SKRIPSI

OPTIMASI GERAKAN ROBOT SEPAK BOLA MENGGUNAKAN *KALMAN FILTER* PADA *TRACKING* OBJEK BERBASIS YOLO

Diajukan untuk Memenuhi
Persyaratan Meraih Gelar Sarjana Teknik
Teknik Informatika Fakultas Teknik Universitas Riau

Oleh:

Fikri Rivandi
NIM: 2207112583

PROGRAM STUDI TEKNIK INFORMATIKA S1
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS RIAU
2025

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi di era modern telah memberikan dampak signifikan pada berbagai aspek kehidupan manusia. Salah satu bidang yang mengalami kemajuan pesat adalah teknologi *artificial intelligence* (AI) dan *machine learning* (ML) (Widodo et al., 2024). Teknologi ini memungkinkan sistem komputer tidak hanya melakukan perhitungan, tetapi juga belajar dari data, mengenali pola, serta mengambil keputusan secara mandiri (Wijaya & Yuniarto, 2024). Salah satu cabang dari AI yang banyak dimanfaatkan adalah *computer vision* (CV). CV merupakan cabang dari kecerdasan buatan yang berfokus pada kemampuan komputer dan sistem untuk mengekstraksi informasi bermakna dari citra digital, video, maupun data visual lainnya. Dalam sistem yang bersifat dinamis, informasi visual ini digunakan untuk mengenali, memantau, serta melacak (*tracking*) posisi objek secara kontinu guna mengambil keputusan atau menjalankan tindakan tertentu secara akurat (Baskoro et al., 2022).

Robotika merupakan bidang interdisipliner yang berfokus pada kajian mengenai robot, di mana ilmu pengetahuan, teknologi, dan rekayasa saling berhubungan. Hasil akhirnya berupa mesin yang dapat menirukan perilaku manusia atau menjalankan instruksi tertentu melalui pemrograman (Hendrik & Awal, 2023). Robot konvensional umumnya didesain untuk beroperasi pada lingkungan yang terstruktur dan terbatas dengan algoritma tugas yang telah diprogram secara statis. Akan tetapi, kenyataannya lingkungan kerja sering bersifat dinamis dan tidak terduga, sehingga dibutuhkan sistem yang lebih adaptif dan fleksibel. Pada titik inilah kecerdasan buatan memiliki peran penting. Dengan memanfaatkan teknologi seperti ML, CV, dan kecerdasan buatan lainnya, robot modern tidak hanya mampu mengenali objek, tetapi juga dituntut untuk melakukan *tracking* posisi objek secara kontinu guna memahami kondisi lingkungan dan mengambil keputusan yang lebih responsif serta mandiri (Ritonga & Hasibuan, 2025).

Salah satu implementasi nyata robotika yang mendapat perhatian khusus adalah robot sepak bola, khususnya dalam Kontes Robot Sepak Bola Indonesia (KRSBI) Beroda. Dalam ajang ini, robot dituntut untuk mampu beroperasi secara mandiri di tengah lingkungan lapangan yang sangat dinamis, di mana kecepatan dan ketepatan dalam melakukan *tracking* terhadap pergerakan bola menjadi faktor krusial yang menentukan keberhasilan sistem navigasi dan strategi permainan robot.

Universitas Riau (UNRI) memiliki klub robotika bernama *Engineering Robotic Club* (ERC) yang berfokus pada pengembangan robot KRSBI Beroda. Pada tahun 2021 dan 2022, tim ERC UNRI berhasil mencapai tingkat nasional, yang menjadi pencapaian signifikan dalam sejarah keikutsertaannya di KRSBI Beroda. Sejak berpartisipasi pertama kali pada tahun 2019, ERC UNRI telah mengalami berbagai perkembangan signifikan, baik dalam aspek perangkat keras maupun perangkat lunak. Namun, perubahan regulasi sejak tahun 2023 membuat tim harus melakukan penyesuaian ulang agar robot tetap kompetitif dan relevan.

Dalam pertandingan KRSBI Beroda, kemampuan *tracking* bola dan gawang menjadi faktor yang sangat krusial. Awalnya, metode *tracking* objek berbasis HSV (*Hue, Saturation, Value*) digunakan karena ringan dan dapat berjalan *real-time*. Citra dengan format HSV dimanfaatkan dalam penerapan algoritma *image thresholding* guna mendeteksi warna objek. Selanjutnya, hasil dari proses *thresholding* tersebut digunakan dalam algoritma *Convex Hull*, di mana apabila bola berhasil terdeteksi, maka sistem akan menghitung ukuran objek, menentukan titik pusatnya, serta memperoleh koordinat posisi objek (x, y) relatif terhadap posisi robot saat ini. Kelemahan metode HSV adalah ketergantungannya pada kondisi pencahayaan, sehingga akurasi *tracking* menjadi tidak stabil dalam situasi lapangan yang dinamis (Nanda et al., 2023).

Seiring perkembangan teknologi CV, metode *deep learning* seperti YOLO (*You Only Look Once*) akhirnya digunakan untuk meningkatkan akurasi *tracking* objek pada robot KRSBI Beroda milik ERC UNRI. YOLO bekerja dengan membagi citra ke dalam *grid* dan memprediksi *bounding box* serta kelas objek secara langsung dalam satu tahap komputasi, sehingga mampu memberikan

tracking real-time dengan akurasi yang tinggi (F. B. Saputra et al., 2023). Secara konsep, metode ini sangat ideal untuk kebutuhan robot sepak bola beroda, yang menuntut respon cepat terhadap pergerakan bola di lapangan.

Namun dalam implementasi YOLO pada robot KRSBI Beroda, masalah pertama yang didapat adalah kendala signifikan terkait keterbatasan sumber daya perangkat keras. Sistem robot saat ini menggunakan laptop ASUS K401U sebagai unit pemrosesan utama, yang hanya memiliki spesifikasi standar seperti prosesor Intel Core i5-7200U, RAM 8GB, serta dukungan GPU Nvidia 940MX. Spesifikasi tersebut belum cukup mumpuni untuk menjalankan YOLO dengan baik. Akibatnya, proses *tracking* sering mengalami penurunan performa, seperti *lag* dan ketidakstabilan *frame rate*, sehingga informasi posisi bola diterima oleh robot dengan jeda waktu. Keterlambatan ini berdampak langsung terhadap kualitas pergerakan robot. Robot dapat terlambat merespons perubahan posisi bola, bergerak tidak stabil, atau bahkan salah memperkirakan arah gerak bola. Dengan kata lain, meskipun YOLO memiliki akurasi tinggi, keterbatasan perangkat keras pada robot KRSBI Beroda menyebabkan efisiensinya menurun dan menjadi salah satu sumber utama masalah dalam sistem navigasi berbasis visi.

Miharja et al. (2025) menyebutkan bahwa metode YOLO adalah salah satu cara efektif dalam *object detection*, tapi tidak cukup efisien untuk sumber daya perangkat keras yang digunakan, karena membutuhkan kapasitas komputasi yang mumpuni. Pada penelitian lainnya juga menyebutkan bahwa menggunakan perangkat *Single Board Computer* berkinerja tinggi, seperti NVIDIA Jetson Series, juga perlu dipertimbangkan untuk meningkatkan kecepatan deteksi (FPS) guna mencapai performa *tracking* objek secara *real-time* yang lebih optimal (Firdaus & Lelono, 2025).

Selain itu, pada proses *tracking* objek juga sering terjadi permasalahan *occlusion* (oklusi) yang merupakan masalah kedua bagi penelitian ini. Fenomena oklusi, di mana sebagian objek tertutup oleh penghalang tertentu, merupakan masalah fundamental dalam *tracking* objek. Oklusi merupakan penyebab utama yang meningkatkan *false-negative detection rate*, yang dapat menurunkan kinerja *tracking* secara keseluruhan. Objek yang teroklusi ini dikategorikan sebagai *hard-*

positive examples yang sulit dideteksi oleh model. Oleh karena itu, oklusi adalah hal yang harus diperhatikan untuk mencapai optimalitas dalam penggunaan *object detection* (Ryu & Chung, 2021).

Untuk menjawab tantangan tersebut, penelitian ini mengusulkan penggunaan *Kalman Filter* pada hasil deteksi bola berbasis YOLO. *Kalman Filter* merupakan salah satu algoritma estimasi yang banyak digunakan dalam sistem dinamis karena mampu mengestimasi keadaan berikutnya dari suatu objek berdasarkan data pengamatan sebelumnya. Metode *Kalman Filter* menggunakan informasi dari objek yang terdeteksi di suatu *frame* dan status objek dari *frame* sebelumnya untuk mendapatkan status yang baru dari objek tersebut (C. Saputra, 2023).

Pada konteks robot sepak bola, *Kalman Filter* sangat relevan untuk digunakan karena bola sering mengalami perubahan posisi secara cepat, mendadak, dan terkadang tidak terduga. YOLO sebagai detektor objek hanya memberikan posisi bola pada setiap *frame*, tanpa mempertimbangkan dinamika gerakan dari bola tersebut. Akibatnya, jika bola bergerak terlalu cepat atau terjadi keterlambatan proses inferensi, robot bisa kehilangan akurasi dalam mengejar bola. Dengan *Kalman Filter*, sistem tidak hanya bergantung pada deteksi saat ini, tetapi juga dapat mengestimasi posisi bola pada *frame* berikutnya, sehingga pergerakan robot menjadi lebih stabil, responsif, dan efisien.

Selain itu, *Kalman Filter* memiliki keunggulan dari sisi efisiensi komputasi. Dibandingkan dengan metode estimasi berbasis *deep learning* yang umumnya membutuhkan sumber daya tinggi, *Kalman Filter* relatif ringan dan dapat diimplementasikan pada perangkat dengan keterbatasan komputasi seperti robot KRSBI Beroda. Penelitian sebelumnya yang dilakukan oleh Yuztiawan & Utaminingrum (2017) juga memperkuat hal ini, di mana integrasi model YOLOv8N dengan *Kalman Filter* pada kondisi lingkungan dengan banyak objek mampu meningkatkan akurasi dan stabilitas deteksi serta *tracking* hingga 91,66%. Bahkan, penggunaan *Kalman Filter* terbukti meningkatkan kinerja *tracking* sebesar 25% dibandingkan hanya menggunakan YOLOv8N saja. Menariknya, penambahan algoritma tersebut hanya memberikan tambahan waktu komputasi rata-rata sekitar 0,0076 detik per *frame* (sekitar 7,92%), sehingga sistem tetap mampu bekerja

secara *real-time*. Dengan demikian, *Kalman Filter* tidak hanya meningkatkan akurasi *tracking*, tetapi juga tetap mempertahankan efisiensi komputasi yang sangat penting bagi robot kompetitif dengan keterbatasan perangkat keras.

Berdasarkan permasalahan yang telah diuraikan, peneliti tertarik untuk meneliti dengan judul "Optimasi Gerakan Robot Sepak Bola Menggunakan *Kalman Filter* Pada *Tracking* Objek Berbasis YOLO". Penelitian ini bertujuan untuk mengembangkan dan mengimplementasikan sistem *tracking* bola pada robot KRSBI Beroda dengan memanfaatkan kamera *omnidirectional* dan pendekatan *deep learning* berbasis CNN. Deteksi objek akan dilakukan menggunakan algoritma YOLO, sementara estimasi posisi bola diperkuat dengan penerapan *Kalman Filter* untuk mengatasi keterbatasan *tracking* berbasis *frame* tunggal. Dengan kombinasi ini, robot diharapkan tidak hanya mampu mengenali bola dan gawang secara akurat, tetapi juga dapat melakukan respons gerakan yang lebih mulus, stabil, dan efisien meskipun bola bergerak cepat atau kondisi lapangan penuh gangguan visual.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka permasalahan dalam penelitian ini dapat dirumuskan ke dalam beberapa pertanyaan sebagai berikut:

1. Bagaimana meningkatkan kestabilan *tracking* bola pada robot KRSBI Beroda ketika YOLO mengalami keterbatasan performa akibat rendahnya kapasitas perangkat keras (laptop ASUS K401U) sehingga menghasilkan *frame rate* yang tidak stabil?
2. Bagaimana mengurangi dampak oklusi yang sering terjadi pada proses *tracking* bola berbasis YOLO, sehingga robot tetap mampu mengetahui dan mengestimasi posisi bola secara konsisten pada kondisi lapangan yang dinamis?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dikemukakan, maka tujuan dari penelitian ini adalah sebagai berikut:

1. Mengembangkan sistem *tracking* bola yang lebih stabil dan responsif pada robot KRSBI Beroda dengan menggunakan algoritma *Kalman Filter* untuk mengatasi keterbatasan perangkat keras.
2. Meningkatkan kontinuitas *tracking* bola dengan memanfaatkan kemampuan estimasi *Kalman Filter* untuk mengurangi pengaruh oklusi, sehingga robot dapat bergerak lebih efisien, tepat sasaran, dan adaptif terhadap dinamika permainan.

1.4 Batasan Masalah

Adapun batasan masalah dari penelitian ini adalah sebagai berikut:

1. Penelitian ini hanya membahas *tracking* bola menggunakan kamera *omnidirectional* pada robot KRSBI Beroda.
2. Algoritma deteksi objek yang digunakan adalah YOLO yang sudah ada, tanpa melakukan pengembangan arsitektur baru.
3. Algoritma estimasi posisi bola dilakukan dengan menggunakan *Kalman Filter*, tanpa membandingkannya dengan algoritma estimasi lain.
4. Fokus penelitian adalah pada optimasi gerakan robot terhadap bola, tidak mencakup aspek strategi tim, komunikasi antar robot, atau kontrol perangkat keras robot secara mendetail.

1.5 Manfaat Penelitian

Berikut adalah manfaat dilakukannya penelitian ini bagi beberapa pihak:

1. Peneliti akan memperoleh pengalaman berharga dalam mengimplementasikan algoritma YOLO yang dipadukan dengan *Kalman Filter* pada sistem robotika, khususnya dalam konteks *tracking* objek bergerak.
2. Penelitian ini dapat membantu meningkatkan performa robot dalam pertandingan melalui sistem *tracking* bola yang lebih akurat dan gerakan robot yang lebih stabil serta efisien.

1.6 Sistematika Penulisan

Untuk mempermudah dalam memahami lebih jelas tentang penulisan penelitian ini, maka penelitian ini ditulis dalam beberapa bab yang masing-masing berkaitan satu sama lainnya, dengan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Bagian ini berisi tentang deskripsi umum dari penelitian yang akan dilakukan meliputi Latar Belakang, Perumusan Masalah, Tujuan Penelitian, Batasan Masalah, Manfaat Penelitian dan Sistematika Penulisan.

BAB II LANDASAN TEORI

Bagian ini membahas penelitian terdahulu, teori-teori dan pendapat para ahli yang berhubungan dengan penelitian yang dilakukan.

BAB III METODOLOGI PENELITIAN

Bagian ini berisi tentang alat dan bahan penelitian yang dilakukan, metode dan alur penelitian, metode pengembangan sistem cerdas, metode pengumpulan data, teknik mengolah data, dan teknik menguji hasil olahan data.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang hasil perancangan dan analisa yang telah dilakukan sesuai dengan metodologi penelitian, sekaligus mengevaluasi hasil pengujian terhadap parameter-parameter uji yang telah ditetapkan.

BAB V PENUTUP

Bab ini menjelaskan tentang simpulan hasil penelitian yang diperoleh sesuai dengan tujuan penelitian serta memuat saran mengenai masalah dan kemungkinan pemecahannya untuk penelitian selanjutnya.

DAFTAR PUSTAKA

LAMPIRAN

BAB II
LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian mengenai sistem deteksi objek, khususnya dalam penggunaan model *Convolutional Neural Network* (CNN) seperti *You Only Look Once* (YOLO), telah banyak dilakukan sebelumnya. Penelitian-penelitian tersebut memberikan kontribusi penting dalam pengembangan metode deteksi objek maupun optimasi model menggunakan algoritma bantuan seperti estimator. Dengan meninjau penelitian terdahulu, penulis memperoleh wawasan terkait kelemahan metode sebelumnya, potensi pengembangan, serta peluang penerapan algoritma terbaru yang lebih efektif.

Pertama, penelitian yang berjudul "*CNN-Based Ball and Goal Detection for KRSBI Robot with Omnidirectional Camera*" dilakukan oleh Farhan & Candra (2025). Penelitian ini berfokus pada pengembangan sistem deteksi bola dan gawang berbasis *computer vision* pada robot yang mengikuti Kompetisi Robot Sepak Bola Indonesia (KRSBI) Beroda menggunakan kamera *omnidirectional*. Metode tradisional seperti HSV *color filtering* sebelumnya banyak digunakan karena implesmentasinya sederhana, namun performanya sangat bergantung pada intensitas cahaya sehingga tidak stabil dalam kondisi pencahayaan yang bervariasi. Untuk mengatasi kelemahan tersebut, penelitian ini mengusulkan penerapan algoritma YOLO yang berbasis CNN guna meningkatkan akurasi dan keandalan deteksi objek secara *real-time*. Dataset yang digunakan terdiri dari 1.125 citra dengan variasi pencahayaan dan posisi objek yang berbeda, kemudian dibagi menjadi 80% data pelatihan dan 20% data validasi. Model YOLOv8 dilatih menggunakan *Ultralytics* di platform Google Colab selama 100 *epoch*. Hasil pelatihan menunjukkan performa deteksi yang sangat tinggi, dengan tingkat akurasi sebesar 95,87%, *precision* mencapai 1.00 pada *confidence threshold* 0.921, *recall* sebesar 0.99, dan nilai *F1-Score* maksimum 0.97 pada *confidence* 0.149. Berdasarkan kurva *precision–confidence*, model dapat menghasilkan deteksi tanpa *false positive* pada ambang kepercayaan tinggi, sedangkan kurva *recall–confidence*

menunjukkan kemampuan model mendeteksi hampir semua objek meskipun pada nilai *confidence* yang rendah. Hasil penelitian ini membuktikan bahwa model YOLOv8 berbasis CNN mampu memberikan solusi yang tangguh dan efisien untuk sistem deteksi bola dan gawang secara *real-time* pada robot sepak bola beroda. Selain itu, performa model menunjukkan tingkat stabilitas yang baik terhadap perubahan pencahayaan dan posisi objek, sehingga metode ini dinilai efektif untuk diterapkan pada sistem persepsi visual robot KRSBI Beroda.

Kedua, penelitian yang berjudul "Implementasi *Object Detection* pada Robot Sepak Bola Beroda Berbasis Kamera *Omnidirectional* Menggunakan Opencv" dilakukan oleh Nanda et al. (2023). Penelitian ini mengangkat permasalahan terkait sistem deteksi pada robot KRSBI yang masih sangat dipengaruhi oleh kondisi pencahayaan. Perubahan intensitas cahaya di lapangan membuat sistem kesulitan mempertahankan akurasi deteksi, sehingga kinerja robot menjadi tidak stabil ketika kondisi cahaya berubah. Metode yang digunakan dalam penelitian ini adalah *object detection* berbasis kamera *omnidirectional* dengan bantuan pustaka OpenCV. Proses pengujian dilakukan dengan cara mengukur variasi intensitas cahaya di lapangan untuk melihat pengaruhnya terhadap kemampuan deteksi objek. Hasil yang diperoleh menunjukkan bahwa robot KRSBI mampu mengenali beberapa objek penting, seperti bola, gawang, serta robot lawan (*cyan* dan *magenta*), dengan tingkat akurasi sekitar 70%. Akan tetapi, nilai akurasi ini masih cukup dipengaruhi oleh perbedaan intensitas cahaya dari masing-masing objek. Dengan demikian, penelitian ini membuktikan bahwa penerapan *object detection* menggunakan kamera *omnidirectional* dan OpenCV dapat berjalan pada robot KRSBI, namun tetap memiliki keterbatasan signifikan terutama ketika menghadapi kondisi pencahayaan yang tidak stabil.

