TELEKONTRAN: Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan
Vol. 12, No. 1, April 2024
DOI : 10.34010/telekontran.v12i1.13323
p-ISSN : 2303 – 2901; e-ISSN : 2654 – 7384

Implementasi Integrasi dari *Computer vision* dan Kendali PID untuk Kendali Kecepatan Dinamis pada Robot *Line follower*

Integrated Implementation of Computer vision and PID Control for Dynamic Speed Control of Line follower Robot

Ardy Seto Priambodo[*], Aris Nasuha, Oktaf Agni Dhewa
Teknik Elektronika, Fakultas Vokasi, Universitas Negeri Yogyakarta
Email[*]: ardyseto@uny.ac.id

Abstrak - Penelitian ini bertujuan untuk mengembangkan sistem kendali kecepatan dinamis pada robot line follower dengan mengintegrasikan teknologi *computer vision* dan kendali PID. Tantangan utama dalam pengendalian robot line follower adalah menjaga kestabilan dan kecepatan robot saat menghadapi berbagai jenis belokan dan jalur yang kompleks dengan variasi pencahayaan. Penelitian ini mengusulkan penggunaan *computer vision* dengan metode thresholding *otsu binarization* untuk mendeteksi jalur dengan lebih akurat dan responsif, serta kendali PID untuk menyesuaikan kecepatan robot secara dinamis berdasarkan *distance* dan *angle error* yang terdeteksi. Metode penelitian melibatkan simulasi robot e-puck di lingkungan Webots, pembuatan algoritma deteksi jalur hitam dan perhitungan *error,* serta perancangan sistem kendali PID. Penggunaan *Otsu* binary thresholding untuk deteksi garis menunjukkan bahwa sistem ini mampu mengatasi variasi pencahayaan dengan baik. Hasil pengujian menunjukkan bahwa pada Arena 1, waktu tempuh dengan kecepatan dasar tetap adalah 60.67 detik, sedangkan dengan PID adalah 50.272 detik dan fuzzy 52.608 detik. Pada Arena 2, dengan kecepatan tetap gagal menyelesaikan arena, sementara dengan PID adalah 91.04 detik dan fuzzy 94.048 detik, menunjukkan pengurangan PID lebih baik dibandingkan lainnya. Dengan demikian, sistem kendali ini lebih efektif dalam menjaga robot tetap pada jalur yang diinginkan, mengurangi penyimpangan dan meningkatkan akurasi *tracking* jalur.

Kata kunci: Robot *Line follower*, Kendali PID, *Computer vision*, Simulasi Webots, *Otsu Thresholding*.

Abstract - *This research aims to develop a dynamic speed control system for line follower robots by integrating computer vision technology and PID control. The main challenge in controlling a line follower robot is maintaining the stability and speed of the robot when facing various types of turns and complex paths with variations in lighting. This research proposes the use of computer vision with the Otsu binarization thresholding method to detect paths more accurately and responsively, as well as PID control to dynamically adjust the robot's speed based on the detected distance and angle error. The research method involves simulating an e-puck robot in the Webots environment, creating black line detection algorithms and error calculations, as well as designing a PID control system. The use of Otsu binary thresholding for line detection shows that this system is able to handle lighting variations well. The test results show that in Arena 1, the travel time with a fixed base speed is 60.67 seconds, while with PID it is 50.272 seconds and fuzzy 52.608 seconds. In Arena 2, with constant speed it failed to complete the arena, while with PID it was 91.04 seconds and fuzzy 94.048 seconds, showing better PID reduction than the others. Thus, this control system is more effective in keeping the robot on the desired path, reducing deviations and increasing path tracking accuracy*.

Keywords*: Line follower Robot, PID Control, Computer vision, Webots Simulation, Otsu Thresholding.*

I. PENDAHULUAN

Robotika adalah salah satu cabang ilmu pengetahuan yang mengalami perkembangan pesat, dengan berbagai aplikasi mulai dari industri manufaktur hingga dunia pendidikan. Robot-robot ini dirancang untuk menjalankan tugas-tugas tertentu secara otomatis, menggantikan atau melengkapi peran manusia [1]. Salah satu jenis robot yang menarik perhatian banyak peneliti adalah robot *line follower*. Robot *line follower* adalah robot yang dirancang untuk mengikuti jalur tertentu yang biasanya ditandai dengan garis di atas

permukaan [2]. Aplikasi robot *line follower* sangat luas, termasuk dalam dunia pendidikan di mana robot ini sering digunakan sebagai alat bantu pembelajaran untuk memperkenalkan konsep-konsep dasar robotika dan pemrograman kepada siswa [3]. Selain itu, dalam industri, robot *line follower* digunakan untuk mengotomatisasi proses transportasi material di pabrik atau gudang, meningkatkan efisiensi dan mengurangi kesalahan manusia [4]. Keberhasilan implementasi robot *line follower* dalam berbagai aplikasi ini menunjukkan potensi besar yang dimiliki oleh teknologi ini untuk memecahkan berbagai masalah praktis.

