainer menggunakan lengan robot. Hal ini membuktikan kelayakan sistem inspeksi visual cerdas untuk aplikasi industri teris

4.2 Saran

Sistem yang telah dibangun masih memiliki ruang untuk pengembangan lebih lanjut. Beberapa saran untuk penelitian selanjutnya antara lain:

1. Menerapkan pendekatan *supervised learning* seperti CNN atau varian YOLO untuk klasifikasi cacat, model dapat dikembangkan untuk mengenali berbagai tipe kerusakan secara spesifik sehingga meningkatkan akurasi deteksi dilingkungan industri nyata.
2. Meningkatkan fleksibilitas gerakan lengan robot dengan mengganti kontrol *hard-coded* menjadi pendekatan berbasis *inverse kinematics*. Dengan metode ini, sudut servo dapat dihitung secara dinamis berdasarkan koordinat target (x, y, z), memungkinkan sistem beradaptasi secara *real-time* terhadap posisi objek tanpa pemrograman ulang.
3. Mengembangkan sistem akuisisi citra 360 derajat. Prototipe saat ini hanya menggunakan 1 kamera dengan sudut pandang tetap, yang menyebabkan bagian permukaan kontainer tertentu tidak terjangkau. Penambahan kamera atau sistem rotasi objek dapat memungkinkan inspeksi menyeluruh terhadap seluruh permukaan kontainer, sehingga mendeteksi cacat tersembunyi secara lebih efektif

DAFTAR PUSTAKA

- Jhang, J. Y. & Lin, C. J. 2024. Jhang, Jyun-Yu, & Cheng-Jian Lin. Optimizing parameters of YOLO model through uniform experimental design for gripping tasks performed by an internet of things-based robotic arm. *Internet of Things* 27, 1-12. doi: 10.1016/j.iot.2024.101332.
- Oaki, J., Sugiyama, N., Ishihara, Y., Ooga, J., Kano, H. & Ohno, H., 2023. Micro-Defect Inspection on Curved Surface Using a 6-DOF Robot Arm with One-Shot BRDF Imaging. *IFAC-PapersOnLine* 56(2), 9354-9359. doi: 10.1016/j.ifacol.2023.10.224.
- Lin, C. J., Jhang, J. Y., Gao, Y. J. & Huang, H. M., 2024. Vision-based Robotic Arm in Defect Detection and Object Classification Applications. *Sensors & Materials* 36(2), 655-670. doi: 10.18494/SAM4683.
- Truong, A. M. & Luong, H. Q., 2024. A non-destructive, autoencoder-based approach to detecting defects and contamination in reusable food packaging. *Current Research in Food Science* 8, 1-12. doi: 10.1016/j.crf.2024.100758.
- Graetz, G. & Michaels, G., 2018. Robots at work. *Review of economics and statistics* 100(5), 753-768. doi: 10.1162/rest_a_00754.
- Gihleb, R., Giuntella, O., Stella, L. & Wang, T., 2022. Industrial robots, workers' safety, and health. *Labour economics* 78, 1-12. doi: 10.1016/j.labeco.2022.102205.
- Zhou, Y., & Zhao, Z., 2025. MPA-YOLO: Steel Surface Defect Detection Based on Improved YOLOv8 Framework. *Pattern Recognition* 168, 1-11. doi: 10.1016/j.patcog.2025.111897.
- Dong, L., Zhu, H., Ren, H., Lin, T. Y., & Lin, K. P., 2025. A novel lightweight MT-YOLO detection model for identifying defects in permanent magnet tiles of electric vehicle motors. *Expert Systems with Applications* 288, 1-13. doi: 10.1016/j.eswa.2025.128247.
- Liu, L., Du, D., Sun, Y., & Li, Y., 2025. SFMW-YOLO: A lightweight metal casting surface defect detection method based on modified YOLOv8s. *Expert Systems with Applications* 287. doi: 10.1016/j.eswa.2025.128170.
- Tsai, D. M. & Jen, P. H. 2021., Autoencoder-based anomaly detection for surface defect inspection. *Advanced Engineering Informatics* 48, 1-12. doi: 10.1016/j.aei.2021.101272.
- Chen, Y., Ding, Y., Zhao, F., Zhang, E., Wu, Z. & Shao, L., 2021. Surface defect detection methods for industrial products: A review. *Applied Sciences* 11(16), 1-25. doi: 10.3390/app11167657.