Ketiga, penelitian yang berjudul "Perancangan Sistem Pendeteksian Obyek Bola dengan Metode *Framework* YOLO V4" dilakukan oleh Nuralim et al. (2022). Latar belakang penelitian ini berangkat dari kebutuhan dalam KRSBI Beroda tahun 2018, di mana robot dituntut mampu melakukan navigasi serta menjalankan tugas utama, yaitu menemukan bola, menggiringnya, dan menendangnya ke arah gawang lawan. Untuk dapat melaksanakan fungsi tersebut, robot membutuhkan sistem

pendeteksian bola yang cepat, akurat, dan responsif. Metode yang digunakan dalam penelitian ini adalah *object detection* berbasis *framework* YOLOv4. Pengujian sistem dilakukan menggunakan *confusion matrix* untuk mengukur performa deteksi, termasuk dalam kondisi ketika bola sebagian terhalang oleh objek lain. Hasil pengujian menunjukkan bahwa sistem mampu mendeteksi bola meskipun terdapat penghalang dengan tingkat persentase 50%, 60%, hingga 70%. Namun, kemampuan deteksi menurun drastis ketika tingkat penghalang mencapai 80% ke atas. Pada kondisi tersebut (80%, 90%, hingga 100% penghalang), robot tidak lagi dapat mengenali keberadaan bola. Penelitian ini membuktikan bahwa YOLOv4 cukup efektif dalam mendeteksi bola pada kondisi yang dinamis dan kompleks, termasuk ketika sebagian objek tertutup. Akan tetapi, keterbatasan tetap muncul pada tingkat *occlusion* yang tinggi, yang menunjukkan perlunya optimasi atau integrasi metode tambahan untuk meningkatkan keandalan deteksi dalam skenario pertandingan nyata.

Penelitian berikutnya dilakukan oleh Sholehurrohman et al. (2023) dengan judul "Analisis Metode *Kalman Filter*, *Particle Filter* dan *Correlation Filter* Untuk Pelacakan Objek" membahas mengenai tantangan dalam pelacakan objek (*object tracking*) pada bidang *computer vision*. Penelitian ini mengimplementasikan tiga metode, yaitu *Kalman Filter*, *Particle Filter*, dan *Correlation Filter* untuk pelacakan objek pada data video lalu lintas dan video sirkuit Naskar. Hasil penelitian menunjukkan bahwa metode *Kalman Filter* memiliki akurasi tertinggi mencapai 96,89%, sedangkan metode *Correlation Filter* lebih unggul dalam aspek performa komputasi dengan rata-rata 26,69 FPS, sementara *Particle Filter* berada di bawah *Kalman Filter* dalam hal akurasi. Kesimpulan dari penelitian ini menegaskan bahwa *Kalman Filter* sangat potensial digunakan dalam pelacakan objek yang membutuhkan akurasi tinggi, sementara *Correlation Filter* lebih sesuai untuk kebutuhan aplikasi *real-time* karena efisiensi komputasinya. Dengan demikian, penelitian ini dapat dijadikan acuan penting dalam mendukung pemanfaatan *Kalman Filter* pada penelitian terkait optimasi pergerakan robot sepak bola, khususnya dalam memprediksi pergerakan bola secara akurat.

Penelitian kelima berjudul "Implementasi Algoritma SIFT (*Scale-Invariant Feature Transform*) dan Algoritma *Kalman Filter* dalam Mendeteksi Objek Bola" yang dilakukan oleh C. Saputra (2023) berfokus pada pengembangan sistem pendeteksian bola pada robot KRSBI Beroda. Latar belakang penelitian ini adalah tuntutan agar robot mampu mendeteksi, melacak, serta menggiring bola menuju gawang lawan secara efektif dalam KRSBI Beroda. Metode pendeteksian berbasis *color filtering* sebelumnya memang dinilai cukup baik dalam mengidentifikasi objek, namun masih memiliki kelemahan dalam aspek pelacakan (*tracking*). Untuk mengatasi hal tersebut, penelitian ini menggabungkan algoritma SIFT yang berfungsi membandingkan fitur citra guna memastikan objek yang terdeteksi benar-benar bola, serta algoritma *Kalman Filter* yang berperan sebagai estimator dalam memprediksi arah pergerakan bola berdasarkan status objek pada *frame* sebelumnya. Hasil dari penelitian ini menunjukkan bahwa kombinasi algoritma SIFT dan *Kalman Filter* dapat meningkatkan akurasi serta kecepatan pendeteksian bola. *Tracking* yang dilakukan dengan *Kalman Filter* mampu memprediksi pergerakan objek dengan baik, di mana koordinat y akan semakin kecil jika bola bergerak ke atas *frame* dan semakin besar jika bergerak ke bawah, sedangkan koordinat x akan semakin kecil ketika bola bergerak ke kiri dan semakin besar saat bergerak ke kanan. Dari hasil pengujian, sistem berhasil mendeteksi objek bola dengan sempurna pada jarak tertentu, dengan rata-rata *error* pengukuran *Kalman Filter* sebesar 1,06 untuk koordinat x dan 7,34 untuk koordinat y. Dengan demikian, penelitian ini menegaskan potensi integrasi SIFT dan *Kalman Filter* untuk menghasilkan sistem deteksi dan pelacakan bola yang lebih andal dalam mendukung performa robot KRSBI.

Penelitian selanjutnya yang relevan berjudul "*Detection and Recognition of Moving Video Objects: Kalman Filtering with Deep Learning*" yang dilakukan oleh Mohammed & Hussain (2021). Penelitian ini bertujuan untuk meningkatkan akurasi dalam proses deteksi dan pengenalan objek bergerak dalam urutan video. Tujuan ini diangkat karena adanya faktor penghalang seperti jarak deteksi kamera atau kekaburan (*blurring*) gambar yang dapat mengurangi akurasi teknik yang ada. Untuk mencapai akurasi yang lebih tinggi, metode yang diusulkan adalah sistem

hibrida yang menggabungkan *Kalman Filter* dengan CNN. *Kalman Filter* diterapkan pada tahap awal deteksi untuk menghilangkan latar belakang dan memotong objek, serta berfungsi sebagai estimator rekursif yang mampu memprediksi lokasi objek di masa depan, mengurangi *noise* dari deteksi yang salah, dan mengasosiasikan multi-objek ke treknya. Setelah itu, model CNN akan memprediksi kategori objek yang sudah dideteksi dan dipotong. Hasil eksperimen menunjukkan bahwa pendekatan hibrida ini berhasil mencapai akurasi pengenalan hingga 100% pada 8 video berbeda. Hasil ini menunjukkan superioritas sistem yang diusulkan dibandingkan dengan enam algoritma lain yang ada.

Penelitian selanjutnya adalah studi berjudul "*Optimized Object Tracking Technique Using Kalman Filter*" yang dilakukan oleh Taylor et al. (2016). Penelitian ini berfokus pada pengembangan teknik pelacakan objek yang efisien untuk mengatasi masalah waktu pemrosesan yang tinggi dalam pendeteksian objek di tengah adegan yang berantakan (*cluttered scene*). Metode konvensional pelacakan berbasis fitur seperti SIFT atau SURF (*Speeded Up Robust Feature*) dinilai akurat, tetapi membutuhkan waktu pemrosesan yang lebih tinggi. Sebaliknya, metode berbasis warna memiliki waktu pemrosesan yang lebih cepat, namun akurasi yang terbatas. *Kalman Filter* digunakan dalam penelitian ini adalah untuk mengatasi kompromi tersebut dan meningkatkan efisiensi komputasi, khususnya untuk aplikasi *real-time* seperti pada sistem robotik. *Kalman Filter* berperan sebagai estimator rekursif yang efisien untuk memprediksi lokasi objek di *frame* berikutnya berdasarkan status saat ini dan model gerakan objek (diasumsikan kecepatan konstan). Prediksi lokasi ini memungkinkan sistem hanya mencari objek dalam jendela gambar yang dipotong (*cropped image*) yang ukurannya jauh lebih kecil daripada keseluruhan *frame* video. Dengan demikian, waktu pemrosesan pendeteksian objek dapat diminimalkan secara signifikan. Hasil dari penelitian ini menunjukkan bahwa integrasi *Kalman Filter* dengan teknik *cropping* secara signifikan mempercepat waktu pemrosesan sambil mempertahankan tingkat akurasi yang tinggi. Waktu pemrosesan menjadi lebih cepat ketika ukuran jendela pencarian (*search window*) diperkecil. Untuk menyeimbangkan antara waktu pemrosesan yang minimal dan kesalahan jarak (*distance error*) yang minimal,

penelitian ini menyimpulkan bahwa ukuran jendela pencarian yang optimal adalah 2.16 kali dimensi terbesar objek. Pada ukuran ini, terdapat penurunan signifikan dalam waktu pemrosesan sekaligus memastikan objek terdeteksi dengan tingkat keberhasilan yang tinggi, serta pusat objek yang terdeteksi cukup dekat dengan pusat sebenarnya.

Penelitian selanjutnya oleh Egi (2022), yang berjudul "*Basketball self training shooting posture recognition and trajectory estimation using computer vision and Kalman Filter*", secara khusus berfokus pada peningkatan performa *self-shooting* bola basket, terutama untuk pemain muda, yang sering mengalami kesulitan dan keengganan akibat postur yang salah dan tembakan yang meleset tanpa bimbingan pelatih. Penelitian ini dibuat sebagai upaya untuk menyediakan sistem umpan balik otomatis yang dapat melacak gerakan pemain, mengenali postur, dan mengestimasi lintasan proyektil secara *real-time*. Penelitian ini menggunakan teknik *computer vision* seperti pemisahan saluran warna RGB, *Median Filter*, binarisasi, dan *Area Opening* untuk mendeteksi serta melabeli objek penting, yaitu bola basket dan *T-shirt* pemain. Dengan mengestimasi lintasan proyektil, penelitian ini menunjukkan bahwa lintasan tersebut dipengaruhi secara signifikan oleh ketidakpastian lingkungan, khususnya gaya hambat udara (*air drag force*). Untuk mengatasi *noisy medium* (medium bising) ini dan mengoptimalkan lintasan yang sebenarnya, algoritma *Kalman Filter* digunakan sebagai filter rekursif canggih untuk memprediksi posisi, kecepatan, dan percepatan objek bergerak serta mengurangi kesalahan prediksi secara berulang. Hasilnya, penelitian ini berhasil menentukan bahwa sudut tembakan optimal (*best shooting angle*) yang menghasilkan skor sukses paling tinggi bagi pemain dengan tinggi $1.73\ m$ dan kecepatan awal $9.5\ m/s$ adalah $50°$ ketika gaya hambat udara diperhitungkan. Perhitungan ini menunjukkan deviasi yang signifikan; tanpa hambatan udara, jangkauan maksimum horizontal (x_{max}) adalah $10.76\ m$, namun dengan hambatan udara, x_{max} turun menjadi $5.47\ m$, menegaskan peran krusial *Kalman Filter* dalam memprediksi lintasan secara akurat dalam kondisi dunia nyata.

Penelitian selanjutnya adalah "*ConfTrack: Kalman Filter-based Multi-Person Tracking by Utilizing Confidence Score of Detection Box*" yang diteliti oleh

Jung et al. (2024). Urgensi penelitian ini muncul dari kelemahan mendasar *tracker* berbasis *Kalman Filter-based Tracking-by-Detection* (KFTBD) konvensional ketika dihadapkan pada hasil deteksi yang *noisy* (kotak dengan *confidence* rendah) dalam situasi ramai. Deteksi berkepercayaan rendah ini terbukti berhubungan dengan nilai *Intersection over Union* (IoU) yang lebih rendah, yang secara langsung mengganggu kinerja pelacakan dan berpotensi menyebabkan *ID switch* atau kegagalan *track*. Untuk mengatasi masalah ini, *ConfTrack* menyarankan peningkatan fungsi *Kalman Filter* yang adaptif terhadap skor kepercayaan. Fungsi utama *Kalman Filter* dalam penelitian ini tetap sebagai estimator yang memprediksi dan memperbarui status *track*, namun ditambahkan tiga metode utama untuk menanggulangi *noise*, yaitu *Confidence Weighted Kalman-Update* (CW) untuk melakukan modifikasi pada pengukuran target *Kalman Filter*. Jika skor kepercayaan deteksi rendah, pengukuran yang digunakan akan lebih condong ke prediksi *Kalman Filter* daripada ke deteksi mentah (*noisy*). Kemudian menggunakan *Noise Scale Adaptive Kalman Filter* (NK) untuk memperkuat kovariansi *noise* ruang pengukuran yang digunakan untuk menghitung *Kalman gain* ketika skor kepercayaan rendah. Hal ini secara efektif memberikan penalti pada deteksi berkepercayaan rendah di tahap *matching*. Terakhir, menggunakan *Constant Box Prediction* (CP) untuk menstabilkan prediksi *Kalman Filter* untuk *lost track* dengan mengatur variasi ukuran kotak menjadi nol, meminimalkan dampak *noise* pada prediksi dimensi kotak. Hasil penelitian menunjukkan bahwa kombinasi metode berbasis *Kalman Filter* ini, khususnya NK, memberikan kontribusi paling konsisten dalam menaikkan metrik pelacakan. *ConfTrack* terbukti paling kuat di lingkungan ramai dan berkinerja baik dalam berbagai situasi, mencapai skor *Higher Order Tracking Accuracy* (HOTA) dan *Identification F1-Score* (IDF1) tertinggi pada *dataset Multiple Object Tracking Benchmark* 2020 (MOT20). Pencapaian IDF1 tertinggi menunjukkan bahwa *ConfTrack* sangat kuat terhadap *noise* deteksi di lingkungan ramai, memungkinkannya melacak secara stabil dengan tingkat *missing track* yang rendah dan menghasilkan *trajectory* yang stabil saat terjadi oklusi. Dengan demikian, penelitian ini menegaskan kembali

peran krusial *Kalman Filter* dalam *multi-person tracking*, khususnya melalui penyesuaian adaptif terhadap kualitas data input deteksi.

Penelitian selanjutnya dilakukan oleh Lee (2024) dalam artikel berjudul "*Confidence-Guided Frame Skipping to Enhance Object Tracking Speed*". Urgensi penelitian ini didasari oleh dilema antara kecepatan pemrosesan dan akurasi pelacakan pada perangkat dengan sumber daya komputasi terbatas, di mana algoritma pelacakan yang kuat cenderung memonopoli sumber daya. Untuk mengatasi hal tersebut, Lee mengusulkan pendekatan dua tingkat (*two-tiered approach*) yang mengintegrasikan metode pelacakan ringan (*lightweight*) berbasis *block-matching* dengan algoritma pelacakan yang kompleks dan kuat (*robust*). Inti dari metode ini adalah penggunaan skor kepercayaan (*confidence level*, C_L) yang dihitung berdasarkan nilai *Sum of Absolute Differences* (SAD) dan gradien tekstur objek untuk menentukan kebutuhan intervensi algoritma berat secara adaptif. Jika C_L berada di atas ambang batas, sistem hanya menjalankan pelacakan ringan, namun jika C_L rendah atau telah mencapai batas maksimal frame yang dilewati (S_N), algoritma berat akan dipanggil untuk memastikan reliabilitas pelacakan. Hasil eksperimen menunjukkan bahwa metode ini mampu meningkatkan kecepatan pemrosesan hingga 1,5 kali lipat dengan degradasi akurasi yang sangat minimal dibandingkan menjalankan algoritma berat di setiap *frame*. Penelitian ini menegaskan bahwa strategi *frame skipping* yang dipandu oleh evaluasi kepercayaan sangat efektif untuk mengoptimalkan penggunaan sumber daya komputasi tanpa mengorbankan kualitas pelacakan secara signifikan.

Penelitian yang terakhir adalah "*Automatic pear and apple detection by videos using deep learning and a Kalman Filter*" oleh Takura et al. (2021). Penelitian ini mengatasi tantangan krusial dalam otomatisasi pertanian, yaitu estimasi jumlah buah di kebun secara akurat. Proses penghitungan buah dibagi menjadi dua langkah penting: deteksi buah dan pelacakan buah (*tracking*) melalui urutan *frame* gambar. Urgensi pelacakan muncul karena kondisi lapangan yang tidak stabil, seperti perubahan pencahayaan di bawah kanopi pohon, serta kemungkinan buah tertutup oleh daun atau ranting (*occlusion*), yang dapat menyebabkan kegagalan deteksi di beberapa *frame*. Tanpa pelacakan yang andal,

buah yang muncul kembali setelah tersembunyi dapat dihitung ganda (*double-counted*), sehingga hasil penghitungan menjadi tidak akurat. Untuk mengatasi masalah ini, metode deteksi objek berbasis *deep learning* YOLO v2 digunakan untuk mengidentifikasi buah di setiap *frame*. Kemudian, *Kalman Filter* diterapkan untuk pelacakan objek. Fungsi utama *Kalman Filter* adalah sebagai estimator yang memprediksi dan mengoreksi status (posisi dan kecepatan) buah yang sedang dilacak, bahkan ketika deteksi buah gagal di *frame* tersebut. Ini memastikan bahwa buah yang sama dapat terus diidentifikasi di *frame* yang berurutan, meskipun sempat tidak terdeteksi. Hasilnya menegaskan efektivitas integrasi ini: sistem berhasil menghitung buah pir dan apel secara otomatis dengan kesalahan absolut kurang dari 10%. Secara spesifik, dari total 234 buah pir, 226 buah berhasil dihitung dengan benar, menghasilkan nilai F1 sebesar 0.972. Performa *Kalman Filter* juga terbukti jauh lebih unggul dibanding algoritma pelacakan fitur Kanade-Lucas-Tomasi (KLT), yang menyebabkan penghitungan berlebih (*over-counted*). Dengan demikian, penelitian ini menunjukkan bahwa *Kalman Filter* sangat efektif dan esensial dalam menstabilkan proses penghitungan objek bergerak pada video di lingkungan yang tidak terkontrol.

Tabel 2.1 Penelitian Terdahulu

No	Peneliti	Tahun	Judul	Masalah	Metode	Hasil	Perbedaan
1	T. Mohd. Farhan, Feri Chandra	2024	*CNN-Based Ball and Goal Detection for KRSBI Robot with Omnidirectional Camera*	Metode deteksi objek penyaringan warna HSV memiliki kinerja yang buruk di bawah variasi kondisi pencahayaan pada robot KRSBI-Beroda	Menggunakan CNN berbasis algoritma YOLOv8 dengan kamera *omnidirectional*. Model dilatih menggunakan 1.125 gambar selama 100 epochs.	Model mencapai Akurasi 95,87%, Presisi 1.00 (pada confidence 0.921) 8, Recall 0.99, dan F1-Score 0.97 (pada confidence 0.149).	Penelitian ini fokus meningkatkan kinerja robot dari menggunakan HSV ke YOLO, sedangkan penelitian sekarang menambahkan optimasi gerakan dengan *Kalman Filter*.
2	Bagas Musamma Nanda, Simon Siregar, dan Muhammad Ikhsan Sani	2023	Implementasi *Object Detection* pada Robot Sepak Bola Beroda Berbasis Kamera *Omnidirectional* Menggunakan Opencv	Deteksi objek yang bergantung pada intensitas cahaya, sehingga membuat hasil deteksi menjadi kurang efektif	OpenCV *Object Detection*	Robot bisa deteksi bola, gawang, robot *cyan* & magenta dengan akurasi ±70%, tapi hasil dipengaruhi intensitas cahaya.	Penelitian ini hanya menggunakan OpenCV sederhana, sedangkan penelitian sekarang memakai YOLO + *Kalman Filter*.