Computer vision adalah bidang ilmu yang berfokus pada pengolahan dan analisis gambar untuk memperoleh informasi yang dapat digunakan oleh sistem komputer [5]. Perkembangan pesat dalam bidang ini didorong oleh kemajuan teknologi komputasi yang memungkinkan pengolahan data dalam jumlah besar dan kecepatan tinggi. Teknologi *computer vision* telah diterapkan dalam berbagai bidang, mulai dari pengenalan wajah dan objek dalam keamanan, diagnosis medis melalui analisis gambar radiologi, hingga navigasi otonom dalam kendaraan tanpa pengemudi [8]. Dalam dunia robotika, *computer vision* memainkan peran penting dalam memberikan kemampuan visual kepada robot, memungkinkan mereka untuk memahami dan berinteraksi dengan lingkungan sekitarnya secara lebih cerdas. Dengan kemampuan ini, robot dapat melakukan tugas-tugas kompleks seperti pengenalan objek, penghindaran rintangan, dan navigasi yang lebih akurat dan efisien.

Robot *line follower* secara spesifik menggunakan sensor untuk mendeteksi jalur yang harus diikuti. Sensor yang paling umum digunakan adalah sensor inframerah yang mampu mendeteksi perbedaan warna antara garis dan permukaan [9]. Meskipun sensor ini efektif dalam kondisi ideal, mereka sering menghadapi masalah dalam kondisi jalur yang kompleks. Misalnya, pada jalur dengan tikungan tajam atau persimpangan, sensor inframerah mungkin tidak dapat memberikan informasi yang akurat, menyebabkan robot keluar dari jalur. Selain itu, variasi dalam pencahayaan lingkungan juga dapat mempengaruhi kinerja sensor inframerah, mengurangi keandalannya. Oleh karena itu, ada kebutuhan untuk meningkatkan sistem deteksi jalur agar robot *line follower* dapat beroperasi dengan lebih baik dalam berbagai kondisi.

Pengendalian robot, termasuk robot *line follower*, umumnya dilakukan dengan menggunakan berbagai metode kendali seperti PID (*Proportional-Integral-Derivative*), *fuzzy logic*, dan lainnya [6]. Kendali PID adalah salah satu metode yang paling populer dan telah terbukti efektif dalam berbagai aplikasi [7]. Metode ini bekerja dengan menghitung tiga komponen utama: proporsional, integral, dan derivatif, untuk menentukan tindakan korektif yang diperlukan agar robot dapat mengikuti jalur dengan tepat. Meskipun efektif, kendali PID masih menghadapi tantangan dalam kondisi jalur yang dinamis dan kompleks. Alternatif lain seperti kendali fuzzy juga telah digunakan, yang lebih adaptif terhadap perubahan lingkungan, tetapi memerlukan perancangan yang lebih kompleks.

Robot *line follower* sering menghadapi tantangan yang signifikan saat berbelok pada kecepatan tinggi. Ketika robot bergerak dengan kecepatan tinggi, momentum yang dihasilkan membuat robot sulit untuk mengikuti tikungan tajam dengan presisi [16]. Hal ini menyebabkan robot sering kali keluar dari jalur yang telah ditentukan. Kesulitan ini terutama disebabkan oleh inersia yang menghalangi robot untuk beradaptasi dengan perubahan arah secara cepat. Selain itu, pada kecepatan tinggi, sensor mungkin tidak memiliki cukup waktu untuk memberikan umpan balik yang diperlukan kepada sistem kontrol untuk melakukan penyesuaian yang diperlukan secara real-time.

Masalah pencahayaan juga merupakan tantangan signifikan dalam penggunaan kamera untuk deteksi garis pada robot *line follower* [14]. Kondisi pencahayaan yang bervariasi, seperti bayangan, cahaya yang terlalu terang, atau perubahan intensitas cahaya secara tiba-tiba, dapat mempengaruhi kemampuan kamera untuk mendeteksi garis dengan akurat. Ketidakstabilan pencahayaan ini dapat menyebabkan kesalahan dalam interpretasi gambar, yang pada gilirannya dapat mengarahkan robot keluar jalur. Sensor kamera yang bergantung pada visibilitas garis sering kali tidak dapat mengatasi variasi pencahayaan yang dinamis, sehingga mempengaruhi keandalan dan konsistensi dari sistem navigasi robot.

Penelitian sebelumnya telah mengusulkan berbagai solusi untuk mengatasi masalah ini. Salah satu pendekatan yang umum adalah dengan

mengimplementasikan algoritma kontrol adaptif yang dapat menyesuaikan kecepatan robot berdasarkan kondisi lintasan [15]. Misalnya, penggunaan kontrol PID dan metode *fuzzy logic* telah terbukti efektif dalam meningkatkan respons robot terhadap perubahan arah pada kecepatan tinggi. Selain itu, beberapa studi juga mengeksplorasi penggunaan sensor tambahan seperti sensor *gyro*atau akselerometer untuk memberikan informasi tambahan mengenai orientasi dan kecepatan robot [13], sehingga memungkinkan sistem kontrol untuk membuat penyesuaian yang lebih tepat waktu dan akurat.