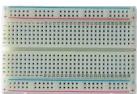
- Liu, Y., Qiu, W., Fu, K., Chen, X., Wu, L. & Sun, M., 2025. An improved YOLOv8 model and mask convolutional autoencoder for multi-scale defect detection of ceramic tiles. *Measurement* 248, 1-11. doi: 10.1016/j.measurement.2025.116847.
- Wang, L., 2024. Robot arm grasping based on YOLOv5 in the perspective of automated production. *Engineering Research Express*, 6(4), 1-12. doi:10.1088/2631-8695/ad88dc.
- Kato, H., Nagata, F., Murakami, Y. & Koya, K., 2022. Partial depth estimation with single image using YOLO and CNN for robot arm control. *IEEE International Conference on Mechatronics and Automation (ICMA)*, 1727-1731. IEEE. doi: 10.1109/ICMA54519.2022.9856055.
- Kim, M. & Kim, S. 2021., YOLO-based robotic grasping. *International Conference on Control, Automation and Systems (ICCAS)* 21, 1120-1122. doi: 10.23919/ICCAS52745.2021.9649837.
- Jin, T., Han, X., Wang, P., Zhang, Z., Guo, J. & Ding, F., 2025. Enhanced deep learning model for apple detection, localization, and counting in complex orchards for robotic arm-based harvesting. *Smart Agricultural Technology* 10, 1-25. doi: 10.1016/j.atech.2025.100784.
- Bionda, A., Frittoli, L. & Boracchi, G., 2022. Deep autoencoders for anomaly detection in textured images using CW-SSIM. *International Conference on Image Analysis and Processing*, 669-680. doi: 10.1007/978-3-031-06430-2_56.
- Yun, H., Kim, H., Jeong, Y. H. & Jun, M. B., 2023. Autoencoder-based anomaly detection of industrial robot arm using stethoscope based internal sound sensor. *Journal of Intelligent Manufacturing* 34(3), 1427-1444. doi: 10.1016/j.ymssp.2004.10.013.
- Jia, H. & Liu, W., 2023. Anomaly detection in images with shared autoencoders. *Frontiers in Neurorobotics* 16, 1-11. doi: 10.3389/fnbot.2022.1046867.
- Kozamernik, N. & Bračun, D., 2025. A novel FuseDecode Autoencoder for industrial visual inspection: Incremental anomaly detection improvement with gradual transition from unsupervised to mixed-supervision learning with reduced human effort. *Computers in Industry* 164, 1-19. doi: 10.1016/j.compind.2024.104198.
- Ruediger-Flore, P., Klar, M., Hussong, M., Mukherjee, A., Glatt, M. & Aurich, J. C., 2024. Comparing Binary Classification and Autoencoders for Vision-Based Anomaly Detection in Material Flow. *Procedia CIRP* 121, 138-143. doi: 10.1016/j.procir.2023.09.241.
- Ragab, M. G., Abdulkadir, S. J., Muneer, A., Alqushaibi, A., Sumiea, E. H., Qureshi, R. et al., 2024. A comprehensive systematic review of YOLO for medical object detection (2018 to 2023). *IEEE Access* 12, 57815-57836. doi: 10.1109/ACCESS.2024.3386826.