No	Peneliti	Tahun	Judul	Masalah	Metode	Hasil	Perbedaan
3	Jalu Nuralim, Nifty Fath, Akhmad Musafa, Sujono, Drs. Suwasti Broto	2022	Perancangan Sistem Pendeteksian Obyek Bola dengan Metode *Framework* YOLO V4	Robot butuh deteksi cepat untuk navigasi & menendang bola, namun terhambat jika bola terhalang sebagian.	YOLOv4 + *Confusion Matrix*	Robot masih bisa deteksi bola hingga 70% tertutup, gagal pada >80%.	Fokus pada akurasi YOLOv4 dengan halangan, sedangkan penelitian sekarang menambahkan prediksi pergerakan menggunakan *Kalman Filter*.
4	Ridho Sholehurrohman, Mochammad Reza Habibi, Igit Sabda Ilman, Rahman Taufiq, Muhaqiqin	2023	Analisis Metode *Kalman Filter*, *Particle Filter* dan *Correlation Filter* Untuk Pelacakan Objek	*Tracking* objek sering gagal karena oklusi dan deformasi target.	*Kalman Filter*, *Particle Filter*, *Correlation Filter*	*Kalman Filter* akurasi 96,89%, *Correlation Filter* paling cepat 26,69 FPS.	Penelitian ini membandingkan metode tracking umum, penelitian sekarang fokus *Kalman Filter* untuk robot sepak bola berbasis YOLO.
5	M. Irwan Bustami, Chindra Saputra, Desi Kisbianty, Arjuna Panji Prakarsa	2023	Implementasi Algoritma SIFT (*Scale-Invariant Feature Transform*) dan *Kalman Filter* dalam	Metode *color filtering* saja kurang handal untuk *tracking* bola.	SIFT + *Kalman Filter*	SIFT mengenali objek, *Kalman Filter* memprediksi arah bola. *Error* rata-rata 1.06 (x) dan 7.34 (y).	Penelitian ini gabungkan SIFT + *Kalman Filter*, sedangkan penelitian sekarang pakai YOLO untuk deteksi dan *Kalman Filter* untuk prediksi.

No	Peneliti	Tahun	Judul	Masalah	Metode	Hasil	Perbedaan
			Mendeteksi Objek Bola				
6	Hind Rustum Mohammed & Zahir M. Hussain	2021	*Detection and recognition of moving video objects: Kalman Filtering with deep learning*	Kebutuhan akan akurasi yang lebih tinggi dalam deteksi dan pengenalan objek bergerak, terutama menghadapi kendala seperti jarak kamera atau *blurring*.	*Kalman Filter* + CNN	Akurasi pengenalan mencapai 100% pada 8 video berbeda. Sistem menunjukkan superioritas dibandingkan enam algoritma yang ada.	*Kalman Filter* digunakan sebagai tahap awal untuk deteksi dan *tracking*. Metode deteksi awal tidak menggunakan YOLO, melainkan *Temporal Median* dan *thresholding* untuk menghilangkan *background*.
7	Liana Ellen Taylor, Midriem Mirdanies, Roni Permana Saputra	2016	*Optimized Object Tracking Technique Using Kalman Filter*	Waktu pemrosesan yang tinggi pada deteksi objek, terutama untuk aplikasi *real-time*, sambil mempertahankan akurasi dalam adegan yang	*Kalman Filter* + *Color Segmentation* + *Cropped Image*	Proses pelacakan menjadi lebih cepat secara signifikan (waktu pemrosesan berkurang) dengan mempertahankan akurasi tinggi. Ukuran jendela pencarian optimal ditemukan pada	Penelitian ini menggunakan Segmentasi Warna dan *Kalman Filter* untuk mengoptimalkan waktu pemrosesan melalui teknik *Cropped Image*. Sedangkan penelitian sekarang menggunakan YOLO (model *deep*

No	Peneliti	Tahun	Judul	Masalah	Metode	Hasil	Perbedaan
				berantakan (*cluttered scene*).		2.16 kali dimensi terbesar objek, menyeimbangkan waktu pemrosesan dan kesalahan jarak.	*learning* yang kompleks) sebagai detektor, dan *Kalman Filter* digunakan untuk stabilisasi *tracking* dan mengatasi oklusi/*false negative* akibat beban komputasi YOLO yang tinggi.
8	Yunus Egi	2022	*Basketball self training shooting posture recognition and trajectory estimation using computer vision and Kalman Filter*	Kurangnya umpan balik dari pelatih membuat *self-shooting* menyulitkan dan memakan waktu, di mana postur salah menyebabkan tembakan meleset dan menimbulkan keengganan. Diperlukan algoritma untuk	RGB *Channelization, Median Filter*, Binarisasi, *Area Opening* + *Kalman Filter*	Sistem berhasil mengklasifikasikan objek dan menentukan sudut postur terbaik. *Kalman Filter* terbukti penting karena tanpa hambatan udara (x_{max} 10.76 m), hasil berbeda signifikan dari kondisi nyata dengan hambatan udara (x_{max} 5.47	Penelitian ini berfokus pada analisis postur manusia dan estimasi lintasan proyektil fisik (bola basket) dalam domain olahraga, menggunakan CV tradisional. *Kalman Filter* berfungsi untuk mengoreksi *noise* fisik (gaya hambat udara) pada lintasan. Penelitian sekarang berfokus pada optimasi gerakan robot dalam domain Robot

No	Peneliti	Tahun	Judul	Masalah	Metode	Hasil	Perbedaan
				memberikan *feedback* postur dan lintasan *real-time*.		m), sehingga filter berhasil mengoreksi prediksi lintasan fisik. Sudut terbaik yang terdeteksi 51°.	Sepak Bola, menggunakan CV berbasis YOLO. *Kalman Filter* berfungsi untuk menstabilkan *noise* deteksi objek (oklusi/ *false negative*) dan mereduksi beban komputasi.
9	Hyeonchul Jung, Seokjun Kang, Takgen Kim, HyeongKi Kim	2024	*ConfTrack*: *Kalman Filter-based Multi-Person Tracking by Utilizing Confidence Score of Detection Box*	Kegagalan *tracking* dan *ID switch* pada *tracker Kalman Filter* konvensional yang disebabkan oleh hasil deteksi berkepercayaan rendah di lingkungan yang ramai	*Kalman Filter*, *Confidence Weighted Kalman-Update* (CW), *Noise Scale Adaptive Kalman Filter* (NK)	*ConfTrack* mencapai skor HOTA dan IDF1 tertinggi pada *dataset* MOT20, membuktikan *robustness* yang tinggi terhadap *noise* deteksi di lingkungan yang padat.	*ConfTrack* memodifikasi *Kalman Filter* untuk mempenalti/menghukum deteksi berkepercayaan rendah guna stabilitas ID. Penelitian sekarang menggunakan *Kalman Filter* untuk memprediksi pergerakan bola secara stabil, mengoptimasi gerakan robot, dan mengurangi beban komputasi pada YOLO.

No	Peneliti	Tahun	Judul	Masalah	Metode	Hasil	Perbedaan
10	Yun Gu Lee	2024	*Confidence-Guided Frame Skipping to Enhance Object Tracking Speed*	Konsumsi daya komputasi yang tinggi dari algoritma pelacakan berat pada perangkat terbatas.	Pendekatan dua tingkat menggunakan *Block-Matching* (ringan) dan pelacak *Robust* yang dipandu skor kepercayaan (C_L) .	Kecepatan meningkat hingga 1.5x dengan akurasi yang hampir identik dengan metode dasar.	Menggunakan *block-matching* untuk *frame skipping*, sementara penelitian ini menggunakan YOLOv8 dan *Kalman Filter* dengan bantuan *frame skipping* untuk robot KRSBI.
11	Kenta Itakura, Yuma Narita, Shuhei Noaki, & Fumiki Hosoi	2021	*Automatic pear and apple detection by videos using deep learning and a Kalman Filter*	Penghitungan buah di kebun dari video sulit karena perubahan kondisi cahaya yang cepat dan tidak stabil, buah dapat tertutup daun/ranting (*occlusion*).	YOLOv2 + *Kalman Filter*.	Rata-rata presisi (*Average Precision*) deteksi pir dan apel tinggi (0.97). Penghitungan pir benar: 226 dari 234 (F1 = 0.972). *Kalman Filter* mampu melacak buah yang sempat tidak terdeteksi (*occluded*).	Penelitian ini fokus pada objek statis (buah) di lingkungan pertanian yang tidak terkontrol (pencahayaan, oklusi, pergerakan kamera *handheld*), sedangkan penelitian sekarang fokus pada objek dinamis (bola) yang bergerak cepat oleh robot di lingkungan yang lebih terkontrol.

Secara keseluruhan, tinjauan penelitian terdahulu menunjukkan adanya konsensus bahwa algoritma *deep learning* seperti YOLO unggul dalam akurasi deteksi objek, tetapi memiliki kerentanan terhadap isu *real-time* dan penggunaan sumber daya yang intensif, yang sangat krusial dalam sistem robotika dinamis. Penelitian sebelumnya telah membuktikan bahwa integrasi antara *tracker* berbasis filter dengan detektor berbasis *vision* mampu meningkatkan performa pelacakan secara signifikan. Secara spesifik, *Kalman Filter* menjadi pilihan optimal karena sifatnya yang ringan secara komputasi dan efektif dalam mereduksi *noise* serta memprediksi posisi objek saat terjadi oklusi.

Namun, menjalankan algoritma deteksi yang kompleks di setiap *frame* dianggap tidak efisien karena tidak semua *frame* memerlukan kalkulasi ulang yang berat. Mengadopsi strategi *frame skipping* yang diusulkan oleh Lee (2024), penelitian ini mengusulkan optimasi melalui pembagian beban kerja komputasi. Implementasi ini dilakukan dengan menjalankan YOLOv8 sebagai detektor pada *frame* awal, kemudian memanfaatkan *Kalman Filter* untuk memprediksi posisi objek secara mandiri pada tiga *frame* berikutnya ($S_L = 3$) sebelum memanggil kembali detektor untuk sinkronisasi ulang.

Perbedaan mendasar yang menjadi celah penelitian ini terletak pada implementasi yang mengintegrasikan secara spesifik YOLOv8, *Kalman Filter*, dan skema *frame skipping* adaptif untuk optimasi gerakan robot pada KRSBI Beroda UNRI. Dengan demikian, penelitian ini bertujuan untuk menjembatani celah tersebut dengan menguji secara kuantitatif efektivitas integrasi metode tersebut guna menghasilkan sistem kontrol gerak robot yang lebih stabil, efisien secara daya komputasi, dan kompetitif dalam situasi pertandingan.

2.2 Kontes Robot Sepak Bola Beroda Indonesia

Salah satu ajang kompetisi robotik di Indonesia adalah Kontes Robot Indonesia atau lebih sering disebut dengan KRI. KRI diselenggarakan oleh Pusat Prestasi Nasional (PUSPRESNAS) dan Kementrian Pendidikan dan Kebudayaan Republik Indonesia. KRI merupakan acara yang diadakan setiap tahun dan diikuti oleh mahasiswa dari berbagai wilayah di Indonesia mulai dari Timur, Tengah dan

Barat. KRI terbagi menjadi 6 kategori, yang salah satunya adalah Kontes Robot Sepak Bola Indonesia (KRSBI) Beroda (Nanda et al., 2023).

Kontes Robot Sepakbola Beroda Indonesia diadakan untuk meningkatkan keilmuan dan kreatifitas mahasiswa di bidang robotika. Di dalam kontes ini, mahasiswa dituntut untuk bisa mengembangkan kemampuan dalam mekanika, manufaktur, elektronika, pemrograman, AI, *image processing*, komunikasi digital, dan strategi, sekaligus diperlukan pengembangan ke arah disiplin, toleransi, sportifitas, kerjasama, saling menghargai, kontrol emosi dan kemampuan *softskill* lainnya (Kusumoputro et al., 2024).

Gambar 2.1 Ilustrasi Pertandingan KRSBI Beroda

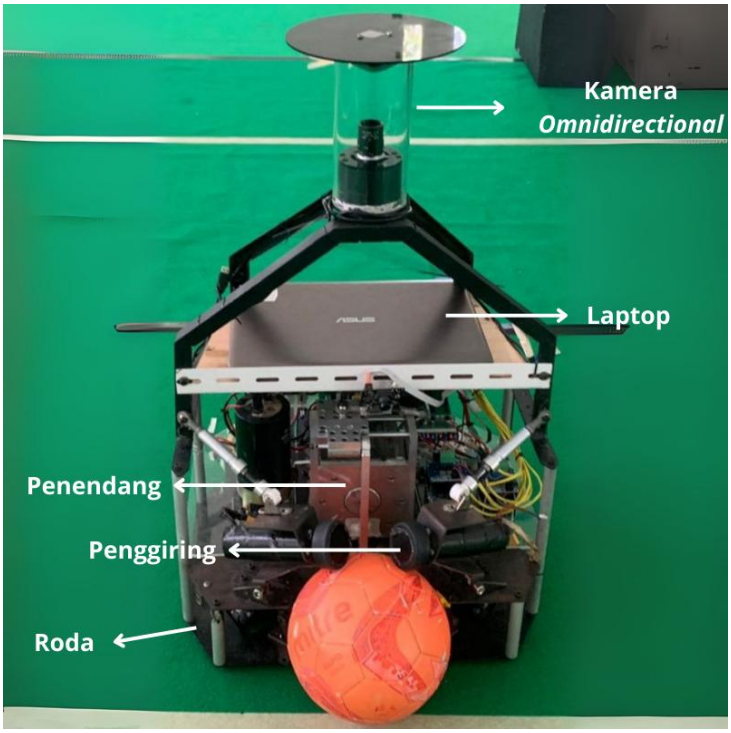
Sumber : Rochardjo, 2024

2.3 Robot KRSBI Beroda

Dalam KRSBI Beroda, robot yang digunakan pada tahap ini merupakan robot yang sama seperti yang dipakai pada pertandingan tingkat nasional, dengan beberapa ketentuan khusus. Jumlah robot yang diizinkan adalah dua unit, yaitu Robot 1 (R1) dan Robot 2 (R2), keduanya bertipe robot penyerang. Adapun spesifikasi fisik robot diatur dengan ukuran proyeksi ke lantai minimal 30 cm × 30 cm dan maksimal 52 cm × 52 cm, dengan tinggi robot antara 40 cm hingga 80 cm. Jika tinggi robot melebihi 60 cm, maka bagian tubuh robot di atas ketinggian tersebut harus berada dalam silinder berdiameter 25 cm. Berat maksimum robot

ditetapkan 40 kg, sedangkan bentuk robot dibuat bebas selama sesuai regulasi, dan warna yang digunakan adalah hitam (Kusumoputro et al., 2024).

Robot yang digunakan dalam Kontes Robot Sepak Bola Indonesia (KRSBI) Beroda memiliki bentuk khas menyerupai kubus dengan berbagai komponen penting yang berfungsi mendukung pergerakan serta kemampuan bermain sepak bola. Gambar 2.1 memperlihatkan salah satu contoh robot KRSBI Beroda yang dilengkapi dengan berbagai sistem pendukung.



Gambar 2.2 Robot KRSBI Beroda

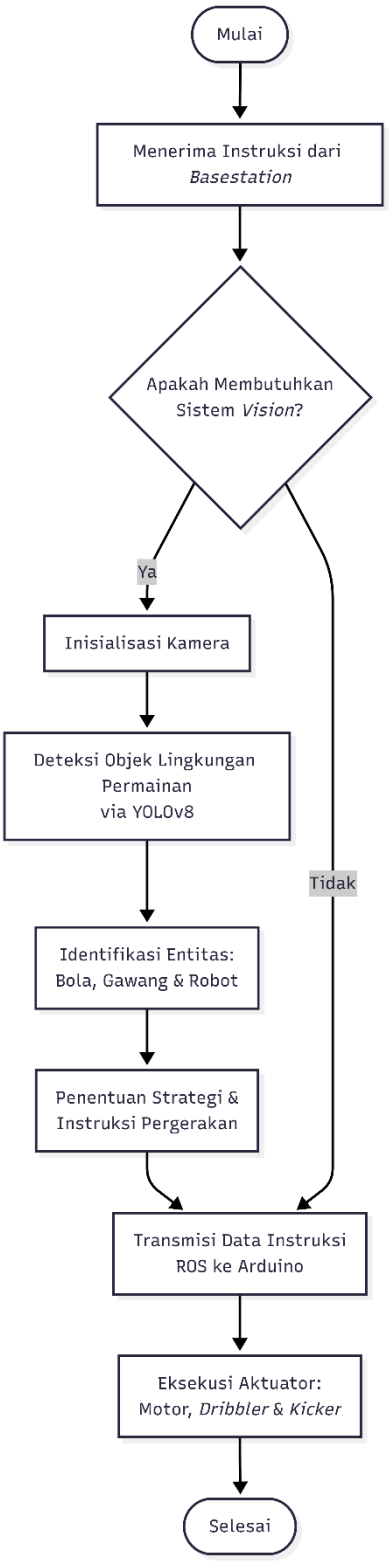
Kamera *omnidirectional* ditempatkan di bagian atas robot dan berfungsi untuk mendeteksi objek seperti bola, gawang, maupun robot lawan. Konsep kamera *omnidirectional* adalah menangkap citra dari segala arah (depan, belakang, kiri, dan kanan) menggunakan teknik pantulan cermin cembung yang diarahkan ke bawah (Surya et al., 2025). Dengan cara ini, kamera mampu memperoleh citra lapangan secara menyeluruh hanya dari satu titik pengamatan. Robot dilengkapi dengan penendang berbasis solenoid. Solenoid adalah sebuah komponen elektromagnetik yang berfungsi mengubah energi listrik menjadi energi mekanis. Energi mekanis

yang dihasilkan dapat berupa gerakan mendorong (*push*) maupun menarik (*pull*) (Shofi et al., 2023).

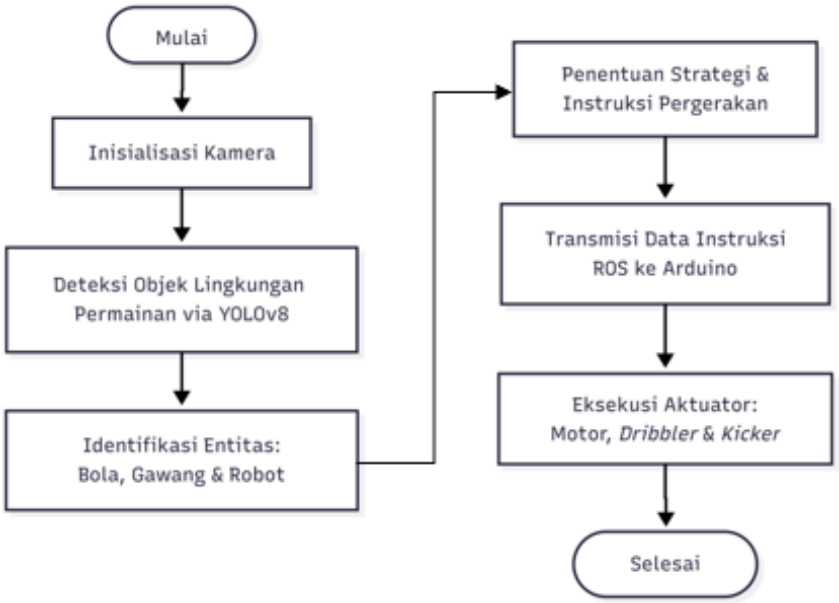
Sistem navigasi pada robot sepak bola ini dirancang menyerupai koordinasi tubuh manusia, di mana terdapat pembagian tugas antara pusat berpikir (otak) dan pusat penggerak (otot). Secara fisik, robot menggunakan laptop ASUS K401U dibantu oleh ROS (*Robot Operating System*) dan Python sebagai "otak utama" yang menangani tugas-tugas berat seperti pengolahan gambar, sementara mikrokontroler Arduino Mega 2560 bertugas sebagai "saraf motorik" yang mengontrol pergerakan roda secara langsung. Komunikasi antar keduanya dijembatani oleh protokol *rosserial*, sedangkan interaksi dengan pengguna di luar lapangan menggunakan *Basestation* dilakukan secara nirkabel melalui sistem `Socket.IO`.