Untuk masalah pencahayaan, beberapa penelitian telah mengusulkan penggunaan teknik pengolahan citra yang lebih canggih, seperti algoritma penyesuaian kontras otomatis dan filter adaptif yang dapat meningkatkan kualitas gambar yang diterima oleh kamera [17]. Pendekatan lain termasuk penggunaan pencahayaan tambahan yang terpasang pada robot untuk memastikan konsistensi cahaya yang diterima oleh kamera, serta penggunaan sensor alternatif seperti sensor inframerah yang kurang dipengaruhi oleh perubahan kondisi pencahayaan. Studi-studi ini menunjukkan bahwa kombinasi dari teknik kontrol yang canggih dan sensor yang lebih andal dapat secara signifikan meningkatkan kinerja robot *line follower* dalam berbagai kondisi lingkungan.

Tentunya dengan penambahan sensor membuat robot menjadi kurang praktis dan memakan sumber daya yang lebih besar sehingga pemanfaatan sensor kamera saja diharapkan mampu menyelesaikan masalah ini secara optimal disertai dengan metode yang sesuai. Penelitian ini mengusulkan pemanfaatan teknologi *computer vision* dalam pengendalian robot *line follower* dengan metode kendali PID untuk pengaturan kecepatan base dan delta sehingga kecepatannya dinamis mengikuti dengan jalur yang ada. Dengan menggunakan *computer vision*, robot dapat mendeteksi jalur dengan lebih akurat dan responsif terhadap perubahan jalur. Informasi visual yang diperoleh dari kamera akan digunakan untuk menghitung *error*, yang kemudian akan diproses oleh algoritma kendali PID untuk menentukan kecepatan robot. Integrasi ini diharapkan dapat mengatasi keterbatasan yang ada pada sistem yang hanya menggunakan sensor inframerah atau metode kendali tunggal. Dengan pendekatan ini, robot *line follower* diharapkan dapat beroperasi dengan lebih stabil dan efisien dalam berbagai kondisi jalur.

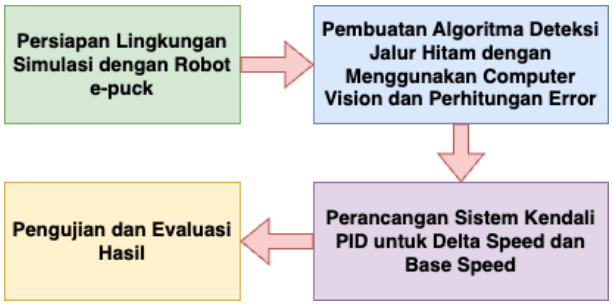
Manfaat dan harapan dari penelitian ini adalah untuk menciptakan sistem kendali kecepatan dinamis yang lebih cerdas dan adaptif bagi robot *line follower*. Dengan integrasi *computer vision* dan kendali PID, robot diharapkan mampu memberikan respon yang lebih cepat dan akurat terhadap perubahan jalur, meningkatkan stabilitas dan performa keseluruhan. Penelitian ini juga bertujuan untuk memberikan kontribusi signifikan dalam bidang robotika dan sistem kendali, menawarkan solusi praktis yang dapat langsung diterapkan dalam industri maupun pendidikan. Selain itu, hasil penelitian ini diharapkan dapat menjadi dasar bagi pengembangan lebih lanjut dalam integrasi teknologi *computer vision* dengan metode kendali lainnya, membuka peluang baru untuk inovasi dalam pengendalian robot.

II. METODOLOGI

Penelitian ini dilakukan melalui beberapa tahapan yang sistematis untuk mengembangkan dan menguji sistem kendali kecepatan dinamis pada robot *line follower* dengan integrasi teknologi *computer vision* dan kendali PID. Berikut adalah tahapan-tahapan yang dilakukan dalam penelitian ini seperti yang terlihat prosesnya pada **Gambar 1** dengan penjelasan, Persiapan Lingkungan Simulasi dengan Robot e-puck: Pada tahap ini, persiapan dilakukan menggunakan aplikasi Webots. Robot e-puck yang dilengkapi dengan kamera akan digunakan dalam simulasi. Beberapa jalur akan dibuat untuk menguji robot dalam berbagai kondisi jalur yang berbeda. Pembuatan Algoritma Deteksi Jalur Hitam dengan Menggunakan *Computer vision* dan Perhitungan *Error*:

Pada tahap ini, teknologi *computer vision* akan digunakan untuk mendeteksi jalur hitam. Proses deteksi ini melibatkan pengolahan citra dari kamera robot untuk mengidentifikasi posisi jalur. Dua titik referensi akan digunakan untuk menghasilkan dua nilai *error*, yang akan digunakan dalam perancangan sistem kendali. Perancangan Sistem Kendali PID untuk *Delta speed* dan Base Speed: Berdasarkan nilai *error* yang diperoleh dari deteksi jalur, sistem kendali PID akan dirancang untuk menghasilkan nilai *base speed* dan *delta speed*. Kendali PID akan digunakan untuk mengatur kecepatan robot agar tetap stabil dan mengikuti jalur dengan tepat. Pengujian dan Evaluasi Hasil: Tahap terakhir melibatkan pengujian dan evaluasi sistem kendali yang telah dirancang. Robot akan diuji dalam berbagai

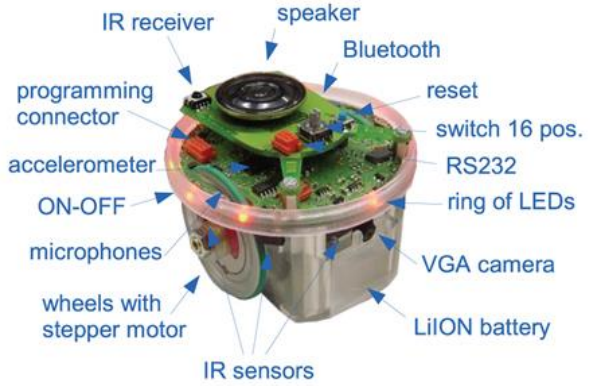
kondisi jalur di lingkungan simulasi, dan kinerjanya akan dievaluasi berdasarkan keakuratan dalam mengikuti jalur, responsivitas terhadap perubahan jalur, dan stabilitas kecepatan.



Gambar 1. Metode Pelaksanaan Penelitian

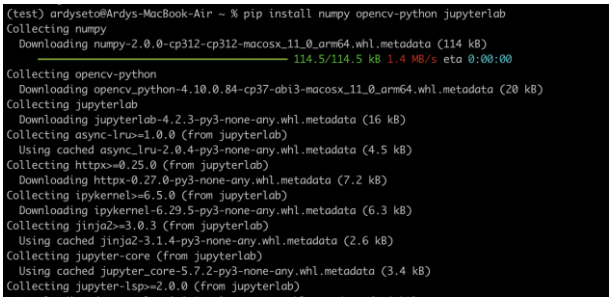
A. Persiapan Lingkungan Simulasidengan Robot E-Puck pada Aplikasi Webots

Tahap pertama dalam penelitian ini adalah persiapan lingkungan simulasi menggunakan aplikasi Webots. Pada tahap ini, robot e-puck dipilih sebagai platform robotika yang akan digunakan karena kemudahan dalam penggunaannya serta *open source* [12]. Bentuk dan komponen yang ada pada robot E-Puck terlihat pada **Gambar 2** [10]. Robot e-puck dilengkapi dengan 2 motor penggerak dan berbagai sensor, termasuk kamera, yang sangat berguna untuk deteksi jalur berbasis *computer vision*. Persiapan lingkungan simulasi melibatkan dua aspek utama. Pertama, pembuatan beberapa arena dengan berbagai variasi bentuk jalur. Jalur-jalur ini akan terdiri dari garis hitam di atas permukaan putih, dengan berbagai bentuk dan kompleksitas untuk menguji performa robot dalam kondisi yang bervariasi. Desain jalur meliputi lintasan lurus, tikungan tajam, dan beberapa variasi lainnya, yang bertujuan untuk mengevaluasi keakuratan dan responsivitas sistem deteksi jalur serta kendali yang akan diimplementasikan.



Gambar 2. Robot E-Puck dan Komponennya

Aspek kedua melibatkan persiapan terkait pembuatan program untuk simulasi. Langkah ini mencakup instalasi Python beserta *library* yang dibutuhkan untuk proses *computer vision*, seperti numpy, matplotlib, dan opencv [11] yang prosesnya terlihat pada **Gambar 3**. *Library* numpy akan digunakan untuk operasi numerik yang efisien, matplotlib untuk visualisasi data, dan opencv untuk pemrosesan citra. Setelah semua *library* terinstal, program simulasi akan dikembangkan untuk mengintegrasikan deteksi jalur menggunakan kamera robot e-puck dan implementasi algoritma kendali.



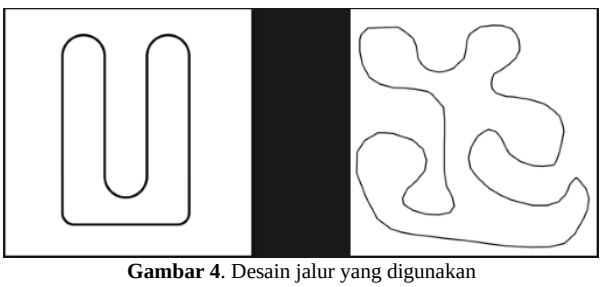
Gambar 3. Instalasi *library* python

Kalibrasi awal akan dilakukan untuk memastikan semua sensor berfungsi dengan baik dan mendapatkan pembacaan sensor yang akurat. Dengan persiapan ini, lingkungan simulasi di Webots diharapkan dapat memberikan kondisi yang mendekati nyata, memungkinkan peneliti untuk melakukan eksperimen dan evaluasi secara komprehensif sebelum aplikasi ke dunia nyata. Spesifikasi komputer yang digunakan untuk simulasi tertera pada **Tabel I**.