- Hussain, M., 2024. Unveiling each variant—a comprehensive review of yolo. *IEEE access* 12, 42816-42833. doi: 10.1109/ACCESS.2024.3378568.
- Padilla, R., Passos, W. L., Dias, T. L., Netto, S. L., & Da Silva, E. A., 2021. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics* 10(3), 1-28. doi: 10.3390/electronics10030279.
- Kaur, R., & Singh, S., 2023. A comprehensive review of object detection with deep learning. *Digital Signal Processing* 132, 1-17. doi: 10.1016/j.dsp.2022.103812.
- Casas, E., Ramos, L., Bendek, E., & Rivas-Echeverría, F., 2023. Assessing the effectiveness of YOLO architectures for smoke and wildfire detection. *IEEE Access* 11, 96554-96583. doi: 10.1109/ACCESS.2023.3312217.
- Cao, L., Shen, Z., & Xu, S., 2024. Efficient forest fire detection based on an improved YOLO model. *Visual Intelligence* 2(20), 1-7. doi: 10.1007/s44267-024-00053-y.
- Mansour, R. F., Escoria-Gutierrez, J., Gamarra, M., Gupta, D., Castillo, O., & Kumar, S. 2021. Unsupervised deep learning based variational autoencoder model for COVID-19 diagnosis and classification. *Pattern Recognition Letters*, 151, 267-274.
- Wei, R., & Mahmood, A., 2020. Recent advances in variational autoencoders with representation learning for biomedical informatics: A survey. *ieee Access* 9, 4939-4956. doi: 0.1109/ACCESS.2020.3048309.
- Najjar, Y., A., 2024. Comparative analysis of image quality assessment metrics: MSE, PSNR, SSIM and FSIM. *International Journal of Science and Research* 13(3), 110-114. doi: orcid.org/0000-0002-3369-4999.
- Nahm, F. S., 2022. Receiver operating characteristic curve: overview and practical use for clinicians. *Korean journal of anesthesiology* 75(1), 25-36. doi: 10.4097/kja.21209.

LAMPIRAN

Lampiran 1. Peralatan penelitian

| No. | Alat dan Bahan | Fungsi |
|-----|---|---|
| 1 |  | Berfungsi sebagai mikrokontroler utama yang mengendalikan motor servo pada lengan robot dan menerima sinyal dari sensor. |
| 2 |  | digunakan sebagai aktuator untuk menggerakkan bagian-bagian lengan robot sesuai perintah dari arduino. |
| 3 |  | struktur mekanik tempat pemasangan motor servo dan berperan sebagai sistem penggerak robotik. |
| 4 |  | Berfungsi menyediakan tegangan stabil untuk pengoperasian motor servo. |
| 5 |  | Digunakan untuk mendeteksi pergerakan dan membantu menghitung jumlah kontainer cacat dan maupun tidak cacat yang melintas. |
| 6 |  | Digunakan untuk mengambil citra kontainer kimia, yang akan diproses oleh model deteksi objek (YOLO) dan deteksi kecacatan (<i>autoencoder</i>). |
| 7 |  | Digunakan untuk memuat program ke Arduino Uno, serta menjalankan model deteksi objek berbasis YOLO dan <i>autoencoder</i> untuk analisis visual. |

| No. | Alat dan Bahan | Fungsi |
|-----|---|--|
| 8 |  | berfungsi menghubungkan berbagai komponen elektronik seperti sensor dan aktuator ke papan rangkaian dan Arduino. |
| 9 |  | berfungsi untuk menyediakan jalur koneksi antar komponen. |
| | Papan Rangkaian | |

Lampiran 2. Program untuk latih model YOLO

```
from ultralytics import YOLO
model = YOLO("yolo12n.pt")
results = model.train(data=config, epochs=50)
inference = model("frame_02318.jpg")
inference[0].show()
```