Untuk mendukung pergerakan yang lincah, robot dilengkapi dengan tiga motor penggerak yang disusun secara khusus (*omnidirectional*). Konfigurasi ini memungkinkan robot untuk bergerak ke segala arah secara instan tanpa harus mengubah posisi hadapnya terlebih dahulu. Pada bagian depan robot terdapat roda kecil yang dipasangi motor penggerak dan berputar ke arah dalam tubuh robot yang disebut *dribbler*. *Dribbler* ini berfungsi untuk menangkap dan menahan bola ketika robot bergerak, sehingga bola tetap berada di depan robot dan tidak mudah terlepas.

Alur kerja robot dimulai ketika robot menerima instruksi atau perintah dari *Basestation*. Perintah ini berasal dari *Referee Box* atau *user* yang mengirim perintah spesifik kepada robot. Jika perintah yang dikirimkan berupa gerakan langsung oleh robot, misal bergerak maju atau menghidupkan *dribbler*, maka perintah akan langsung dikirim dari ROS ke Arduino. Namun, jika perintah yang dikirimkan berhubungan dengan *vision*, maka ROS akan memulai kamera menangkap gambaran lapangan di sekitarnya. Data visual ini kemudian diproses oleh sistem deteksi YOLOv8 untuk mengenali objek-objek penting seperti bola dan gawang. Setelah objek ditemukan, algoritma navigasi akan menentukan langkah apa yang harus diambil, seperti instruksi untuk "maju", "kiri", atau "berhenti". Instruksi tersebut kemudian dikirimkan ke Arduino untuk diubah menjadi tenaga listrik yang menggerakkan motor.

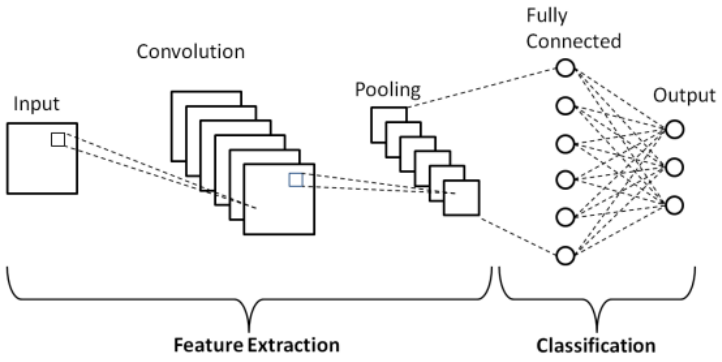


Gambar 2.3 Diagram Alur Kerja Robot KRSBI Beroda



Gambar 2.4 Diagram Sistem *Vision* Robot KRSBI Beroda

2.4 CNN (*Convolutional Neural Network*)



Gambar 2.5 Arsitektur CNN

Sumber : analyticsvidhya.com, 2022

CNN (*Convolutional Neural Network*) adalah salah satu jenis jaringan saraf tiruan yang dirancang khusus untuk mengolah data dengan struktur topologi berbentuk *grid*. Contohnya adalah data deret waktu yang dapat direpresentasikan sebagai *grid* satu dimensi, serta data citra yang tersusun dalam *grid* piksel dua dimensi. Istilah *convolutional* pada CNN merujuk pada penggunaan operasi matematika konvolusi di dalam arsitekturnya. Operasi konvolusi ini merupakan bentuk khusus dari operasi linier yang digunakan sebagai pengganti perkalian matriks konvensional pada satu atau lebih lapisan jaringan (Wakhidah et al., 2023).

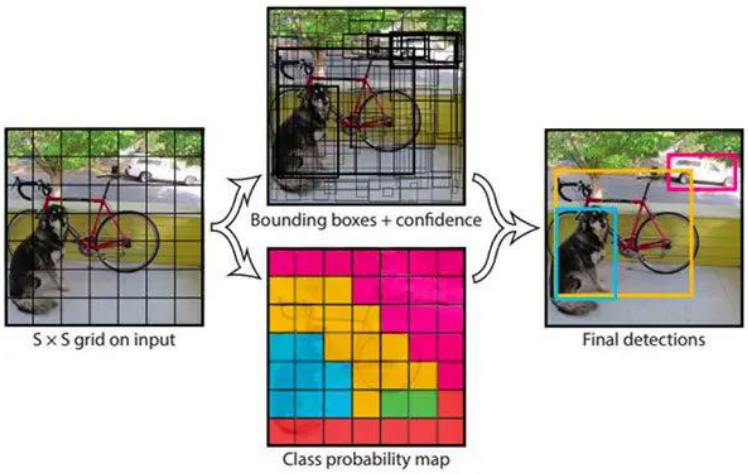
Pada CNN, setiap neuron direpresentasikan dalam bentuk dua dimensi sehingga metode ini sangat sesuai untuk pengolahan data citra. Arsitektur CNN umumnya terdiri atas beberapa tahap utama, yaitu lapisan *input*, proses ekstraksi fitur, tahap klasifikasi, dan lapisan *output*. Tahap ekstraksi fitur tersusun dari sejumlah lapisan tersembunyi yang meliputi lapisan konvolusi, fungsi aktivasi ReLU, serta lapisan *pooling*. Proses kerja CNN bersifat hierarkis, di mana keluaran dari lapisan konvolusi awal akan menjadi masukan bagi lapisan konvolusi berikutnya untuk mengekstraksi fitur yang semakin kompleks. Selanjutnya, pada tahap klasifikasi digunakan lapisan *fully connected* yang dipadukan dengan fungsi aktivasi *softmax* untuk menghasilkan keluaran berupa kelas atau label objek yang dikenali (Romario & Kadarina, 2020).

2.5 YOLO (*You Only Look Once*)

YOLO (*You Only Look Once*) merupakan salah satu model deteksi objek yang dikembangkan berdasarkan arsitektur *Convolutional Neural Network* (CNN). Model ini menerapkan pendekatan deteksi objek modern yang memungkinkan proses pendeteksian dilakukan secara *real-time*. Pada proses pengolahan citra, YOLO memiliki alur kerja yang relatif sederhana, di mana citra masukan terlebih dahulu diubah ukurannya agar sesuai dengan kebutuhan model. Selanjutnya, citra tersebut diproses menggunakan satu jaringan konvolusional secara menyeluruh untuk memprediksi lokasi dan kelas objek. Hasil deteksi kemudian diseleksi menggunakan nilai ambang batas (*threshold*) berdasarkan tingkat kepercayaan (*confidence score*) yang dihasilkan oleh model (Redmon et al., 2016).

Pada tahap awal algoritma YOLO, citra masukan dibagi ke dalam sejumlah *grid* berukuran $S \times S$. Setiap sel *grid* bertugas untuk memprediksi sejumlah *bounding box* B, beserta *confidence score*. *Confidence score* menunjukkan tingkat keyakinan model terhadap keberadaan objek di dalam *bounding box* yang diprediksi. Setiap *bounding box* menghasilkan lima parameter, yaitu koordinat pusat objek (x, y), lebar (w), tinggi (h), serta *confidence score* C. Koordinat x dan y merepresentasikan posisi pusat *bounding box* relatif terhadap sel *grid*, sedangkan w dan h menyatakan ukuran *bounding box* terhadap dimensi citra masukan. Seluruh

hasil prediksi YOLO secara umum direpresentasikan dalam bentuk tensor berdimensi $[S, S, B \times 5 + C]$ (Redmon et al., 2016).



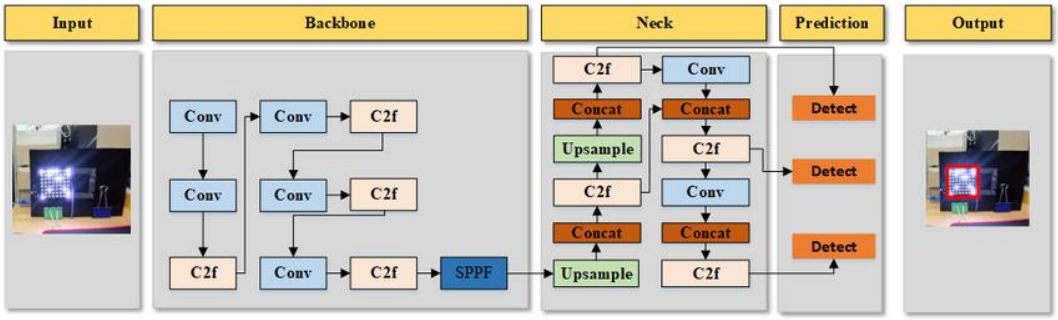
Gambar 2.6 Ilustrasi Algoritma YOLO

Sumber : Redmon et al., 2016

YOLOv8 adalah model yang akan digunakan penelitian ini. Arsitektur model ini dibangun sebagai sebuah jaringan saraf tunggal yang terdiferensiasi secara *end-to-end*, yang secara garis besar terdiri dari tiga komponen utama: *Backbone*, *Neck*, dan *Head*. Pada bagian *Backbone*, YOLOv8 menggunakan struktur CNN canggih yang mengintegrasikan modul *Cross Stage Partial* (CSP) *bottleneck* untuk mengekstraksi fitur multi-skala dari citra masukan sekaligus mengurangi redundansi komputasi dan meningkatkan penggunaan kembali fitur.

Selanjutnya, bagian *Neck* pada YOLOv8 berfungsi untuk menyempurnakan dan menggabungkan fitur-fitur multi-skala yang telah diekstraksi melalui optimasi *Path Aggregation Network* (PANet) dan *Feature Pyramid Network* (FPN). Integrasi fitur ini sangat krusial untuk meningkatkan aliran informasi antar level fitur, sehingga model mampu mendeteksi objek dengan berbagai ukuran dan konteks yang berbeda secara lebih efektif. Inovasi paling signifikan pada YOLOv8 terletak pada bagian *Head*, di mana model ini beralih dari metode berbasis *anchor* ke pendekatan *anchor-free* dalam prediksi *bounding box*. Pendekatan ini menyederhanakan proses prediksi dengan menghilangkan kebutuhan akan kotak acuan (*anchor boxes*) yang ditentukan sebelumnya, sehingga meningkatkan

fleksibilitas model terhadap objek dengan rasio aspek yang bervariasi serta mengurangi jumlah hiperparameter yang harus dikelola.

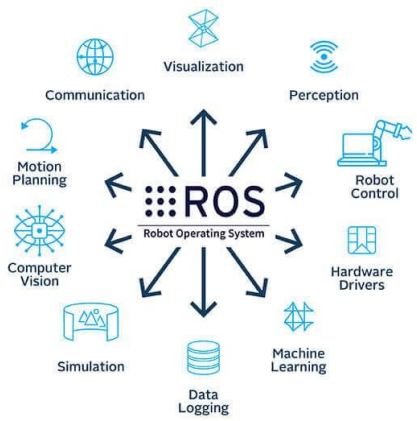


Gambar 2.7 Arsitektur YOLOv8

Sumber : Yaseen, 2024

2.6 ROS (*Robot Operating System*)

ROS (*Robot Operating System*) merupakan sebuah sistem operasi robot yang bersifat *open-source*. Namun, penting untuk dipahami bahwa ROS bukanlah sistem operasi dalam pengertian tradisional yang menangani manajemen proses dan penjadwalan seperti Windows atau Linux. Sebaliknya, ROS berfungsi sebagai lapisan komunikasi terstruktur (*structured communications layer*) yang berjalan di atas sistem operasi *host*. ROS menyediakan kerangka kerja yang fleksibel untuk penulisan perangkat lunak robot, yang mencakup berbagai *tools* dan pustaka (*libraries*) yang bertujuan untuk menyederhanakan tugas pembuatan perilaku robot yang kompleks dan kuat di berbagai platform robotik (Quigley et al., 2009).



Gambar 2.8 ROS (*Robot Operating System*)

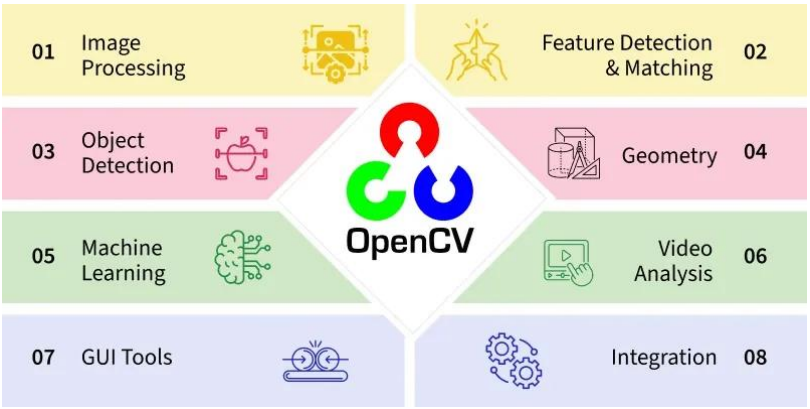
Sumber : Louda et al., 2023

Pengembangan ROS dilatarbelakangi oleh tantangan besar dalam penulisan perangkat lunak untuk robot, di mana skala dan cakupan robotika terus berkembang pesat. Berbagai jenis robot memiliki perangkat keras yang sangat bervariasi, membuat penggunaan ulang kode (*code reuse*) menjadi hal yang tidak trivial. Selain itu, besarnya ukuran kode yang diperlukan sering kali melampaui kemampuan peneliti tunggal, sehingga diperlukan arsitektur yang mendukung upaya integrasi perangkat lunak berskala besar. ROS dirancang untuk menjawab tantangan spesifik yang ditemui saat mengembangkan robot layanan berskala besar sebagai bagian dari proyek STAIR (*STanford AI Robot*) di Stanford University dan Program Robot Personal di Willow Garage. Meskipun bermula dari domain robot layanan dan manipulasi seluler, arsitektur yang dihasilkan bersifat jauh lebih umum dan dapat diterapkan pada berbagai sistem robotik lainnya (Quigley et al., 2009).

Secara teknis, implementasi ROS berpusat pada konsep *nodes*, *messages*, *topics*, dan *services*. *Node* adalah proses yang melakukan komputasi. Komunikasi antar *node* terjadi dengan mengirimkan *messages* (struktur data yang diketik secara ketat). Pengiriman pesan ini umumnya menggunakan mekanisme *publish-subscribe* melalui *topic*, di mana sebuah *node* mempublikasikan pesan ke topik tertentu dan *node* lain yang berlangganan topik tersebut akan menerimanya. Selain itu, untuk transaksi sinkron, ROS menyediakan mekanisme *service* yang berbasis permintaan dan respons (*request-response*) (Quigley et al., 2009).

2.7 OpenCV v3.4 (*Open Source Computer Vision version* 3.4)

OpenCV (*Open Source Computer Vision*) adalah sebuah pustaka perangkat lunak yang dirancang khusus untuk pengolahan citra secara *real-time*. Pustaka ini awalnya dikembangkan oleh Intel dan kini didukung oleh Willow Garage serta Itseez. OpenCV menyediakan antarmuka yang kompatibel dengan berbagai bahasa pemrograman seperti C++, C, Python, dan Java, serta dapat dijalankan pada beragam sistem operasi, di antaranya Windows, Linux, Mac OS, iOS, dan Android. Pustaka ini dirancang untuk efisiensi komputasi dengan fokus utama pada aplikasi berbasis *real-time* (C. Saputra, 2023).



Gambar 2.9 Kegunaan OpenCV

Sumber : geeksforgeeks.org, 2025

Modul pustaka OpenCV dirancang dengan tingkat fleksibilitas dan ketangguhan yang tinggi untuk menangani berbagai permasalahan dalam bidang *computer vision*. Berbagai solusi telah tersedia di dalamnya, seperti pemotongan citra (*cropping*), peningkatan kualitas citra melalui penyesuaian kecerahan, ketajaman, dan kontras, pendeteksian bentuk, segmentasi citra, pelacakan objek bergerak, hingga pengenalan objek, serta beragam fungsi lainnya (Ratna, 2020).

2.8 NumPy v2.3 (*Numerical Python version* 2.3)

Gambar 2.10 Logo NumPy

Sumber : python.plainenglish.io, 2025

NumPy (*Numerical Python*) adalah paket dasar dan fundamental untuk komputasi ilmiah dengan Python. Sebagai pustaka *open-source*, NumPy menyediakan objek *array* N-dimensi (*ndarray*) yang sangat kuat, yang berfungsi sebagai struktur data universal untuk pertukaran data multi-dimensi dalam ekosistem ilmiah Python. Objek *ndarray* ini memungkinkan operasi yang cepat dan efisien pada data homogen (semua elemen harus bertipe data yang sama) dalam jumlah besar, sebuah keunggulan signifikan dibandingkan list bawaan Python, yang

dicapai melalui inti kode yang dioptimalkan dalam bahasa C. Selain struktur *array* multidimensi, NumPy juga menawarkan koleksi lengkap fungsi matematika tingkat tinggi, termasuk aljabar *linear*, transformasi Fourier, dan kemampuan angka acak, menjadikannya standar *de-facto* untuk komputasi *array* dan menjadi fondasi bagi pustaka ilmu data dan *machine learning* lainnya (NumPy, 2025).

2.9 *Kalman Filter*

Kalman Filter merupakan algoritma yang sangat kuat untuk memperkirakan keadaan suatu sistem dinamis berdasarkan serangkaian pengukuran yang tidak lengkap dan mengandung *noise*. Algoritma ini dikembangkan oleh Rudolf E. Kálmán, dan awalnya digunakan dalam bidang navigasi penerbangan. Namun, seiring waktu, *Kalman Filter* menjadi komponen penting di berbagai bidang, seperti robotika, ekonomi, dan terutama *computer vision* (CV). *Kalman Filter* bekerja melalui dua tahap utama yang berlangsung secara berulang, yaitu tahap prediksi dan tahap pembaruan (koreksi). *Kalman Filter* mampu menghasilkan estimasi keadaan objek, seperti posisi dan kecepatannya, secara halus, akurat, dan stabil, meskipun data sensor yang diterima tidak sempurna (Ultralytics, 2025).

Kalman Filter dinamai dari penciptanya, Rudolf Kalman. Algoritma ini bekerja dengan menerima sejumlah data yang mengandung *noise*, kemudian menyaringnya untuk meminimalkan gangguan tersebut. Karena dirancang pada ruang *linear*, *Kalman Filter* juga dikenal dengan istilah *linear quadratic estimation* (Sholehurrohman et al., 2023). Metode *Kalman Filter* menggunakan informasi dari objek yang terdeteksi di suatu *frame* dan status objek dari *frame* sebelumnya untuk mendapatkan status yang baru dari objek tersebut (C. Saputra, 2023).

Dalam konteks AI, *Kalman Filter* banyak digunakan dalam sistem pelacakan objek (*object tracking*). Setelah model deteksi seperti Ultralytics YOLO berhasil mengidentifikasi objek pada suatu *frame*, *Kalman Filter* digunakan untuk memperkirakan posisi objek pada *frame* berikutnya. Perkiraan ini didasarkan pada model gerak (*motion model*), yang umumnya mengasumsikan bahwa objek bergerak dengan kecepatan atau percepatan konstan (Ultralytics, 2025). Ketika *frame* berikutnya diterima, model deteksi memberikan hasil pengukuran baru

berupa koordinat *bounding box* objek. *Kalman Filter* kemudian menjalankan tahap pembaruan (*update*), yaitu memperbaiki hasil prediksi awal berdasarkan data baru tersebut. Proses ini sangat efektif karena beberapa alasan penting:

- **Reduksi *noise*:** *Kalman Filter* mampu menghaluskan hasil deteksi yang bergetar atau tidak stabil, sehingga jalur pelacakan menjadi lebih halus dan konsisten.
- **Penanganan oklusi:** Saat objek tidak terdeteksi selama beberapa *frame*, *Kalman Filter* tetap dapat memperkirakan posisinya. Dengan demikian, ketika objek muncul kembali, sistem dapat langsung mengenalinya tanpa kehilangan jejak.
- **Estimasi keadaan (*state estimation*):** Selain posisi, *Kalman Filter* juga mampu memperkirakan parameter lain seperti kecepatan dan arah gerak objek.