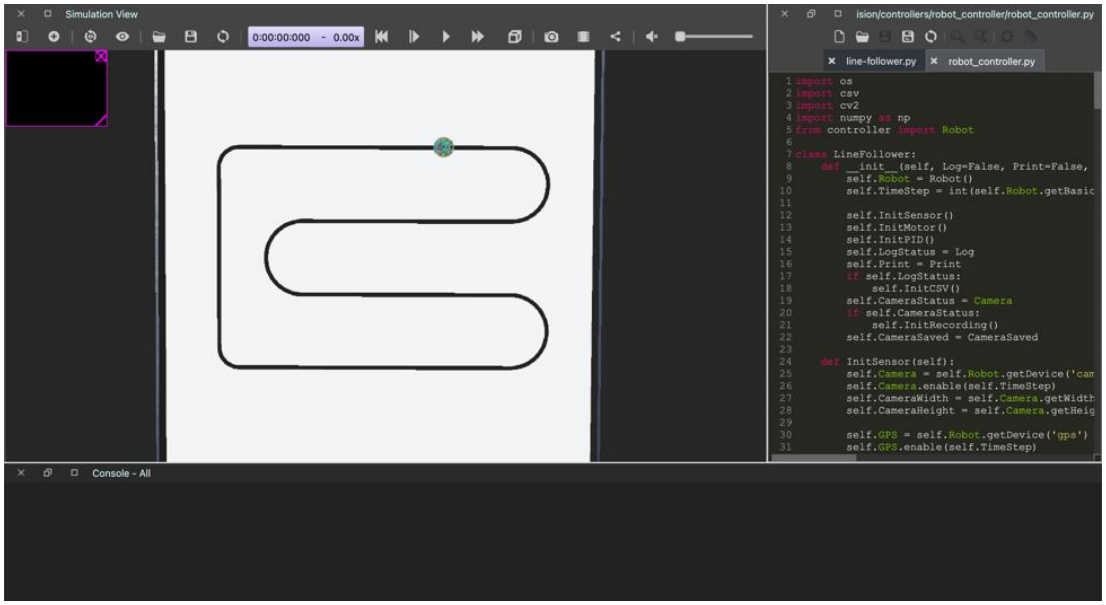
Tabel I. Tabel hasil pengujian

Komponen	Spesifikasi
CPU	Intel Processor i5 12400F
Kartu Grafis	NVidia RTX 3060 12GB
RAM	DDR4 32GB
Penyimpanan	SSD NVMe Gen 4 256GB
Monitor	24 Inch Full HD

Gambaran dari salah satu arena yang digunakan terlihat pada **Gambar 4** dan tampilan dari aplikasi webots dengan robot epuck dan desain jalur yang digunakan ditunjukkan pada **Gambar 5**.



Gambar 4. Desain jalur yang digunakan



Gambar 5. Perancangan Simulasi Robot *Line follower* pada Webots

B. Pembuatan Algoritma Deteksi Jalur Hitam dengan *Computer vision* dan Perhitungan *Error*

Tahapan kedua dalam penelitian ini adalah pembuatan algoritma untuk deteksi jalur hitam menggunakan teknologi *computer vision* dan perhitungan *error* yang terkait. Proses ini terdiri dari dua langkah utama: deteksi jalur dan perhitungan *error*. Pada langkah pertama, algoritma deteksi jalur memproses gambar yang diambil oleh kamera robot untuk mengidentifikasi posisi jalur hitam. Setelah jalur terdeteksi, langkah kedua melibatkan perhitungan *error*, yaitu deviasi dari jalur yang diinginkan, yang digunakan sebagai umpan balik untuk sistem kendali robot.

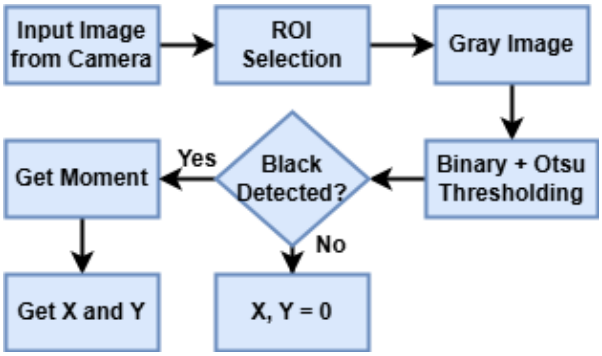
Deteksi jalur hitam dimulai dengan menangkap gambar jalur menggunakan kamera yang terpasang pada robot. Gambar yang diambil kemudian diproses dengan memotong area tertentu menjadi *region of interest* (ROI), yang berfokus pada bagian gambar di mana jalur hitam kemungkinan besar berada. ROI ini diubah menjadi gambar *grayscale* untuk menyederhanakan proses analisis.