Lampiran 3. Program untuk latih model *autoencoder*

```
import os
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import transforms, datasets
import torchvision.transforms as transforms
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np

class CVAE(nn.Module):
    def __init__(self, latent_dim=128):
        super(CVAE, self).__init__()
        self.latent_dim = latent_dim
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(256, 512, kernel_size=4, stride=2, padding=1),
            nn.ReLU()
        )
        self.fc_mu = nn.Linear(512 * 4 * 4, latent_dim)
        self.fc_logvar = nn.Linear(512 * 4 * 4, latent_dim)
        self.decoder_input = nn.Linear(latent_dim, 512 * 4 * 4)
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 32, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 3, kernel_size=4, stride=2, padding=1),
```

```

        nn.Sigmoid()
    )

def reparameterize(self, mu, logvar):
    """Applies the reparameterization trick to sample from N(mu, var)."""
    std = torch.exp(0.5 * logvar)
    eps = torch.randn_like(std)
    return mu + eps * std

def forward(self, x):
    enc = self.encoder(x)
    enc = enc.view(x.size(0), -1)
    mu = self.fc_mu(enc)
    logvar = self.fc_logvar(enc)
    z = self.reparameterize(mu, logvar)
    dec_input = self.decoder_input(z)
    dec_input = dec_input.view(x.size(0), 512, 4, 4)
    reconstruction = self.decoder(dec_input)
    return reconstruction, mu, logvar

def loss_function(recon_x, x, mu, logvar):
    """
    Compute the VAE loss as the sum of a reconstruction loss (MSE)
    and the KL divergence loss.
    """
    recon_loss = F.mse_loss(recon_x, x, reduction='sum')
    kl_loss = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
    return recon_loss + kl_loss

def train(model, dataloader, optimizer, device):
    model.train()
    train_loss = 0
    for batch_idx, (data, _) in enumerate(dataloader):
        data = data.to(device)
        optimizer.zero_grad()
        recon, mu, logvar = model(data)
        loss = loss_function(recon, data, mu, logvar)
        loss.backward()
        train_loss += loss.item()
        optimizer.step()
        if batch_idx % 100 == 0:
            print(f'Batch {batch_idx}/{len(dataloader)} Loss: {loss.item()}/{len(data)}')

```

```

avg_loss = train_loss / len(dataloader.dataset)
print(f'===== Average training loss: {avg_loss:.4f}')


def infer_anomaly_with_heatmap(image_path):
    """Runs inference on a single image and highlights anomalies with a heatmap."""

    image = Image.open(image_path).convert("RGB")
    image_tensor = transform(image).unsqueeze(0).to(device)

    with torch.no_grad():
        reconstructed, _, _ = model(image_tensor)

    anomaly_map = torch.abs(image_tensor - reconstructed).mean(dim=1, keepdim=True)

    anomaly_map = anomaly_map.squeeze().cpu().numpy()
    anomaly_map = (anomaly_map - anomaly_map.min()) / (anomaly_map.max() - anomaly_map.min())

    orig_np = image_tensor.squeeze(0).permute(1, 2, 0).cpu().numpy()
    recon_np = reconstructed.squeeze(0).permute(1, 2, 0).cpu().numpy()

    fig, axs = plt.subplots(1, 3, figsize=(12, 4))

    axs[0].imshow(orig_np)
    axs[0].set_title("Original Image")
    axs[0].axis("off")

    axs[1].imshow(recon_np)
    axs[1].set_title("Reconstructed Image")
    axs[1].axis("off")

    axs[2].imshow(orig_np)
    axs[2].imshow(anomaly_map, cmap='jet', alpha=0.5)
    axs[2].set_title("Anomaly Heatmap")
    axs[2].axis("off")

    plt.show()

mse_loss = torch.mean((reconstructed - image_tensor) ** 2).item()
print(f'Reconstruction Error (MSE): {mse_loss:.6f}')

threshold = 0.007183

```

```
is_anomalous = mse_loss > threshold
print(f"Anomaly Detected: {is_anomalous}")

return mse_loss, is_anomalous

if __name__ == "__main__":
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    transform = transforms.Compose([
        transforms.Resize((128, 128)),
        transforms.ToTensor(),
    ])

    train_dir = "data_ae/train"

    train_dataset = datasets.ImageFolder(train_dir, transform=transform)

    train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True, num_workers=4)

    model = CVAE(latent_dim=128).to(device)
    optimizer = optim.Adam(model.parameters(), lr=1e-3)
    epochs = 50

    for epoch in range(1, epochs + 1):
        print(f"Epoch {epoch}/{epochs}")
        train(model, train_loader, optimizer, device)
```