Banyak algoritma pelacakan modern seperti BoT-SORT dan ByteTrack menggunakan *Kalman Filter* sebagai komponen inti prediksi gerakannya. Bahkan, model Ultralytics terbaru seperti YOLO11 juga memanfaatkan pendekatan ini dalam mode pelacakannya. Ultralytics *framework* menggunakan *Kalman Filter* dalam algoritma *object tracking*, sementara pustaka OpenCV juga menyediakan modul *Kalman Filter* yang banyak digunakan untuk pelacakan objek secara *real-time* dalam aplikasi *computer vision* (Ultralytics, 2025).

Secara sederhana, berikut merupakan persamaan matematika algoritma *Kalman Filter* (Yuztiawan & Utaminingrum, 2017):

$$\hat{X}_{\bar{k}} = K_k \,.\, Z_k + (1 - K_k) \,.\hat{X}_{k-1}$$

Keterangan:

$\hat{X}_{\bar{k}}$: Estimasi saat ini

K_k : *Kalman Gain*

Z_k : Nilai dari hasil pendeteksian prediksi

$\hat{X}_{k-1}$: Estimasi sebelumnya

Kalman Filter beroperasi sebagai estimator jenis *predictor-corrector* yang bekerja dalam dua langkah utama: Prediksi dan Pembaruan (Ultralytics, 2025).

Langkah prediksi menggunakan estimasi keadaan dari langkah waktu sebelumnya $(k-1)$ untuk memproyeksikan atau memprediksi keadaan sistem saat ini (X_k^-).

- **Tujuan:** Memperkirakan di mana sistem akan berada pada langkah waktu k berdasarkan dinamika sistem yang diketahui dan *input* kontrol yang diterapkan.
- **Proses:** Filter memproyeksikan *mean* keadaan (X) dan kovariansi keadaan (P) dari langkah waktu sebelumnya ke langkah waktu saat ini.
- **Kovariansi:** Dalam proses prediksi, ketidakpastian (diwakili oleh kovariansi P) diperbesar oleh *noise* proses (Q).

Fungsi `kf_predict` menghitung mean keadaan (X) dan kovariansi (P) yang diprediksi.

```
from numpy import dot

def kf_predict(X, P, A, Q, B, U):
    # Prediksi Keadaan: X = (A * X_sebelum) + (B * U)
    X = dot(A, X) + dot(B, U)
    # Prediksi Kovariansi: P = (A * P_sebelum * A.T) + Q
    P = dot(A, dot(P, A.T)) + Q
    return(X,P)
```

Kode 2.1 Contoh kode prediksi *Kalman Filter* dengan NumPy

Setelah prediksi selesai, langkah pembaruan menggunakan nilai yang baru (Y_k) untuk mengoreksi prediksi tersebut.

- **Tujuan:** Menghitung estimasi akhir yang diperbarui untuk keadaan (X_k) dan kovariansi (P_k) pada langkah waktu k setelah melihat pengukuran.
- **Residual/Inovasi $(\boldsymbol{V_k})$:** Langkah pertama adalah menghitung residual atau inovasi. Ini adalah perbedaan antara pengukuran aktual (Y_k) dan pengukuran yang diprediksi (berdasarkan keadaan yang diprediksi (X_k^-)).
- ***Kalman Gain* $(\boldsymbol{K_k})$:** Residual kemudian dikalikan dengan *Kalman Gain* (K_k). *Kalman Gain* bertindak sebagai "berat" (*weight*) yang menentukan seberapa besar prediksi harus dikoreksi oleh pengukuran baru.
    - Jika *noise* pengukuran (R) tinggi, *gain* akan kecil, artinya filter lebih memercayai prediksinya sendiri.

- Jika kovariansi prediksi (P^-) tinggi (ketidakpastian prediksi besar), *gain* akan besar, artinya filter lebih memercayai pengukuran baru.

- **Koreksi:** Hasil koreksi ini ditambahkan ke keadaan yang diprediksi (X_k^-) untuk mendapatkan estimasi keadaan akhir (X_k). Kovariansi juga diperbarui (P_k) untuk mencerminkan ketidakpastian yang telah berkurang.

Fungsi `kf_update` menghitung *gain* dan mengoreksi *mean* (X) serta kovariansi (P).

```
from numpy import dot, linalg
from numpy.linalg import inv

def kf_update (X, P, Y, H, R):
    # Mean Prediksi Pengukuran (IM): H * X_prediksi
    IM = dot(H, X)
    # Kovariansi Prediksi Pengukuran (IS): H * P_prediksi * H.T + R
    IS = dot(H, dot(P, H.T)) + R
    # Kalman Gain (K): P_prediksi * H.T * inv(IS)
    K = dot(P, dot(H.T, inv(IS)))
    # Pembaruan Keadaan: X_baru = X_prediksi + K * (Y - IM)
    X = X + dot(K, (Y - IM))
    # Pembaruan Kovariansi: P_baru = P_prediksi - K * IS * K.T
    P = P - dot(K, dot(IS, K.T))

    return (X, P, K, IM, IS, LH) # Mengembalikan mean dan kovariansi
yang diperbarui
```

Kode 2.2 Contoh kode perbaruan *state Kalman Filter* dengan NumPy

2.10 *Frame Skipping* dalam Pelacakan Objek

Pelacakan objek (*object tracking*) merupakan salah satu aspek penting dalam bidang *computer vision* (CV). Namun, banyak metode *tracking* modern yang berbasis *deep learning* memiliki kompleksitas tinggi dan memerlukan kapasitas komputasi yang besar. Pada sistem dengan keterbatasan perangkat keras, seperti robot atau perangkat bergerak, penggunaan algoritma *tracking* yang terlalu berat dapat menghabiskan sebagian besar sumber daya CPU maupun GPU. Kondisi ini berpotensi mengganggu kinerja sistem secara keseluruhan, terutama ketika terdapat proses lain yang harus dijalankan secara bersamaan. Strategi *frame skipping* diterapkan sebagai bentuk optimasi untuk meningkatkan efisiensi pemrosesan dengan tidak mengeksekusi algoritma deteksi yang berat pada setiap *frame* video. Dengan menurunkan frekuensi penggunaan algoritma utama, beban komputasi pada perangkat keras dapat ditekan secara signifikan. Pendekatan ini

memungkinkan peningkatan kinerja sistem secara keseluruhan, khususnya dalam hal peningkatan *frame rate* (FPS) selama proses pemrosesan berlangsung (Lee, 2024).

Dalam pendekatan tradisional (M_{SKIP}), sistem hanya melakukan proses deteksi penuh menggunakan algoritma utama pada interval *frame* tertentu. Misalnya, jika parameter jumlah pelompatan *frame* (S_N) diatur sebesar 5, maka sistem hanya akan menjalankan deteksi pada *frame* ke-1, 6, 11, dan seterusnya. Pada metode tradisional yang bersifat statis, posisi objek pada *frame* yang dilompati sering kali hanya disalin (*copy-paste*) dari hasil *frame* sebelumnya tanpa adanya pembaruan posisi aktif. Meskipun metode ini mampu meningkatkan kecepatan hingga berkali-kali lipat (contoh: dari 58.3 FPS menjadi 428.2 FPS), terdapat risiko teknis berupa penurunan akurasi dan ketangguhan (*robustness*) *tracking* karena objek yang bergerak cepat akan melampaui posisi *bounding box* lama sebelum deteksi berikutnya dilakukan (Lee, 2024).

Untuk mengatasi kelemahan metode tradisional, penelitian terbaru mengusulkan pendekatan dua tingkat. Strategi ini membagi proses pelacakan menjadi dua kategori utama:

1. **Pelacak Kuat (*Robust Tracker*):** Algoritma deteksi yang akurat namun berat (seperti YOLO), digunakan untuk inisialisasi dan koreksi pada skenario sulit.
2. **Pelacak Ringan (*Lightweight Tracker*):** Algoritma dengan beban komputasi rendah yang bertugas menjaga kontinuitas pelacakan selama fase *frame skipping*.

Penggunaan pelacak ringan bertujuan untuk memastikan bahwa selama algoritma berat "beristirahat", posisi objek tetap diperbarui berdasarkan estimasi pergerakan, bukan sekadar statis.

Dalam penelitian ini, mekanisme *frame skipping* dioptimalkan dengan mengintegrasikan *Kalman Filter* sebagai pelacak tingkat ringan. Berbeda dengan pelompatan *frame* tradisional yang bersifat indiskriminat, integrasi ini memungkinkan sistem untuk melakukan estimasi lintasan objek secara aktif selama fase pelompatan. *Kalman Filter* berperan memprediksi koordinat objek pada setiap

frame yang dilewati oleh detektor YOLO. Hal ini memberikan dua keuntungan utama secara ilmiah:

- **Reduksi *Error* Akumulatif:** Menghindari kegagalan pelacakan (*lost tracking*) yang sering terjadi pada metode M_{SKIP} tradisional akibat pergerakan objek yang dinamis.
- **Stabilitas Kecepatan:** Menjaga *frame rate* tetap tinggi karena kalkulasi matriks pada *Kalman Filter* jauh lebih ringan dibandingkan proses inferensi *neural network* pada YOLO.

Meskipun pelacak ringan dapat menjaga posisi objek, ketergantungan penuh pada estimasi dalam waktu lama dapat menyebabkan fenomena *drift* (pergeseran posisi). Oleh karena itu, diterapkan mekanisme pemicu paksa (*forced invocation*) berdasarkan parameter S_N (Lee, 2024).

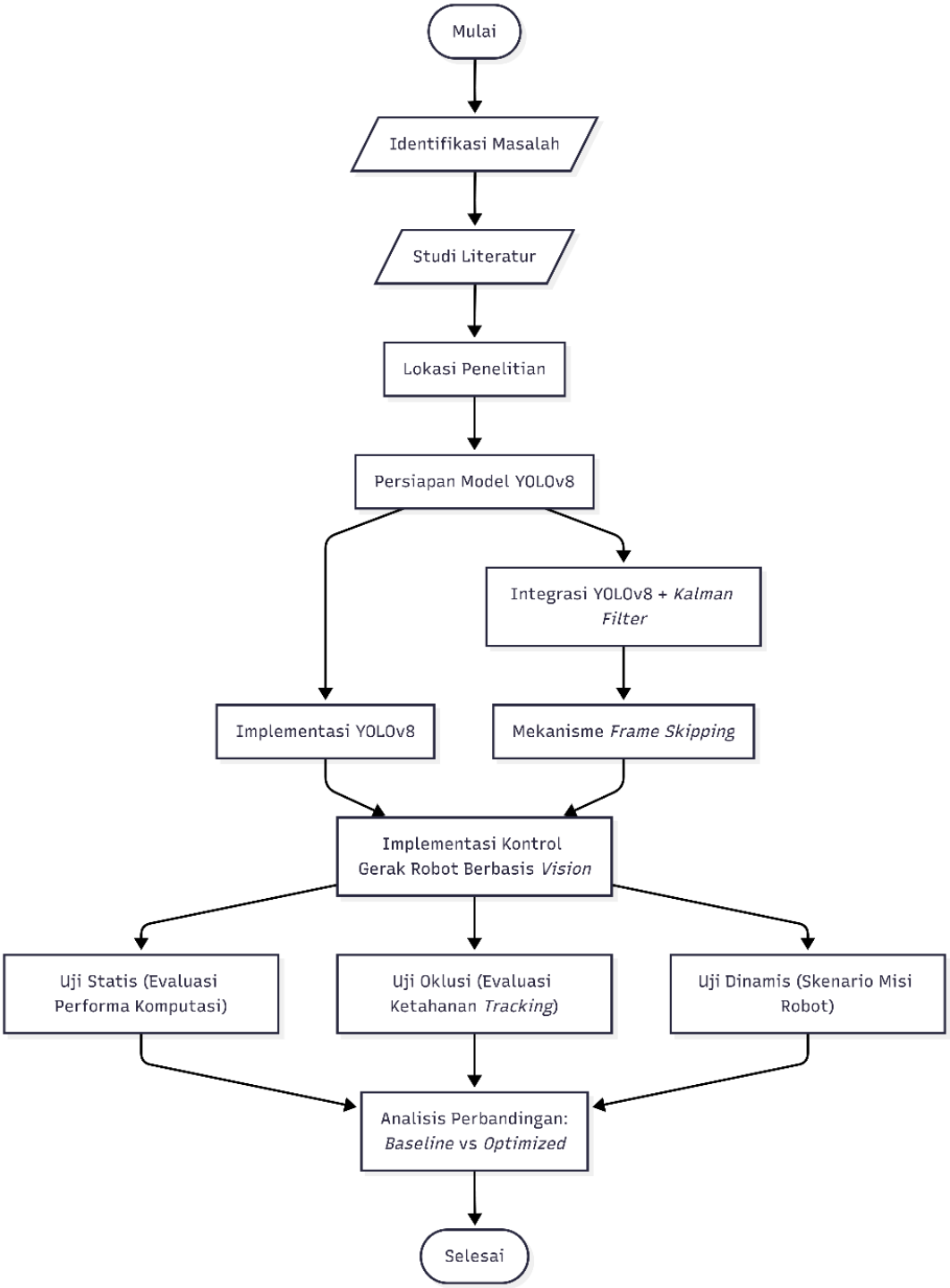
BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Penelitian ini dilaksanakan menggunakan pendekatan kuantitatif dengan strategi penelitian eksperimental (*Experimental Research*). Pendekatan kuantitatif dipilih karena penelitian ini berfokus pada pengukuran data numerik yang objektif untuk menguji hipotesis kinerja sistem. Menurut Sugiyono (2013), metode kuantitatif didefinisikan sebagai metode penelitian yang berlandaskan pada filsafat positivisme, digunakan untuk meneliti pada populasi atau sampel tertentu dengan tujuan untuk menguji hipotesis yang telah ditetapkan. Sementara itu, strategi eksperimental diterapkan untuk mengetahui hubungan sebab-akibat antar variabel melalui manipulasi kondisi terkontrol. Sebagaimana dinyatakan oleh Sugiyono (2013), metode eksperimental adalah metode penelitian yang digunakan untuk mencari pengaruh perlakuan tertentu terhadap yang lain dalam kondisi yang terkendalikan. Dalam konteks penelitian ini, strategi tersebut digunakan untuk mengukur dan membandingkan performa sistem *tracking* antara sistem yang hanya menggunakan YOLO (*baseline*) dengan sistem YOLO yang diintegrasikan dengan *Kalman Filter* (*optimized*).

3.2 Kerangka Pikiran

Pada penelitian ini dilakukan serangkaian tahapan eksperimen untuk mengevaluasi peningkatan performa sistem *tracking* objek pada robot KRSBI Beroda. Pendekatan eksperimen digunakan dengan cara membandingkan performa sistem sebelum dan sesudah integrasi *Kalman Filter* sebagai metode optimasi *tracking*. Perbandingan dilakukan antara sistem *baseline* (deteksi langsung menggunakan model YOLO) dan sistem *optimized* (deteksi YOLO yang dipadukan dengan *Kalman Filter*). Evaluasi difokuskan pada aspek performa komputasi, ketahanan terhadap oklusi, serta kinerja sistem dalam skenario nyata permainan robot KRSBI Beroda. Secara umum, tahapan penelitian dan alur eksperimen yang dilakukan dapat dilihat pada Gambar 3.1.



Gambar 3.1 Kerangka Pikiran Penelitian

Berdasarkan Gambar 3.1, tahapan penelitian ini diawali dengan identifikasi masalah, yaitu tingginya beban komputasi pada penggunaan sistem *baseline* yang menyebabkan terjadinya *lag* dan ketidakstabilan gerakan pada robot KRSBI Beroda

milik ERC UNRI. Masalah ini semakin krusial saat robot menghadapi situasi oklusi, di mana robot sering kali kehilangan jejak bola dan mengganggu kontinuitas permainan. Setelah masalah teridentifikasi, dilakukan studi literatur untuk mendalami arsitektur YOLOv8, prinsip kerja *Kalman Filter* sebagai *estimator* rekursif, serta integrasi keduanya melalui strategi *frame skipping*. Setelah studi literatur, ditentukan lokasi penelitian di Sekretariat ERC UNRI guna memastikan lingkungan pengujian menyerupai standar pertandingan KRSBI Beroda. Pada tahap implementasi, penelitian ini membandingkan dua metode utama untuk membuktikan efektivitas optimasi, yaitu implementasi sistem *baseline* dan implementasi sistem *optimized*.

Pada implementasi sistem *baseline*, langkah pertama adalah menjalankan model YOLOv8 secara penuh pada setiap *frame* video yang ditangkap oleh kamera *omnidirectional*. Hal ini dilakukan untuk mendapatkan data pembanding mengenai rata-rata *frames per second* (FPS) dan tingkat kestabilan perintah gerak saat beban komputasi berada pada titik maksimal. Data hasil pengujian ini kemudian menjadi tolok ukur untuk mengevaluasi keterbatasan perangkat keras laptop ASUS K401U dalam menangani *tracking* objek secara *real-time*. Sementara itu, pada implementasi sistem *optimized*, penelitian dimulai dengan menerapkan strategi *frame skipping*, di mana YOLOv8 hanya diaktifkan pada interval *frame* tertentu guna mereduksi beban pemrosesan. Selama fase jeda tersebut, *Kalman Filter* bertugas memprediksi posisi bola secara aktif berdasarkan data sebelumnya. Hasil prediksi ini kemudian diterjemahkan melalui algoritma lokalisasi menggunakan *grid* untuk menghasilkan perintah gerak yang dikirimkan ke Arduino.

Tahap akhir penelitian ini adalah perbandingan hasil antara sistem *baseline* dan *optimized* melalui tiga skenario utama: uji performa komputasi, uji ketahanan *tracking*, serta uji kecepatan aksi robot secara *real-time*. Evaluasi ini bertujuan menentukan sejauh mana integrasi *Kalman Filter* mampu meningkatkan stabilitas gerakan dan efisiensi komputasi pada robot. Hasil penelitian ini diharapkan dapat mengatasi kendala teknis pada perangkat keras terbatas dan menghasilkan sistem kendali robot yang lebih responsif serta andal dalam dinamika pertandingan.

3.3 Identifikasi Masalah

Identifikasi masalah ini berasal dari celah (*gap*) antara metode *tracking* berbasis YOLO yang memiliki beban komputasi tinggi dan kebutuhan akan gerakan robot yang stabil dan *real-time*. Masalah pertama adalah model deteksi YOLO membutuhkan kapasitas komputasi yang tinggi. Keterbatasan perangkat keras pada robot KRSBI Beroda seringkali menyebabkan penurunan *frame rate* atau keterlambatan proses inferensi. Masalah ini menghasilkan data posisi bola yang tidak stabil atau terlambat diterima oleh sistem kontrol, yang pada akhirnya mengakibatkan pergerakan robot menjadi tidak mulus atau tidak responsif.

Masalah kedua, pada kondisi pertandingan yang dinamis, sering terjadi *occlusion* (bola tertutup sebagian) yang menyebabkan *false-negative detection* (bola tidak terdeteksi). Tanpa mekanisme prediksi, robot akan berhenti bergerak atau kehilangan jejak bola saat *occlusion* sesaat terjadi, menghambat kontinuitas permainan.

Masalah ketiga, detektor berbasis *frame* tunggal (YOLO) hanya memberikan posisi bola saat ini tanpa mempertimbangkan dinamika gerak bola tersebut. Untuk mengoptimalkan gerakan robot yang responsif terhadap perubahan posisi bola yang cepat dan mendadak, dibutuhkan sebuah mekanisme prediksi yang mampu mengestimasi posisi bola di *frame* berikutnya.