Langkah selanjutnya adalah pemberian *threshold* pada gambar *grayscale* untuk mengidentifikasi jalur hitam dengan lebih jelas, menghasilkan gambar biner di mana piksel hitam menunjukkan jalur dan piksel putih menunjukkan latar belakang.

Untuk memastikan bahwa nilai *threshold* bersifat dinamis dan sesuai dengan pencahayaan yang ada, digunakan teknik *Otsu binarization*. Teknik ini secara otomatis menentukan nilai *threshold* optimal dengan meminimalkan varians intra-kelas (intra-class variance) dari piksel yang ada. Dengan menggunakan *Otsu binarization*, algoritma dapat beradaptasi dengan berbagai kondisi pencahayaan, sehingga deteksi jalur hitam tetap akurat meskipun terjadi perubahan pencahayaan pada lingkungan sekitar.

Setelah gambar biner diperoleh, teknik pengolahan citra seperti momen kemudian digunakan untuk menghitung pusat massa jalur hitam, yang diwakili oleh koordinat pusat horizontal dan vertikal. Pusat massa ini ditandai pada gambar untuk visualisasi, memastikan bahwa jalur terdeteksi dengan benar. Proses dari deteksi ini digambarkan pada **Gambar 6**.

Dengan mengimplementasikan *Otsu binarization*, algoritma tidak lagi bergantung pada nilai *threshold* tetap yang mungkin tidak optimal di berbagai kondisi pencahayaan. Sebaliknya, algoritma ini mampu menyesuaikan nilai *threshold* secara otomatis, meningkatkan keandalan dan akurasi deteksi jalur hitam.

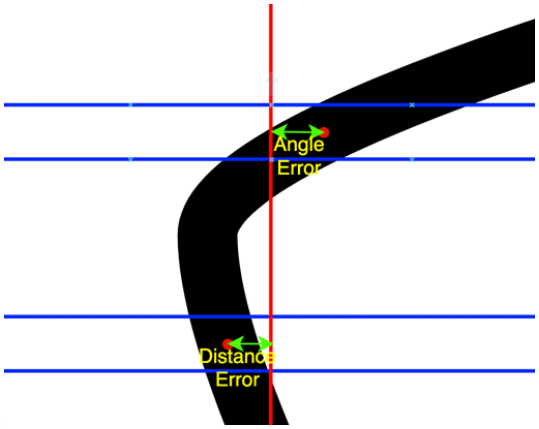


Gambar 6. Alur Program Deteksi Jalur

Langkah selanjutnya adalah perhitungan *error*, yang menggunakan nilai referensi dari deteksi jalur untuk menghitung deviasi dari jalur yang diinginkan. *Error* dihitung sebagai selisih antara koordinat pusat massa jalur yang terdeteksi (posisi aktual) dan posisi referensi ideal (posisi target). Persamaan yang digunakan untuk menghitung *error* adalah:

$$Error\ X = Aktual\ X - Target\ X \ldots. (1)$$

Error ini kemudian digunakan sebagai umpan balik untuk sistem kendali, memungkinkan robot untuk melakukan koreksi yang diperlukan untuk tetap mengikuti jalur dengan benar. Dengan menghitung *error* ini secara terus-menerus, sistem kendali dapat menyesuaikan gerakan robot untuk menjaga posisinya tetap pada jalur yang diinginkan, meningkatkan akurasi dan stabilitas pergerakan robot. Dalam proses perhitungan *error* ini, terdapat dua jenis *error* yang dihitung: *distance error* dan *angle error*. *Distance error* adalah perbedaan posisi horizontal antara pusat jalur yang terdeteksi dan posisi referensi yang diinginkan, yang digunakan untuk umpan balik kendali dalam mengatur *delta speed* (perubahan kecepatan). *Angle error*, di sisi lain, adalah perbedaan sudut antara orientasi robot dan jalur yang diinginkan, yang digunakan untuk umpan balik kendali dalam mengurangi *base speed* (kecepatan dasar). Dengan menghitung dan memanfaatkan kedua *error* ini, sistem kendali dapat melakukan penyesuaian yang lebih tepat, memastikan robot tetap mengikuti jalur dengan akurasi dan stabilitas yang tinggi. Ilustarasi dari perhitungan *error* ini terlihat pada **Gambar 7**.