Lampiran 4. Program kombinasi YOLO dan *autoencoder*

```
import cv2
import torch
import torch.nn as nn
import torchvision.transforms as transforms
from PIL import Image
from ultralytics import YOLO
import time
import collections
import serial
import os
from datetime import datetime
import matplotlib.pyplot as plt

class CVAE(nn.Module):
    def __init__(self, latent_dim=128):
        super(CVAE, self).__init__()
        self.latent_dim = latent_dim
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(256, 512, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
        )
        self.fc_mu = nn.Linear(512 * 4 * 4, latent_dim)
        self.fc_logvar = nn.Linear(512 * 4 * 4, latent_dim)
        self.decoder_input = nn.Linear(latent_dim, 512 * 4 * 4)
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 32, kernel_size=4, stride=2, padding=1),
            nn.ReLU(),
```

```

        nn.ConvTranspose2d(32, 3, kernel_size=4, stride=2, padding=1),
        nn.Sigmoid(),
    )

def reparameterize(self, mu, logvar):
    std = torch.exp(0.5 * logvar)
    eps = torch.randn_like(std)
    return mu + eps * std

def forward(self, x):
    enc = self.encoder(x)
    enc = enc.view(x.size(0), -1)
    mu = self.fc_mu(enc)
    logvar = self.fc_logvar(enc)
    z = self.reparameterize(mu, logvar)
    dec_input = self.decoder_input(z)
    dec_input = dec_input.view(x.size(0), 512, 4, 4)
    reconstruction = self.decoder(dec_input)
    return reconstruction, mu, logvar

def get_centroid(x1, y1, x2, y2):
    return ((x1 + x2) // 2, (y1 + y2) // 2)

def check_anomaly(crop_img):
    image = Image.fromarray(cv2.cvtColor(crop_img, cv2.COLOR_BGR2RGB))
    image_tensor = transform(image).unsqueeze(0).to(device)
    with torch.no_grad():
        reconstructed, _, _ = ae_model(image_tensor)
    mse_loss = torch.mean((reconstructed - image_tensor) ** 2).item()
    return mse_loss

def save_anomaly_heatmap(image_tensor, reconstructed, loss_value, obj_id):
    anomaly_map = torch.abs(image_tensor - reconstructed).mean(dim=1, keepdim=True)
    anomaly_map = anomaly_map.squeeze().cpu().numpy()
    anomaly_map = (anomaly_map - anomaly_map.min()) / (
        anomaly_map.max() - anomaly_map.min() + 1e-8
    )

    orig_np = image_tensor.squeeze(0).permute(1, 2, 0).cpu().numpy()
    recon_np = reconstructed.squeeze(0).permute(1, 2, 0).cpu().numpy()

    fig, axs = plt.subplots(1, 3, figsize=(12, 4))

```

```

    axs[0].imshow(orig_np)
    axs[0].set_title("Original Image")
    axs[0].axis("off")

    axs[1].imshow(recon_np)
    axs[1].set_title("Reconstructed Image")
    axs[1].axis("off")

    axs[2].imshow(orig_np)
    axs[2].imshow(anomaly_map, cmap="jet", alpha=0.5)
    axs[2].set_title(f"Anomaly Heatmap\nLoss: {loss_value:.6f}")
    axs[2].axis("off")

    os.makedirs("output", exist_ok=True)
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"output/heatmap_{obj_id}_{timestamp}.png"
    plt.savefig(filename)
    plt.close(fig)

    print(f"[Saved] Heatmap image saved to {filename}")

TRANSFORM = transforms.Compose([transforms.Resize((128, 128)), transforms.ToTensor()])
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
AE_MODEL_PATH = "ae.pth"
YOLO_MODEL_PATH = "./runs/detect/train/weights/best.pt"
CONFIDENCE_THRESHOLD = 0.90
LOSS_BATCH_SIZE = 20
DELAY_SECONDS = 0.005
OBJECT_WARMUP_FRAMES = 5
CENTROID_TOLERANCE = 30
SPIKE_FILTER_THRESHOLD = 0.0099
SERIAL_PORT = "/dev/ttyACM0"
BAUD_RATE = 9600
DEFECT_THRESHOLD = 0.007183

print("Loading models...")
device = DEVICE
transform = TRANSFORM
ae_model = CVAE(latent_dim=128).to(device)
ae_model.load_state_dict(torch.load(AE_MODEL_PATH, map_location=device))
ae_model.eval()