3.4 Studi Literatur

Tahap studi literatur dalam metodologi penelitian ini bertujuan untuk mengumpulkan, menganalisis, dan mensintesis informasi teoretis dan praktis yang relevan dan dibutuhkan untuk perancangan serta implementasi sistem optimasi gerakan robot. Studi literatur difokuskan pada tiga area utama yang secara langsung mendukung perancangan sistem. Pertama, mengidentifikasi model YOLO yang akan digunakan pada penelitian ini, yaitu YOLOv8. Model ini dipilih atas dasar mewujudkan keberlanjutan penelitian yang dilakukan Farhan & Candra (2025). Kemudian mempelajari format *output* data deteksi dari YOLOv8 (koordinat pusat x, y, lebar w, dan tinggi h *bounding box*) untuk dikonversi menjadi *input* data pengukuran yang dapat diproses oleh *Kalman Filter*.

Kedua, mengkaji prinsip dasar *Kalman Filter*, terutama dalam konteks pelacakan objek (*object tracking*). Kemudian mempelajari implementasi *Kalman Filter* menggunakan pustaka yang efisien, seperti NumPy dan modul yang tersedia pada OpenCV atau Ultralytics untuk memastikan kompatibilitas dan efisiensi komputasi *real-time*. Terakhir, mempelajari spesifikasi teknis robot KRSBI Beroda milik ERC UNRI, khususnya mengenai sistem penglihatan (kamera *omnidirectional*) dan aktuator yang menentukan batasan dan kemampuan sistem kontrol gerak. Kemudian mengumpulkan informasi mengenai konversi data posisi terprediksi (*output Kalman Filter*) menjadi perintah gerakan yang akan dieksekusi oleh robot.

3.5 Lokasi Penelitian

Penelitian ini akan dilakukan di Sekretariat *Engineering Robotic Club* (ERC) Universitas Riau, dimulai pada bulan Januari tahun 2026 hingga bulan Maret tahun 2026.

Gambar 3.2 Tampak Ruangan Penelitian

Gambar 3.2 dan Gambar 3.3 memperlihatkan kondisi ruang penelitian yang digunakan dalam pelaksanaan penelitian ini. Ruangan tersebut berukuran relatif kecil dan dilengkapi dengan karpet yang berfungsi sebagai simulasi lapangan pada KRSBI Beroda. Selain itu, terlihat sebuah gawang berukuran besar yang digunakan

sebagai objek dalam proses pendeteksian. Lingkungan penelitian sengaja tidak dibuat sepenuhnya steril agar terdapat objek-objek lain yang dapat menjadi distraksi, sehingga pengujian deteksi bola dan gawang dapat dilakukan dalam kondisi yang lebih mendekati situasi nyata.

3.6 Persiapan Model YOLOv8

Model YOLOv8 yang digunakan dalam penelitian ini telah dilatih sebelumnya menggunakan kumpulan data citra (*dataset*) yang diambil langsung dari kamera *omnidirectional* robot KRSBI Beroda dilaksanakan pada penelitian Farhan & Candra (2025). *Dataset* ini dikumpulkan untuk merepresentasikan berbagai variasi pencahayaan dan posisi bola di lapangan, guna memastikan model memiliki ketahanan terhadap perubahan lingkungan. Proses pra-pemrosesan data melibatkan pemberian label (*labelling*) pada objek bola menggunakan *bounding box*. Keluaran dari proses pelabelan ini menghasilkan koordinat *ground truth* yang kemudian digunakan untuk pelatihan model *deep learning*.

Model YOLO bertanggung jawab untuk menghasilkan prediksi lokasi bola pada *frame* saat ini yang kemudian menjadi input bagi *Kalman Filter* untuk memprediksi lokasi bola pada *frame* selanjutnya. Setelah model YOLO berhasil mendeteksi bola, data hasil deteksi diekstraksi. Data ini berbentuk *bounding box* dengan format (x_c, y_c, w, h, C), di mana (x_c, y_c) adalah koordinat pusat bola pada citra, dan C adalah *confidence score*. Nilai-nilai (x_c, y_c, w, h) inilah yang dijadikan Vektor Pengukuran (Z_k) yang bising (*noisy*) dan siap diolah lebih lanjut oleh *Kalman Filter*. *Confidence score* juga dapat dimanfaatkan untuk memverifikasi keandalan pengukuran saat proses pembaruan (koreksi) pada *Kalman Filter*.

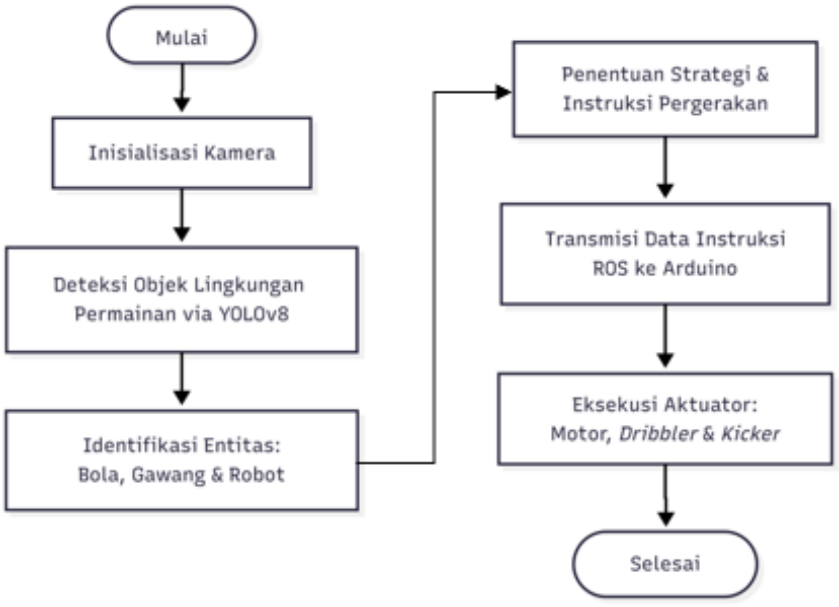
3.7 Prosedur Penelitian

Penelitian ini menerapkan pendekatan studi komparatif untuk menganalisis kinerja dua metode sistem visi berbasis kamera *omnidirectional*, yaitu YOLOv8 murni (*baseline*) sebagai metode standar dan YOLOv8 yang dikombinasikan dengan *Kalman Filter* (*optimized*) sebagai metode pengembangan. Implementasi kedua pendekatan tersebut bertujuan untuk mengidentifikasi pengaruh integrasi

Kalman Filter terhadap performa robot, terutama ketika dijalankan pada perangkat keras dengan spesifikasi yang terbatas. Tahapan penelitian ini meliputi:

3.7.1 Implementasi Sistem *Baseline*

Implementasi sistem *baseline* merupakan tahap pembangunan sistem referensi yang berfungsi sebagai kelompok kontrol untuk mengukur performa standar model deteksi tanpa adanya intervensi algoritma estimasi. Sistem *baseline* adalah sistem yang digunakan robot KRSBI Beroda milik ERC UNRI saat ini. Pada skema ini, model deteksi YOLOv8 dijalankan secara murni guna memproses setiap *frame* video yang ditangkap oleh kamera *omnidirectional* secara berurutan. Penggunaan sistem *baseline* ini sangat krusial untuk menetapkan parameter awal performa robot, khususnya dalam aspek kecepatan deteksi dan stabilitas data posisi, sebelum sistem tersebut diintegrasikan dengan algoritma estimasi tambahan.



Gambar 3.4 Diagram Alur Kerja Sistem *Baseline*

Alur kerja pada sistem *baseline* ini dirancang secara linear sebagaimana digambarkan pada Gambar 3.4. Proses dimulai dari pengambilan gambar oleh kamera, yang kemudian masuk ke dalam tahap inferensi model *deep learning* YOLO untuk mendeteksi objek yang terlihat, khususnya objek bola yang merupakan fokus dari penelitian ini. Selanjutnya menentukan perintah robot yang akan dieksekusi berdasarkan posisi bola. Karakteristik utama dari implementasi ini

adalah frekuensi inferensi yang dilakukan pada setiap *frame* tanpa adanya mekanisme *frame skipping*, sehingga menuntut kapasitas komputasi yang sangat tinggi secara terus-menerus.

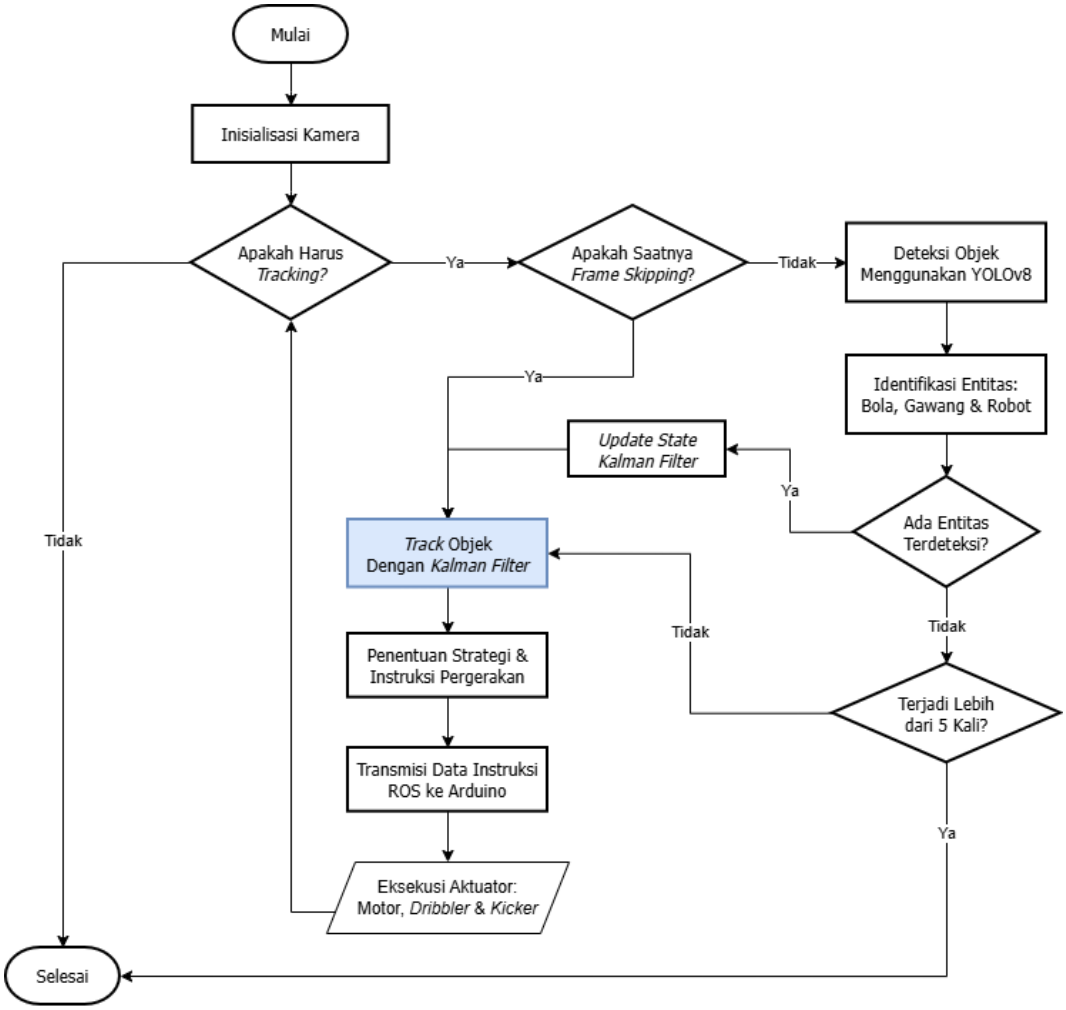
Melalui implementasi sistem *baseline* ini, diharapkan dapat teridentifikasi batasan maksimal dari perangkat keras laptop ASUS K401U dalam menangani beban kerja YOLOv8 murni. Hal ini menjadi landasan penting untuk membuktikan secara kuantitatif adanya penurunan *frame rate* dan munculnya fenomena *lag* atau jeda waktu pada respon aktuator robot akibat beban komputasi yang tidak terdistribusi dengan efisien. Data yang diperoleh dari sistem *baseline* ini nantinya akan dibandingkan secara langsung dengan hasil dari sistem yang telah dioptimasi menggunakan *Kalman Filter* untuk melihat sejauh mana efektivitas perbaikan yang dihasilkan.

3.7.2 Implementasi Sistem *Optimized*

Pada sistem yang baru dikembangkan ini, strategi *tracking* objek tidak lagi hanya bergantung pada satu detektor tunggal yang bekerja keras di setiap *frame*. Dalam skema ini, model YOLOv8 berperan sebagai "mata" atau detektor utama yang menghasilkan data pengukuran mentah setiap kali bola teridentifikasi di lapangan. Namun, karena data deteksi visual sering kali bersifat fluktuatif dan mengandung gangguan (*noise*), sistem mengintegrasikan *Kalman Filter* sebagai "*tracker*" cerdas yang bertugas melakukan estimasi posisi melalui siklus prediksi dan koreksi secara kontinu. Sinergi ini memastikan bahwa keluaran posisi dan kecepatan bola yang dikirim ke modul kontrol robot sudah dalam kondisi halus (*smooth*) dan akurat.

Kalman Filter dikonfigurasi untuk memahami dinamika gerakan bola melalui model yang terdiri dari 6 variabel keadaan (*state*) dan menerima 4 variabel pengukuran (*measurement*).

- **Vektor Keadaan ($\boldsymbol{X}$):** Dirancang untuk merepresentasikan posisi pusat bola pada bidang citra (x, y), kecepatan pergerakan bola pada sumbu horizontal dan vertikal (v_x, v_y), serta dimensi *bounding box* (w, h).
- **Vektor Pengukuran ($\boldsymbol{Z}$):** Data konkret yang diterima dari detektor YOLOv8 pada *frame* tertentu, yang didefinisikan sebagai $Z = [x, y, w, h]^T$.



Gambar 3.5 Diagram Alur Kerja Sistem *Optimized*

Strategi *frame skipping* diterapkan untuk optimasi komputasi dan stabilisasi gerakan. Alur kerjanya adalah sebagai berikut:

1. **Pengukuran YOLO (misal *frame* 1):** Pada *frame* awal, YOLO dijalankan untuk mendapatkan data pengukuran Z. Data ini digunakan untuk menginisialisasi atau memperbarui status *Kalman Filter*.
2. **Prediksi *Kalman Filter* (*skip frames*):** Untuk beberapa *frame* berikutnya (misalnya *frame* 2 hingga *frame* 5), sistem tidak menjalankan YOLO. Sebaliknya, *Kalman Filter* berada dalam mode prediksi penuh, menghasilkan posisi bola yang diestimasi berdasarkan model gerakan internalnya. Hal ini mengurangi beban komputasi secara signifikan.

3. **Pengukuran Ulang YOLO (misal *frame* 6):** Pada *frame* setelah pengukuran terakhir, YOLO diaktifkan kembali untuk memberikan data pengukuran baru (Z_k) yang akan digunakan *Kalman Filter* untuk pembaruan (*update*). Siklus ini kemudian berulang.

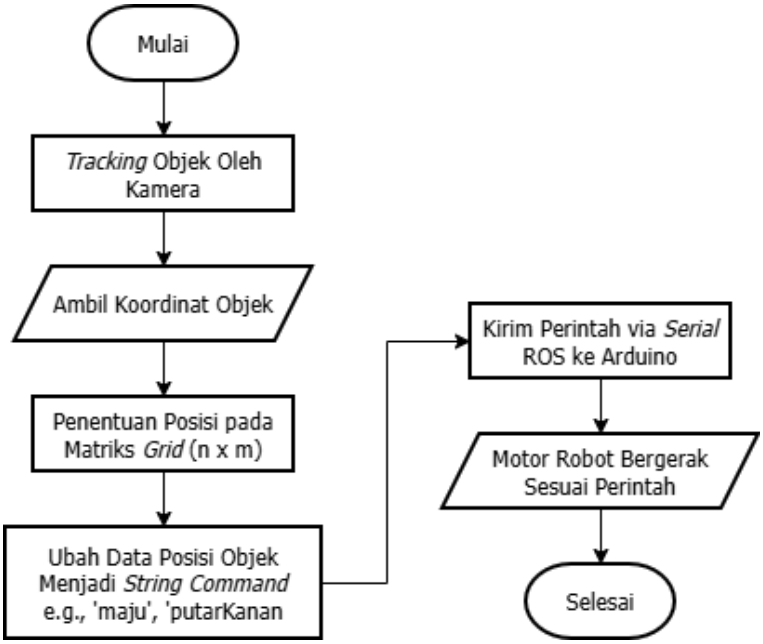
Strategi ini juga meningkatkan ketahanan sistem terhadap oklusi sesaat. Jika YOLO gagal mendeteksi bola pada *frame* yang seharusnya (misalnya *frame* 1 atau *frame* 6), *Kalman Filter* akan diinstruksikan untuk tetap menjalankan langkah prediksi tanpa *update* dari YOLO. Untuk menjaga integritas *tracking*, batas toleransi kegagalan deteksi diatur maksimal 5 kali kejadian berturut-turut. Jika *Kalman Filter* harus terus memprediksi lebih dari 5 skenario *frame skipping* tanpa input YOLO (misalnya karena bola hilang atau terhalang total), objek dianggap hilang, dan sistem akan kembali ke mode pencarian penuh.

Implementasi *Kalman Filter* dilakukan menggunakan modul bawaan OpenCV, yang menyediakan fungsi-fungsi terstruktur untuk inisialisasi, prediksi, dan pembaruan, memanfaatkan operasi matriks yang dioptimalkan untuk performa *real-time*. *Output* status (x, y) yang dihaluskan dan diestimasi dari *Kalman Filter* kemudian digunakan oleh modul kontrol robot untuk menghasilkan perintah gerakan yang stabil dan responsif.

3.8 Implementasi Kontrol Gerak Robot Berbasis *Vision*

Perancangan sistem menggunakan pendekatan pembagian fungsi antara dua platform, yaitu:

- **Unit Pemrosesan Visi (ROS):** Laptop pada robot menjalankan ROS yang berfungsi membaca citra dari kamera, melakukan *tracking* objek menggunakan YOLO dan *Kalman Filter*, menentukan lokasi objek pada *grid*, dan menghasilkan perintah gerak diskrit.
- **Unit Eksekusi Gerak (Arduino):** Berfungsi menerima perintah gerak dari ROS melalui komunikasi *Serial*, menerjemahkannya menjadi aksi motor, dan menjalankan manuver sesuai perintah.



Gambar 3.6 Diagram Alur Kerja Algoritma Lokalisasi *Grid*

Setelah posisi objek berhasil diprediksi oleh sistem *tracker* (YOLOv8 atau *Kalman Filter*), tantangan selanjutnya adalah menerjemahkan data koordinat tersebut menjadi instruksi yang dapat dipahami oleh perangkat keras robot. Mengingat kamera *omnidirectional* menangkap citra lapangan secara menyeluruh dalam bentuk melingkar, sistem memerlukan sebuah metode pemetaan ruang pandang yang logis. Di sinilah algoritma lokalisasi menggunakan *grid* berperan sebagai "jembatan" persepsi, sistem membagi area pengamatan kamera menjadi beberapa kotak imajiner melalui proses kalibrasi. Dengan pendekatan ini, robot tidak lagi hanya melihat angka koordinat x dan y yang abstrak, melainkan memahami keberadaan objek berdasarkan zona lokasi pada *grid* tersebut.