Gambar 7. Perhitungan *Distance Error* dan *Angle Error*

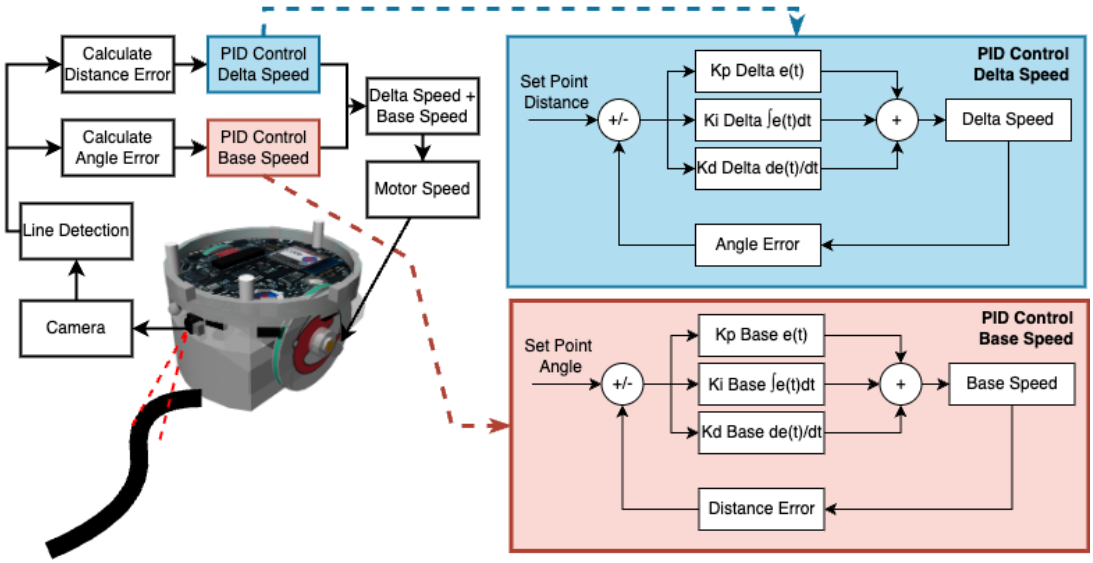
C. Perancangan Sistem Kendali PID untuk *base speed* dan *delta speed*

Tahapan ketiga dalam penelitian ini adalah perancangan sistem kendali PID untuk mengatur kecepatan dasar (base speed) dan perubahan kecepatan (*delta speed*) pada robot *line follower*. Kendali PID (*Proportional-Integral-Derivative*) adalah metode pengendalian yang sangat efektif dalam mengendalikan sistem dinamis dengan menggunakan tiga komponen utama: proporsional, integral, dan derivatif. Dalam konteks penelitian ini, sistem kendali PID dirancang untuk mengontrol dua aspek utama dari pergerakan robot: kecepatan dasar berdasarkan *error* sudut dan perubahan kecepatan berdasarkan *error* jarak.

Pertama, kendali PID untuk kecepatan dasar dirancang untuk menyesuaikan kecepatan robot berdasarkan *error* sudut, yaitu perbedaan antara orientasi robot dan jalur yang diinginkan. Proses ini melibatkan penghitungan tiga komponen proporsional, integral dan *derivative*. Kombinasi dari ketiga komponen ini menghasilkan sinyal kontrol yang digunakan untuk mengurangi kecepatan dasar robot guna menjaga stabilitas saat mengikuti jalur.

Kedua, kendali PID untuk perubahan kecepatan dirancang untuk mengatur kecepatan robot berdasarkan *error* jarak, yaitu perbedaan posisi horizontal antara pusat jalur yang terdeteksi dan posisi referensi yang diinginkan. Sama seperti kendali PID untuk kecepatan dasar, komponen proporsional dalam kendali perubahan kecepatan memberikan respons yang sebanding dengan besarnya *error* jarak saat ini, komponen integral mengakumulasi *error* jarak dari waktu ke waktu, dan komponen derivatif mempertimbangkan laju perubahan *error* jarak. Kombinasi dari ketiga komponen ini menghasilkan sinyal kontrol yang digunakan untuk menyesuaikan perubahan kecepatan robot agar tetap mengikuti jalur dengan akurasi yang tinggi.

Dalam implementasinya, kedua kendali PID ini bekerja secara bersamaan. Sistem kendali pertama menghitung kecepatan dasar robot berdasarkan *error* sudut, sementara sistem kendali kedua menghitung perubahan kecepatan berdasarkan *error* jarak. Hasil dari kedua kendali ini digabungkan untuk menghasilkan kecepatan akhir robot yang optimal. Dengan demikian, sistem kendali PID yang dirancang ini memungkinkan robot untuk menyesuaikan gerakannya secara dinamis, menjaga posisinya tetap pada jalur yang diinginkan, serta meningkatkan akurasi dan stabilitas pergerakan robot dalam berbagai kondisi jalur. Gambaran dari sistem perancangan ini terlihat pada **Gambar 8**.