```

```

yolo_model = YOLO(YOLO_MODEL_PATH)
print(f"Models loaded successfully on {device}.")

print("Initializing camera...")
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Cannot open camera")
    exit()
print("Camera initialized.")

arduino = None
try:
    print(f"Connecting to Arduino on {SERIAL_PORT}...")
    arduino = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
    time.sleep(2)
    print("Arduino connected.")
except serial.SerialException as e:
    print(f"WARNING: Could not connect to Arduino. {e}")
    print("--> Running in SIMULATION MODE. No data will be sent.")

object_history = collections.defaultdict(dict)

print("\nStarting monitoring... Press Ctrl+C to exit.")
try:
    while True:
        ret, frame = cap.read()
        if not ret:
            print("Failed to grab frame. Exiting.")
            break

        results = yolo_model(frame, verbose=False)[0]
        current_time = time.time()

        for box in results.boxes:
            if box.conf[0] < CONFIDENCE_THRESHOLD:
                continue

            x1, y1, x2, y2 = map(int, box.xyxy[0])
            crop = frame[y1:y2, x1:x2]
            centroid = get_centroid(x1, y1, x2, y2)

            for obj_id in list(object_history.keys()):

```

```

if (
    current_time - object_history[obj_id].get("last_seen", current_time)
    > 2
):
    print(f"[Cleanup] Removing stale object ID {obj_id}")
    del object_history[obj_id]

matched_id = None
for obj_id, data in object_history.items():
    prev_centroid = data.get("last_centroid")
    if (
        prev_centroid
        and abs(prev_centroid[0] - centroid[0]) < CENTROID_TOLERANCE
        and abs(prev_centroid[1] - centroid[1]) < CENTROID_TOLERANCE
    ):
        data["seen_frames"] += 1
        data["last_centroid"] = centroid
        data["last_seen"] = current_time
        matched_id = obj_id
        break

if matched_id is None:
    matched_id = len(object_history)
    object_history[matched_id] = {
        "seen_frames": 1,
        "last_centroid": centroid,
        "valid_losses": [],
        "first_sent": False,
        "last_seen": current_time,
    }
    print(f"[New Object] Assigned ID {matched_id}")

current_obj = object_history[matched_id]

if current_obj["seen_frames"] <= OBJECT_WARMUP_FRAMES:
    print(
        f"[Warmup] Skipping object {matched_id}, frame {current_obj['seen_frames']}"
    )
    continue

anomaly_score = check_anomaly(crop)

```

```

if anomaly_score > SPIKE_FILTER_THRESHOLD:
    print(f"[Spike Filter] Score {anomaly_score:.6f} too high, ignored.")
    current_obj["valid_losses"].clear()
    continue

current_obj["valid_losses"].append(anomaly_score)

if len(current_obj["valid_losses"]) >= LOSS_BATCH_SIZE:
    avg_error = sum(current_obj["valid_losses"]) / len(
        current_obj["valid_losses"])
)

if not current_obj["first_sent"]:
    print(
        f"[Skip First Send] Skipping first Arduino send for object {current_obj['id']}")
    current_obj["first_sent"] = True
    current_obj["valid_losses"].clear()
    continue

image_tensor = (
    transform(Image.fromarray(cv2.cvtColor(crop, cv2.COLOR_BGR2RGB))
        .unsqueeze(0)
        .to(device))
)
with torch.no_grad():
    reconstructed, _, _ = ae_model(image_tensor)

save_anomaly_heatmap(image_tensor, reconstructed, avg_error, matched_id)

print(f"[*] Object {matched_id} final avg loss: {avg_error:.6f}")
label = "DEFECT" if avg_error > DEFECT_THRESHOLD else "OK"
msg = f"{label},{avg_error:.6f}\n"

if arduino:
    try:
        print(f"--> Sending to Arduino: {msg.strip()}")
        arduino.write(msg.encode())
    except serial.SerialException as e:
        print(f"ERROR: Arduino write failed: {e}. Disconnecting.")
        arduino.close()
    arduino = None