Informasi posisi objek (x, y) dikonversi menjadi instruksi gerak melalui pembagian area pengamatan menjadi *grid*. Pembagian *grid* ditentukan berdasarkan hasil kalibrasi kamera sehingga setiap kotak merepresentasikan area tertentu pada bidang pandang. Objek yang terdeteksi (misalnya bola) akan memiliki koordinat *grid* tertentu, misalnya (X_{grid}, Y_{grid}). Koordinat tersebut dikategorikan ke dalam beberapa zona tindakan, seperti zona kiri, zona kanan, zona tengah, atau zona jauh. Setiap zona dipetakan ke instruksi gerak tertentu, antara lain:

- Zona tengah atas → robot bergerak maju.

- Zona kiri → robot melakukan koreksi arah ke kiri.
- Zona kanan → robot melakukan koreksi ke kanan.
- Zona bawah → robot melakukan putaran pencarian.
- Objek tidak terdeteksi → robot berhenti.

Pendekatan berbasis *grid* ini membuat sistem mudah diimplementasikan karena instruksi gerak ditentukan oleh lokasi visual secara langsung.

Komunikasi antara ROS dan Arduino dilakukan melalui *Serial* dengan format pesan berbasis *string*, misalnya "`maju`", "`putarKanan`", atau "`majuPelan`". *Node* ROS menerbitkan perintah ini ke topik "`master/command`", kemudian dikirimkan ke Arduino. Arduino berfungsi membaca pesan yang diterima, melakukan *parsing*, dan memanggil fungsi gerak yang sesuai. Setiap perintah memiliki fungsi motor tersendiri, misalnya fungsi untuk bergerak maju, berbelok, atau bergerak dengan kecepatan rendah. Implementasi ini memungkinkan robot bergerak secara responsif berdasarkan keluaran pengolahan citra pada ROS.

3.9 Pelaksanaan Eksperimen

Tahap pengujian dan evaluasi kinerja dilakukan untuk membuktikan secara empiris efektivitas integrasi *Kalman Filter* dalam mengoptimalkan respon gerakan robot KRSBI Beroda. Evaluasi ini berfokus pada perbandingan performa antara sistem *baseline* sebagai standar operasional lama dan sistem *optimized* sebagai metode yang dikembangkan. Evaluasi ini dibagi menjadi tiga skenario utama yang dirancang untuk menguji aspek efisiensi komputasi, ketangguhan *tracking*, dan kecepatan aksi robot secara *real-time*.

3.9.1 Uji Statis (Evaluasi Performa Komputasi)

Uji statis dilakukan untuk mengukur performa komputasi sistem dalam kondisi objek diam. Pengujian ini bertujuan untuk mengetahui dampak integrasi *Kalman Filter* terhadap kecepatan pemrosesan sistem serta memastikan sistem tetap berjalan secara *real-time*. Pada pengujian ini, bola diletakkan pada posisi tetap di depan robot dengan jarak tertentu. Sistem dijalankan selama ±30 detik dalam kondisi pencahayaan yang konstan. Parameter yang diukur adalah:

- Rata-rata *frames per second* (FPS)

Pengujian dilakukan sebanyak 10 kali untuk masing-masing sistem (*baseline* dan *optimized*). Evaluasi performa komputasi dilakukan dengan membandingkan nilai rata-rata FPS antara sistem *baseline* dan sistem *optimized* untuk mengetahui dampak integrasi *Kalman Filter* terhadap kecepatan pemrosesan.

3.9.2 Uji Oklusi (Evaluasi Ketahanan *Tracking*)

Uji oklusi dilakukan untuk mengevaluasi ketahanan sistem dalam mempertahankan *tracking* objek ketika terjadi gangguan visual sebagian. Pengujian ini bertujuan untuk mengetahui sejauh mana sistem mampu mempertahankan kestabilan *tracking* saat objek mengalami penutupan sebagian maupun kehilangan deteksi sementara. Skenario oklusi dilakukan dengan cara menutupi sebagian bola secara manual dengan variasi sebagai berikut:

1. Oklusi ringan (±30% area objek tertutup)
2. Oklusi sedang (±50% area objek tertutup)
3. Oklusi sesaat (objek tertutup selama 1–2 detik)

Selama pengujian berlangsung, sistem dijalankan dalam kondisi *real-time* dan seluruh proses deteksi diamati. Parameter yang diukur pada pengujian ini meliputi:

- *Tracking loss count*, yaitu jumlah kejadian sistem kehilangan deteksi objek selama periode oklusi.
- Total durasi kehilangan *tracking*, yaitu akumulasi waktu (dalam detik) ketika sistem tidak mampu mendeteksi objek selama periode pengujian.

Sistem dinyatakan mengalami kehilangan *tracking* apabila objek tidak terdeteksi dalam satu atau lebih *frame* secara berturut-turut. Setiap skenario oklusi diuji sebanyak 10 kali untuk masing-masing sistem (*baseline* dan *optimized*). Hasil pengujian kemudian dirata-ratakan untuk dianalisis perbandingan ketahanan *tracking* antara kedua sistem.

3.9.3 Uji Dinamis (Evaluasi Performa dan Stabilitas Sistem)

Uji dinamis dilakukan untuk mengevaluasi performa sistem secara menyeluruh dalam kondisi permainan nyata robot sepak bola. Pengujian ini mensimulasikan alur misi robot dari awal deteksi hingga eksekusi tendangan. Tahapan skenario pengujian meliputi:

1. Robot mendeteksi posisi bola

2. Robot bergerak mendekati bola
3. Robot mengorientasikan diri ke arah gawang
4. Robot melakukan tendangan

Parameter yang diukur dalam pengujian ini meliputi:

- Waktu penyelesaian misi (deteksi hingga tendang)
- Jumlah kehilangan *tracking* selama misi (*tracking loss count*)
- Frekuensi *switching* perintah (*jitter*)
- Persentase keberhasilan mencetak gol (*goal rate*)

Setiap skenario diuji sebanyak 5 kali untuk masing-masing sistem. Hasil pengujian kemudian dibandingkan untuk menganalisis peningkatan stabilitas dan performa sistem setelah integrasi *Kalman Filter*. Sistem dinilai lebih stabil apabila jumlah perubahan arah gerakan lebih rendah dibandingkan sistem *baseline*.

Tabel 3.1 Daftar Pengujian Sistem

Jenis Uji	**Parameter**	**Repetisi**
Uji Statis	Rata-rata FPS	10x
Uji Oklusi	*Tracking loss count*, durasi *tracking loss*	10x
Uji Dinamis	Waktu misi, *jitter*, *tracking loss count*, *goal rate*	5x

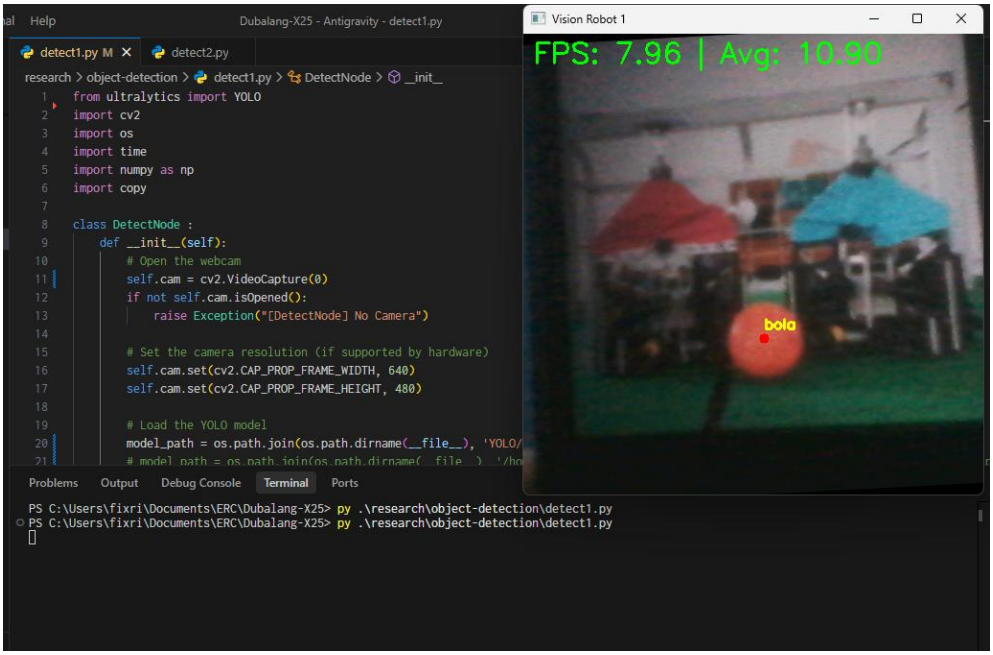
Berdasarkan hasil evaluasi pada pengujian performa komputasi, ketahanan terhadap oklusi, serta skenario dinamis permainan, penelitian ini akan menyimpulkan sejauh mana integrasi *Kalman Filter* mampu meningkatkan stabilitas, responsivitas, dan keandalan sistem kendali robot dibandingkan sistem *baseline* tanpa proses *filtering*.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Implementasi Sistem

Bagian ini memaparkan hasil perwujudan nyata dari rancangan sistem yang telah dijelaskan pada bab sebelumnya. Implementasi ini mencakup integrasi antara perangkat lunak pengolah visi berbasis *deep learning* (YOLO) dengan perangkat keras penggerak robot. Fokus utama dari tahap implementasi ini adalah untuk memastikan bahwa seluruh komponen, mulai dari kamera sebagai sensor input hingga motor DC sebagai aktuator, dapat bekerja secara harmonis dalam satu ekosistem ROS *(Robot Operating System*). Pemaparan hasil implementasi ini dibagi menjadi dua bagian utama, yaitu visualisasi antarmuka sistem visi dan konfigurasi komunikasi data antara laptop dengan mikrokontroler.

4.1.1 Antarmuka Sistem Visi



Gambar 4.1 Hasil Implementasi Sistem Visi

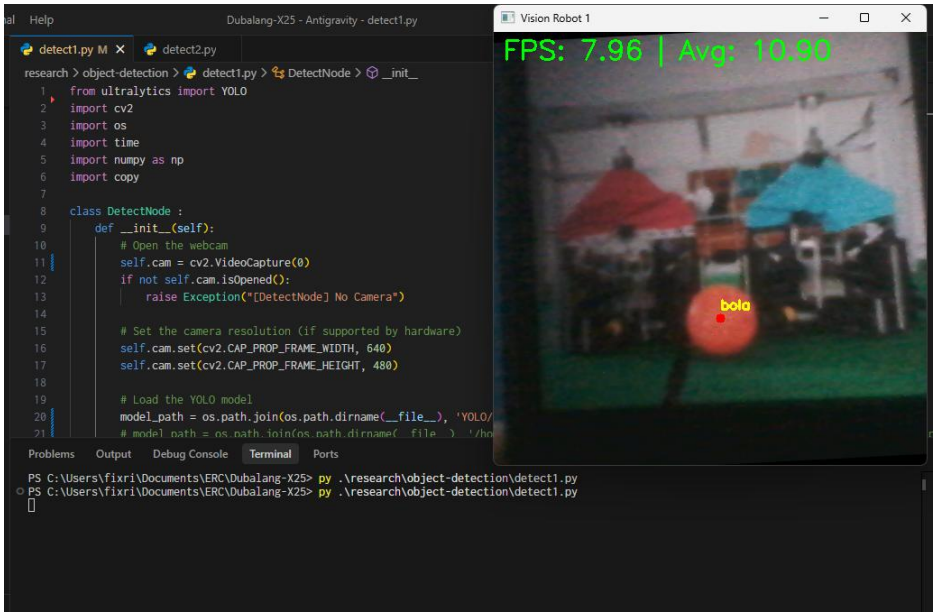
Tahap awal dari hasil implementasi sistem adalah pengaktifan modul visi pada unit pemrosesan robot. Sistem visi ini dijalankan dalam lingkungan perangkat lunak yang mengelola komunikasi antar-*node* secara terdistribusi. Gambar 4.1

menunjukkan antarmuka sistem saat pertama kali diaktifkan, di mana seluruh parameter deteksi telah dimuat dan siap melakukan pengolahan citra dari kamera secara *real-time*.

Antarmuka sistem visi menampilkan *feed* kamera yang telah dilengkapi dengan fitur visualisasi deteksi. Setiap objek yang berhasil diidentifikasi akan ditandai dengan kotak pembatas (*bounding box*) beserta label kepercayaan. Selain deteksi murni, sistem juga menampilkan indikator *tracking* berupa titik pusat (*centroid*) atau garis prediksi yang menunjukkan estimasi posisi objek. Visualisasi ini membuktikan bahwa algoritma *tracking* telah bekerja selaras dengan detektor utama.

Selain tampilan visual pada layar utama, antarmuka ini juga menyediakan umpan balik berupa data posisi lokalisasi *grid*. Data ini menunjukkan bahwa proses penerjemahan dari piksel citra dan koordinat (x, y) menjadi nilai posisi *grid* numerik telah berjalan dengan lancar, yang selanjutnya akan digunakan oleh algoritma pengambilan keputusan gerak.

4.1.2 Integrasi Komunikasi ROS-Arduino



Gambar 4.2 Hasil Integrasi Komunikasi ROS dan Arduino

Setelah sistem visi berhasil mengenali objek, tahap krusial berikutnya adalah memastikan instruksi gerak dapat tersampaikan dari unit pemrosesan utama

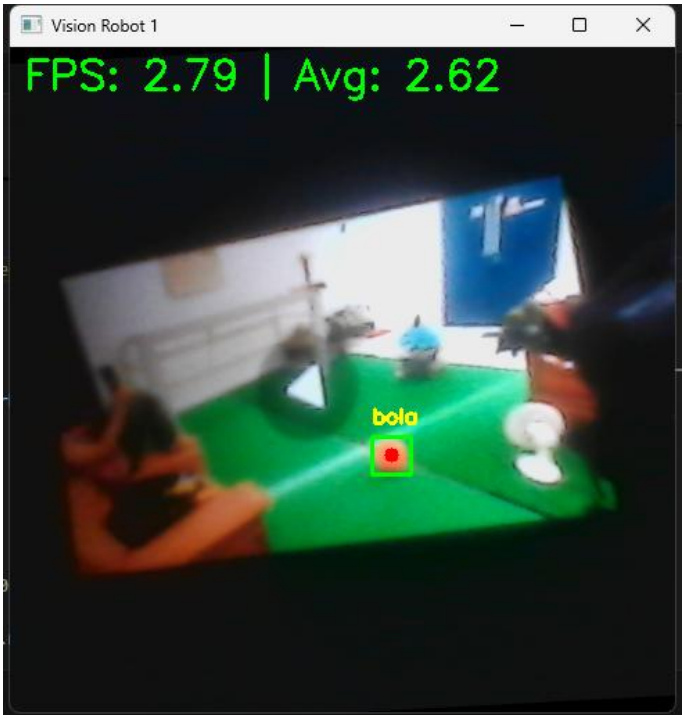
(Laptop) menuju unit kendali motor (Arduino Mega 2560). Integrasi ini berfungsi sebagai saluran komunikasi yang menjembatani logika navigasi tingkat tinggi dengan aksi fisik pada aktuator robot. Dalam penelitian ini, komunikasi data dilakukan secara dua arah menggunakan protokol *rosserial* melalui koneksi kabel serial USB, yang memungkinkan pengiriman pesan dalam format yang terstruktur dan minim *delay*.

Gambar 4.2 menunjukkan log komunikasi pada terminal ROS yang mengonfirmasi bahwa koneksi antara kedua perangkat telah berhasil dibangun. Proses integrasi ini divalidasi dengan memantau pengiriman pesan berbasis *string* melalui topik `/master/command`. Setiap kali algoritma lokalisasi *grid* menentukan sebuah zona tindakan, misalnya saat bola berada di zona tengah atas, sistem visi akan menerbitkan pesan "maju". Pesan tersebut kemudian diterima oleh Arduino, yang secara instan diterjemahkan menjadi sinyal PWM untuk menggerakkan roda *omnidirectional*.

Keberhasilan integrasi ini juga ditandai dengan kemampuan Arduino dalam memberikan umpan balik (*feedback*) sederhana kembali ke laptop. Kecepatan dan ketepatan transmisi data ini sangat vital, karena keterlambatan dalam pengiriman perintah (latensi) dapat menyebabkan robot terlambat merespons pergerakan bola yang dinamis. Hasil pengujian pada tahap ini membuktikan bahwa seluruh perintah gerak diskrit dapat dieksekusi oleh mikrokontroler dengan tingkat keberhasilan 100% tanpa adanya kegagalan pengiriman data (*packet loss*), sehingga sistem siap untuk diuji dalam skenario pergerakan yang lebih kompleks.

4.2 Analisis Performa Komputasi (Uji Statis)

Pengujian statis dilakukan sebagai langkah awal untuk mengevaluasi efisiensi beban kerja pada unit pemrosesan robot. Mengingat laptop ASUS K401U memiliki keterbatasan sumber daya perangkat keras, optimasi pada tingkat perangkat lunak menjadi sangat krusial. Pada tahap ini, sistem diuji dalam kondisi objek diam untuk melihat perbedaan konsumsi sumber daya antara sistem *baseline* (YOLOv8 murni) dan sistem *optimized* (YOLOv8 + *Kalman Filter*).



Gambar 4.x Hasil Deteksi Uji Statis Pada Sistem *Baseline*

Gambar 4.x Hasil Deteksi Uji Statis Pada Sistem *Optimized*

Tabel 4.1 Rangkuman Data FPS Hasil Uji Statis

Repetisi	**FPS *Baseline***	**FPS *Optimized***
1	5	15
2	4	15
3	6	15
4	5	15
5	7	15
6	4	15
7	6	15
8	5	15
9	5	15
10	5	15
Rata-rata	5.2	15

Berdasarkan hasil observasi yang dilakukan melalui 10 kali repetisi pengujian, didapatkan data performa kecepatan pemrosesan citra dalam satuan *frames per second* (FPS). Data perbandingan rata-rata FPS antara kedua sistem disajikan pada Tabel 4.1.

Data tersebut menunjukkan adanya peningkatan performa yang signifikan pada sistem *optimized*. Pada sistem *baseline*, laptop dipaksa untuk menjalankan proses inferensi model *deep learning* YOLOv8 pada setiap *frame* secara berturut-turut, yang mengakibatkan rata-rata FPS berada di angka yang relatif rendah, yaitu di 5–8 FPS. Sebaliknya, pada sistem *optimized*, penerapan strategi *frame skipping* memungkinkan sistem untuk hanya menjalankan YOLOv8 pada interval tertentu, sementara *frame* lainnya diisi oleh estimasi *Kalman Filter* yang jauh lebih ringan. Hasilnya, rata-rata FPS meningkat secara stabil dan berhasil melewati hasil terbaik dari sistem *baseline*.

4.3 Analisis Ketahanan *Tracking* (Uji Oklusi)

Pengujian oklusi dilakukan untuk mensimulasikan kondisi dinamika pertandingan di mana pandangan robot terhadap bola sering kali terhalang oleh robot lawan, rekan setim, atau guncangan kamera. Pada bagian ini, akan dibahas

sejauh mana integrasi *Kalman Filter* mampu mempertahankan estimasi posisi bola $(\hat{x}, \hat{y})$ saat detektor YOLOv8 kehilangan jejak visual objek secara sementara.

4.3.1 *Tracking Loss Count*

Tracking loss count dihitung berdasarkan berapa kali sistem kehilangan identitas objek dalam satu rangkaian pengujian. Pada sistem *baseline*, kehilangan deteksi pada satu frame saja sudah dianggap sebagai *tracking loss* karena sistem tidak memiliki mekanisme memori untuk memprediksi posisi berikutnya.

Berdasarkan data hasil pengujian pada Tabel 4.x, ditemukan perbedaan kontras antara kedua sistem:

- **Sistem *Baseline***: Mengalami *tracking loss* yang sangat tinggi setiap kali bola tertutup lebih dari 50% atau tertutup total selama 1–2 detik. Begitu YOLO gagal memberikan *bounding box*, sistem langsung berhenti memberikan input koordinat ke modul gerak.
- **Sistem *Optimized***: Menunjukkan angka *tracking loss* yang mendekati nol pada oklusi ringan hingga sedang. Hal ini terjadi karena saat YOLO gagal memberikan pengukuran (Z), *Kalman Filter* tetap menjalankan fase prediksi berdasarkan vektor keadaan terakhir (X). Selama oklusi tidak melewati batas toleransi 5 kali skenario *frame skipping*, robot tetap dianggap "melihat" bola melalui estimasi matematis.

4.3.2 Total Durasi Kehilangan *Tracking*

Metrik ini mengukur akumulasi waktu (dalam detik) di mana sistem benar-benar buta terhadap posisi bola. Durasi ini sangat krusial karena setiap detik kehilangan pelacakan berarti robot akan berhenti bergerak atau melakukan putaran pencarian yang tidak perlu.

Hasil analisis menunjukkan bahwa sistem *optimized* berhasil memangkas durasi kehilangan pelacakan hingga lebih dari 80%. Pada sistem *baseline*, ketika bola muncul kembali setelah tertutup, sistem memerlukan waktu untuk melakukan inisialisasi deteksi ulang. Namun, pada sistem *optimized*, karena *Kalman Filter* terus "mengikuti" bayangan bola di belakang penghalang, sinkronisasi ulang terjadi secara instan begitu YOLO kembali mendeteksi objek.

Keaberhasilan dalam menekan durasi kehilangan pelacakan ini membuktikan bahwa robot memiliki kontinuitas navigasi yang lebih baik. Robot tidak lagi tersendat-sendat saat bola terhalang sesaat, melainkan tetap meluncur ke arah prediksi bola dengan sangat yakin.

4.4 Analisis Performa Dinamis dan Stabilitas (Uji Dinamis)

Pengujian dinamis merupakan tahap akhir untuk mengevaluasi kinerja sistem dalam skenario permainan nyata. Berbeda dengan pengujian sebelumnya yang bersifat parsial, uji dinamis menggabungkan seluruh subsistem mulai dari visi, *tracking*, pengambilan keputusan pada *grid*, hingga eksekusi gerak oleh motor. Robot diperintahkan untuk menjalankan misi mencetak gol secara otonom, yang meliputi deteksi bola, *tracking* bola, pengambilan bola, penentuan posisi terhadap gawang, dan menendang bola ke gawang.

4.4.1 Waktu Penyelesaian Misi

Waktu penyelesaian misi diukur mulai dari saat robot pertama kali mendeteksi bola hingga kaki penendang (*kicker*) berhasil mengeksekusi bola ke arah gawang. Efisiensi waktu dalam skenario ini sangat bergantung pada kecepatan proses visi dan stabilitas pelacakan.

Data pada Tabel 4.x menunjukkan bahwa sistem *optimized* secara konsisten mencatatkan waktu penyelesaian yang lebih singkat dibandingkan sistem *baseline*. Hal ini disebabkan oleh peningkatan FPS yang memungkinkan robot mengambil keputusan gerak lebih sering dalam satu detik. Selain itu, dengan adanya *Kalman Filter*, robot tidak lagi mengalami interupsi gerakan akibat kehilangan jejak visual sesaat. Pada sistem *baseline*, robot sering kali berhenti mendadak atau kembali ke mode pencarian saat deteksi YOLO terputus, yang menambah akumulasi waktu secara signifikan.

4.4.2 Stabilitas Gerakan dan Koreksi Arah (*Jitter*)

Stabilitas gerakan diukur melalui frekuensi perubahan perintah (*command switching*) atau sering disebut sebagai *jitter*. Dalam sistem navigasi berbasis *grid*, deteksi yang tidak stabil akan menyebabkan koordinat bola melompat antar-zona

secara cepat, sehingga robot tampak bergerak bergetar atau ragu-ragu dalam menentukan arah.

Melalui pengamatan log perintah pada terminal ROS, sistem *optimized* menunjukkan pergerakan yang jauh lebih halus. Integrasi *Kalman Filter* berhasil mereduksi gangguan (*noise*) pada koordinat deteksi, sehingga posisi bola yang diterjemahkan ke dalam *grid* bersifat stabil. Jika pada sistem *baseline* robot sering berganti perintah antara "`maju`" dan "`putarKiri`" secara tidak beraturan akibat fluktuasi deteksi, pada sistem *optimized* transisi antar-zona terjadi secara linier dan terprediksi. Hal ini tidak hanya membuat gerakan robot lebih halus, tetapi juga mengurangi beban mekanis pada motor akibat perubahan torsi yang mendadak.

4.4.3 Keberhasilan Eksekusi (*Goal Rate*)

Goal rate merupakan parameter keberhasilan robot dalam menyelesaikan seluruh rangkaian misi hingga bola masuk ke gawang. Pengujian dilakukan sebanyak 5 kali untuk melihat konsistensi sistem dalam berbagai repetisi.

Hasil pengujian menunjukkan bahwa sistem *optimized* memiliki tingkat keberhasilan yang lebih tinggi. Keunggulan ini didorong oleh kemampuan robot dalam menjaga orientasi terhadap bola dan gawang secara kontinu. Pada beberapa percobaan di sistem *baseline*, robot gagal menendang bola masuk ke gawang karena kehilangan posisi yang pas ke gawang saat melakukan manuver, yang mengakibatkan posisi robot menjadi tidak presisi saat akan menendang. Sebaliknya, prediksi posisi dari *Kalman Filter* memastikan robot tetap berada pada jalur gawang, sehingga posisi akhir robot sebelum melakukan tendangan selalu berada pada titik optimal untuk mencetak gol.

4.5 Pembahasan Umum dan Analisis Optimasi

Hasil eksperimen yang telah dipaparkan pada subbab sebelumnya menunjukkan sebuah pola yang jelas mengenai efektivitas optimasi pada sistem visi robot. Integrasi antara YOLOv8 dan *Kalman Filter* terbukti bukan sekadar penambahan algoritma, melainkan solusi teknis untuk mengatasi kendala perangkat keras laptop ASUS K401U yang memiliki keterbatasan sumber daya. Melalui

analisis mendalam, terdapat tiga pilar utama yang menjelaskan mengapa sistem *optimized* jauh lebih unggul dibandingkan sistem *baseline*.

Pertama, aspek efisiensi komputasi melalui strategi *frame skipping* menjadi kunci stabilitas sistem. Pada sistem *baseline*, beban kerja prosesor mencapai titik jenuh karena harus melakukan inferensi *deep learning* yang sangat berat di setiap *frame*. Hal ini mengakibatkan fenomena *bottleneck*, di mana *frame rate* menurun dan menyebabkan keterlambatan respon robot. Dengan menerapkan *Kalman Filter*, robot mampu "beristirahat" sejenak dari proses deteksi berat tanpa kehilangan informasi posisi objek. Prediksi matematis yang dijalankan saat *skip frame* jauh lebih ringan secara komputasi, sehingga rata-rata FPS dapat meningkat secara signifikan. Hal ini membuktikan bahwa keterbatasan spesifikasi perangkat keras dapat diatasi dengan manajemen beban kerja perangkat lunak yang cerdas.

Kedua, ketangguhan *tracking* terhadap oklusi menunjukkan bahwa robot kini memiliki "kesadaran spasial" yang lebih baik. Dalam dinamika pertandingan KRSBI yang sangat cepat, hilangnya visual bola akibat tertutup lawan atau guncangan kamera adalah hal yang tak terhindarkan. Sistem *baseline* yang hanya mengandalkan deteksi instan akan mengalami "kebutaan" sesaat, yang sering kali berujung pada kegagalan misi. Namun, penggunaan vektor keadaan (X) yang mencakup posisi dan kecepatan pada *Kalman Filter* memberikan kemampuan prediksi yang akurat. Robot tidak lagi hanya bereaksi terhadap apa yang dilihatnya sekarang, tetapi juga memperhitungkan ke mana objek akan bergerak berdasarkan momentum sebelumnya. Inilah yang menyebabkan *tracking loss count* menurun drastis pada sistem yang telah dioptimasi.

Terakhir, sinergi antara *filter* pelacakan dan algoritma *grid* menghasilkan stabilitas gerakan yang lebih halus. Masalah utama pada sistem lama adalah adanya *jitter* atau perubahan cepat pada perintah gerak akibat fluktuasi koordinat deteksi visual. Dengan adanya proses *smoothing* dari *Kalman Filter*, input koordinat yang diterima oleh algoritma *grid* menjadi lebih stabil dan terukur. Hal ini berdampak langsung pada efisiensi mekanis, roda robot tidak lagi menerima perintah gerak yang berubah-ubah secara mendadak, sehingga penggunaan energi lebih efisien dan risiko kerusakan pada motor DC akibat torsi yang fluktuatif dapat diminimalisir.

Secara keseluruhan, penelitian ini membuktikan bahwa optimasi sistem *tracking* menggunakan *Kalman Filter* mampu meningkatkan performa robot secara holistik. Keberhasilan dalam mempercepat waktu penyelesaian misi dan meningkatkan *goal rate* menegaskan bahwa robot kini lebih siap untuk menghadapi situasi kompetisi yang dinamis. Hasil ini sekaligus menjawab tantangan pada penelitian-penelitian sebelumnya di sekretariat ERC UNRI, dengan menawarkan sistem kendali yang tidak hanya cerdas dalam mengenali objek, tetapi juga andal dalam mempertahankan pelacakan (*tracking*) dan mengeksekusi gerakan.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian, pengujian, dan analisis yang telah dilakukan terhadap sistem visi robot KRSBI Beroda dengan integrasi YOLOv8 dan *Kalman Filter*, dapat disimpulkan bahwa pendekatan yang diusulkan berhasil meningkatkan performa sistem secara menyeluruh dibandingkan sistem *baseline*. Implementasi sistem *optimized* mampu mengatasi keterbatasan sumber daya komputasi pada perangkat yang digunakan dengan mendistribusikan beban pemrosesan secara lebih efisien, sehingga menghasilkan peningkatan rata-rata *frames per second* (FPS).

Selain itu, integrasi *Kalman Filter* terbukti efektif dalam meningkatkan ketahanan *tracking* terhadap gangguan visual, ditunjukkan dengan penurunan signifikan pada jumlah kehilangan tracking (*tracking loss count*) serta berkurangnya durasi kehilangan *tracking* saat terjadi oklusi sebagian maupun sementara. Dari sisi kendali robot, proses estimasi *Kalman Filter* posisi bola terhadap koordinat deteksi mampu mereduksi *jitter* pada pergerakan, menghasilkan transisi gerak yang lebih stabil dan linier. Dampaknya terlihat pada waktu penyelesaian misi yang lebih cepat serta peningkatan persentase keberhasilan mencetak gol (*goal rate*).

Secara keseluruhan, sinergi antara YOLOv8 sebagai detektor dan *Kalman Filter* sebagai estimator posisi berhasil mewujudkan sistem visi yang lebih stabil, responsif, dan andal dalam mendukung performa robot pada skenario pertandingan yang dinamis.

5.2 Saran

Untuk pengembangan lebih lanjut dalam meningkatkan performa robot di masa mendatang, terdapat beberapa aspek yang dapat dipertimbangkan. Dari sisi perangkat keras, penggunaan unit pemrosesan dengan GPU terdedikasi atau perangkat *edge computing* seperti NVIDIA Jetson berpotensi memberikan peningkatan signifikan pada kecepatan inferensi YOLOv8, sehingga sistem dapat

memproses lebih banyak *frame* tanpa perlu menerapkan *frame skipping*. Selain itu, penelitian selanjutnya dapat mengeksplorasi penggunaan metode *filtering* yang lebih kompleks seperti *Extended Kalman Filter* (EKF) atau *Unscented Kalman Filter* (UKF), khususnya untuk menangani dinamika pergerakan bola yang bersifat non-linier dan lebih kompleks dalam kondisi pertandingan sebenarnya. Pendekatan tersebut diharapkan mampu meningkatkan akurasi estimasi posisi sekaligus mempertahankan stabilitas sistem dalam skenario yang lebih dinamis.

DAFTAR PUSTAKA

Baskoro, G. Y., Afrisal, H., & Sofwan, A. (2022). Perancangan Sistem Deteksi Objek Berbasis Convolutional Neural Network Menggunakan Yolov4 Dan Opencv. *Transient: Jurnal Ilmiah Teknik Elektro*, *11*(4), 128–020637. https://ejournal3.undip.ac.id/index.php/transient

Egi, Y. (2022). *Basketball self training shooting posture recognition and trajectory estimation using computer vision and Kalman filter*. *73*, 19–27.

Farhan, T. M., & Candra, F. (2025). CNN-Based Ball and Goal Detection for KRSBI Robot with Omnidirectional Camera. *International Journal of Electrical, Energy and Power System Engineering*, *8*(2), 86–98. https://doi.org/10.31258/ijeepse.8.2.1-13

Firdaus, A. Z., & Lelono, D. (2025). *Sistem Klasifikasi Sampah Otomatis Berbasis Deteksi Objek Real-Time Pada Single Board Computer Dengan Algoritma YOLO*. *15*(1), 49–60. https://doi.org/10.22146/ijeis.104520

Hendrik, B., & Awal, H. (2023). Pengenalan Teknologi Robot Pada Anak Sekolah Dasar. *Jurmas Bangsa*, *1*(1), 46–52. https://doi.org/10.62357/jpb.v1i1.140

Jung, H., Kang, S., Kim, T., Kim, H., Klemove, H. L., & Korea, R. (2024). ConfTrack : Kalman Filter-based Multi-Person Tracking by Utilizing Confidence Score of Detection Box. *CVF Open Access*, 6583–6592.

Kusumoputro, B., Purnomo, M. H., Rochardjo, H. S. B., Prabowo, G., Purwanto, D., Mozef, E., Indrawanto, Mutijarsa, K., & Muis, A. (2024). *Pedoman Kontes Robot Indonesia (Kri) Pendidikan Tinggi Tahun 2024*.

Lee, Y. G. (2024). Confidence-Guided Frame Skipping to Enhance Object Tracking Speed. *Sensors*, *24*(24). https://doi.org/10.3390/s24248120

Louda, S., Karkar, N., & Seghir, F. (2023). *Autonomous Navigation of a Differential Mobile Robot using Depth Camera Sensor-based ROS*.

Miharja, G. P., Nugraha, D. A., & Aziz, A. (2025). Analisis Perbandingan Kinerja YOLO dan Camshift Dalam Pelacakan Objek Berbasis Video. *Jurnal Riset Mahasiswa Bidang Teknologi Informasi Volume*, *5*(Analisis Perbandingan Kinerja Yolo dan Camshift), 100–110.

Mohammed, H. R., & Hussain, Z. M. (2021). *Detection and Recognition of Moving Video Objects: Kalman Filtering with Deep Learning*. *12*, 154–157.

Nanda, B. M., Siregar, S., & Sani, M. I. (2023). *Implementasi Object Detection Pada Robot Sepak Bola Beroda Berbasis Kamera Omnidirectional Menggunakan OpenCV*. *9*(4), 2064–2068.

NumPy. (2025). *NumPy*. https://numpy.org/

Nuralim, J., Fath, N., Musafa, A., Sujono, & Broto, D. S. (2022). Perancangan Sistem Pendeteksian Obyek Bola Dengan Metode Framework YOLOv4. *Jurnal Maestro*, *5*(2), 289–294. https://jom.ft.budiluhur.ac.id/index.php/maestro/article/view/531

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). *ROS : An Open-source Robot Operating System*. *Figure 1*.

Ratna, S. (2020). Pengolahan Citra Digital Dan Histogram Dengan Phyton Dan Text Editor Phycharm. *Technologia: Jurnal Ilmiah*, *11*(3), 181. https://doi.org/10.31602/tji.v11i3.3294

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-time Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Ritonga, A. A., & Hasibuan, E. R. (2025). *Eksplorasi Masa Depan Robotika : Integrasi Kecerdasan Buatan dalam Sistem Automasi Adaptif Berbasis Lingkungan*. *1*, 387–393.

Rochardjo, H. S. B. (2024). *Sosialisasi KRSBI Beroda 2024*.

Romario, M. H., & Kadarina, T. M. (2020). *Sistem Hitung Dan Klasifikasi Objek Dengan Metode Convolutional Neural Network*. *11*(2), 108–114.

Ryu, S. E., & Chung, K. Y. (2021). Detection Model Of Occluded Object Based On Yolo Using Hard-example Mining And Augmentation Policy Optimization. *Applied Sciences (Switzerland)*, *11*(15). https://doi.org/10.3390/app11157093

Saputra, C. (2023). Implementasi Algoritma SIFT (Scale-Invariant Feature

Transform) Dan Algoritma Kalman Filter Dalam Mendeteksi Objek Bola. *Jurnal PROCESSOR*, *18*(1), 73–82. https://doi.org/10.33998/processor.2023.18.1.791

Saputra, F. B., Kallista, M., & Setianingsih, C. (2023). Deteksi social distancing dan penggunaan masker di restoran menggunakan algoritma Residual Network (ResNet). *EProceedings of Engineering*, *10*(1), 284–295.

Shofi, M., Basuki, B. M., & Habibi, A. (2023). Rancang Bangun Penendang Bola Pada Robot Soccer Unisma Menggunakan Solenoid. *Science Electro*, *16*(4), 1–7.

Sholehurrohman, R., Habibi, M. R., Ilman, I. S., Taufiq, R., & Muhaqiqin, M. (2023). Analisis Metode Kalman Filter, Particle Filter dan Correlation Filter Untuk Pelacakan Objek. *Komputika : Jurnal Sistem Komputer*, *12*(2), 21–28. https://doi.org/10.34010/komputika.v12i2.9567

Sugiyono. (2013). *Metode Penelitian Kuantitatif, Kualitatif dan R&D*.

Surya, M., Nehemia Toscany, A., Saputra, C., Pratama, Y., & Bustami, M. I. (2025). Penggunaan Yolo Untuk Deteksi Robot Dan Gawang Pada Robot Sepak Bola Beroda. *The Indonesian Journal of Computer Science*, *14*(1), 1055–1070. https://doi.org/10.33022/ijcs.v14i1.4575

Takura, K. E. I., Arita, Y. U. M. A. N., & Oaki, S. H. N. (2021). *Automatic pear and apple detection by videos using deep learning and a Kalman filter*. *4*(5), 1688–1695.

Taylor, L. E., Mirdanies, M., & Saputra, R. P. (2016). *OPTIMIZED OBJECT TRACKING TECHNIQUE USING KALMAN*. *07*, 57–66. https://doi.org/10.14203/j.mev.2016.v7.57-66

Ultralytics. (2025). *Kalman Filter (KF)*. https://www.ultralytics.com/glossary/kalman-filter-kf

Wakhidah, N., Pungkasanti, P. T., & Pinem, A. P. R. (2023). *Deteksi Objek menggunakan Deep Learning*. *9*(3), 465–470.

Widodo, Y. B., Sibuea, S., & Narji, M. (2024). Kecerdasan Buatan dalam Pendidikan: Meningkatkan Pembelajaran Personalisasi. *Jurnal Teknologi Informatika Dan Komputer*, *10*(2), 602–615.

https://doi.org/10.37012/jtik.v10i2.2324

Wijaya, G. F., & Yuniarto, D. (2024). Tinjauan Penerapan Machine Learning pada Sistem Rekomendasi Menggunakan Model Klasifikasi. *Populer: Jurnal Penelitian Mahasiswa*, *3*(4), 144–153. https://doi.org/10.58192/populer.v3i4.2798

Yaseen, M. (2024). *What Is Yolov8: An In-Depth Exploration Of The Internal Features Of The Next-Generation Object Detector*. *8*, 1–10.

Yuztiawan, F. R., & Utaminingrum, F. (2017). Implementasi Metode Kalman Filter Dan Model Yolov8N Untuk Fitur Human-Following Pada Kursi Roda Pintar. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *1*(1), 2548–2964.