```

```
    else:
        print(f"--> [SIMULATED] Arduino message: {msg.strip()}")

        current_obj["valid_losses"].clear()
    else:
        print(
            f"[Collecting] Object {matched_id} sample {len(current_obj['val
        )

    time.sleep(DELAY_SECONDS)

except KeyboardInterrupt:
    print("\nInterruption detected. Shutting down.")
finally:
    cap.release()
    print("Camera released.")
    if arduino and arduino.is_open:
        arduino.close()
        print("Arduino connection closed.")
    print("Exited gracefully.")
```

Lampiran 5. Program lengan robot

```
#include <Servo.h>
Servo servo2;
Servo servo3;
Servo servo4;

void write_servo(int from, int to, int delay_rotate, Servo &servo) {
    if (from < to) {
        for (int pos = from; pos <= to; pos++) {
            servo.write(pos);
            delay(delay_rotate);
        }
    } else {
        for (int pos = from; pos >= to; pos--) {
            servo.write(pos);
            delay(delay_rotate);
        }
    }
}

void setup() {
    servo2.attach(5);
    servo3.attach(6);
    servo4.attach(9);

    servo2.write(85);
    servo3.write(30);
    servo4.write(170);
    delay(2000);

    Serial.begin(9600);
}

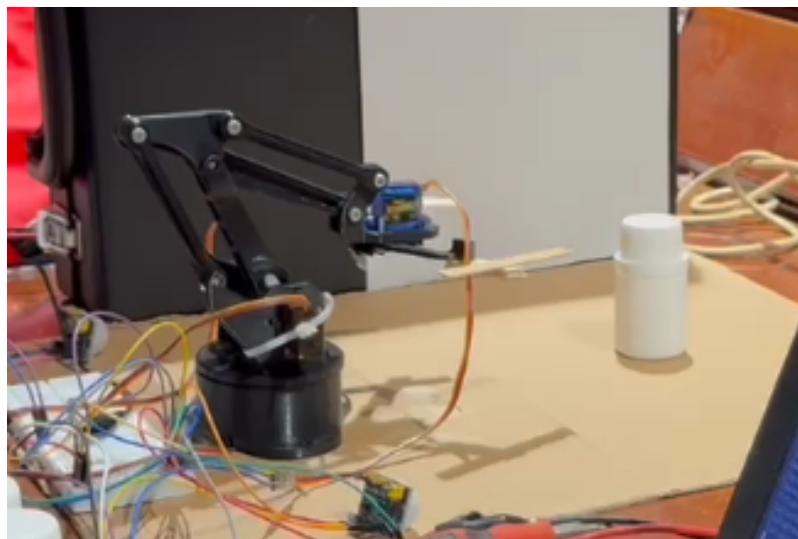
void loop() {
    if (Serial.available()) {
        String msg = Serial.readStringUntil('\n');

        int sepIndex = msg.indexOf(',');
        if (sepIndex > 0) {
            String label = msg.substring(0, sepIndex);
            String errorStr = msg.substring(sepIndex + 1);
            float errorVal = errorStr.toFloat();
        }
    }
}
```

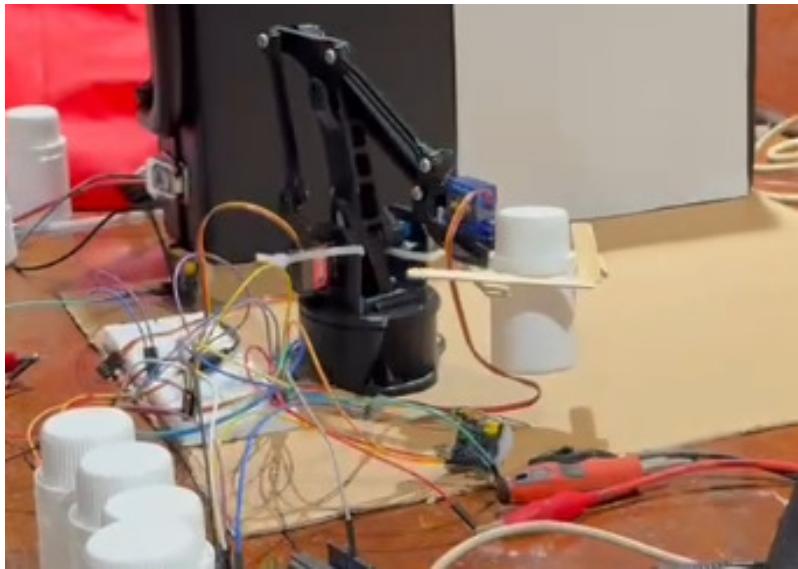
```
if (label == "DEFECT" || label == "OK") {  
    Serial.print(label);  
    Serial.print(",");  
    Serial.println(errorVal, 5);  
  
    if (label == "DEFECT") {  
        // KIRI  
        write_servo(50, 120, 15, servo3);  
        delay(2000);  
  
        write_servo(170, 30, 5, servo4);  
        delay(2000);  
  
        write_servo(120, 50, 15, servo3);  
        delay(2000);  
  
        write_servo(85, 180, 30, servo2);  
        delay(2000);  
  
        write_servo(30, 170, 5, servo4);  
        delay(2000);  
  
        write_servo(180, 85, 30, servo2);  
        delay(2000);  
  
    } else if (label == "OK") {  
        // KANAN  
        write_servo(50, 120, 15, servo3);  
        delay(2000);  
  
        write_servo(170, 30, 5, servo4);  
        delay(2000);  
  
        write_servo(120, 50, 15, servo3);  
        delay(2000);  
  
        write_servo(85, 0, 30, servo2);  
        delay(2000);  
  
        write_servo(30, 170, 5, servo4);  
        delay(2000);  
    }  
}
```

```
    write_servo(0, 85, 30, servo2);
    delay(2000);
}
}
}
}
}
```

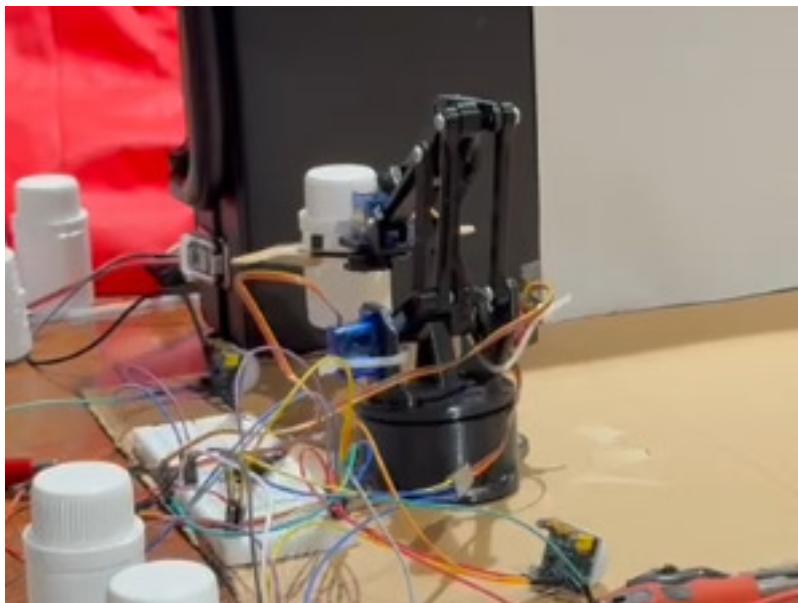
Lampiran 6. Dokumentasi pergerakan lengan robot



Lengan robot menunggu hasil deteksi cacat



Lengan robot ketika meyortir kontainer normal



Lengan robot ketika meyortir kontainer cacat