Gambar 8. Perancangan Kendali PID untuk Delta Speed dan Base Speed

D. Pengujian dan Evaluasi Hasil

Tahapan keempat dalam penelitian ini adalah pengujian dan evaluasi hasil dari sistem kendali yang telah dirancang. Pengujian dilakukan untuk memastikan bahwa algoritma deteksi jalur dan sistem kendali PID dapat berfungsi dengan baik dalam berbagai kondisi jalur. Pengujian dilakukan dengan menggunakan beberapa bentuk tikungan dalam jalur, termasuk tikungan tajam, tikungan melengkung, dan persimpangan. Setiap jenis tikungan memberikan tantangan yang berbeda bagi robot *line follower*, dan penting untuk menguji bagaimana sistem kendali dapat mengatasi berbagai jenis tikungan ini. Data tentang posisi robot, kecepatan, dan *error* dikumpulkan selama pengujian untuk dianalisis lebih lanjut. Hasil pengujian dibandingkan dengan kinerja robot yang menggunakan kecepatan dasar yang tetap. Dalam skenario ini, kecepatan dasar robot tidak disesuaikan secara dinamis berdasarkan *error* sudut atau jarak. Perbandingan ini penting untuk menunjukkan efektivitas sistem kendali PID yang dinamis.

Evaluasi hasil dilakukan dengan menganalisis respon transien dari sistem kendali. Respon transien adalah respons sistem terhadap perubahan mendadak, seperti saat robot memasuki tikungan tajam. Analisis ini mencakup pengukuran waktu respon, *overshoot*, dan *settling time*. Respon transien yang baik menunjukkan bahwa sistem kendali dapat menyesuaikan kecepatan dan posisi robot dengan cepat dan akurat tanpa menimbulkan osilasi atau ketidakstabilan. Evaluasi ini menunjukkan bahwa sistem kendali PID dinamis yang dirancang dapat memberikan respon yang lebih cepat dan stabil dibandingkan dengan sistem kendali dengan kecepatan dasar tetap, terutama dalam menghadapi kondisi jalur yang kompleks. Hasil evaluasi ini menunjukkan bahwa sistem kendali PID dinamis mampu meningkatkan kinerja robot *line follower*, memberikan solusi yang lebih adaptif dan efektif untuk pengendalian robot otonom.

III. HASIL DAN PEMBAHASAN

A. Pengujian Pembacaan Jalur Hitam pada berbagai kondisi pencahayaan

Pada pengujian pembacaan jalur hitam untuk robot *line follower* dengan menggunakan kamera, proses deteksi dimulai dengan tahap pengambilan gambar dari kamera yang dipasang pada robot. Gambar asli yang diperoleh dari kamera (ditunjukkan pada **Gambar 9**) adalah representasi visual dari jalur yang akan diikuti oleh robot. Kamera menangkap seluruh area di depan robot, yang mencakup jalur hitam dan area sekitarnya. Langkah pertama ini sangat penting untuk memastikan bahwa robot memiliki informasi visual yang cukup untuk menentukan jalur yang harus diikuti.

Tahap kedua adalah pemilihan *Region of interest* (ROI) dari gambar asli. ROI adalah bagian dari gambar yang berisi informasi paling relevan untuk deteksi jalur hitam. Pemilihan ROI dilakukan untuk mengurangi beban komputasi dan meningkatkan efisiensi proses pengolahan citra. **Gambar 10** menunjukkan ROI yang telah dipilih, yang fokus pada bagian tengah jalur di mana jalur

hitam kemungkinan besar berada. Pemilihan ROI membantu menyaring bagian gambar yang tidak penting sehingga proses analisis dapat dilakukan lebih cepat dan akurat.

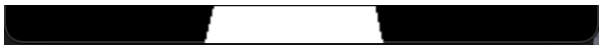
Gambar 9. Gambar original tangkapan kamera

Gambar 10. Hasil gambar dari *Region of interest* yang digunakan

Setelah ROI dipilih, gambar tersebut diubah menjadi gambar skala abu-abu seperti yang ditunjukkan pada **Gambar 11**. Konversi ke skala abu-abu dilakukan untuk menyederhanakan data yang akan diproses. Gambar skala abu-abu menghilangkan informasi warna yang tidak diperlukan dan menyoroti perbedaan intensitas cahaya, membuat jalur hitam lebih menonjol dibandingkan dengan latar belakang. Tahap ini penting karena mempermudah proses selanjutnya yaitu thresholding biner.

Gambar 11. Hasil gambar proses grayscale

Proses thresholding biner adalah langkah berikutnya, di mana gambar skala abu-abu diubah menjadi gambar biner. Dalam gambar biner, setiap piksel dikonversi menjadi hitam atau putih berdasarkan nilai ambang tertentu. Proses ini memisahkan jalur hitam dari latar belakang, memungkinkan deteksi yang lebih jelas dan akurat. Setelah thresholding, momen dari gambar biner dihitung untuk menentukan titik tengah dari jalur hitam. **Gambar 12** menunjukkan hasil perhitungan momen dengan penandaan titik merah sebagai set poin. Titik merah ini merupakan pusat dari jalur hitam yang akan digunakan sebagai acuan navigasi robot.

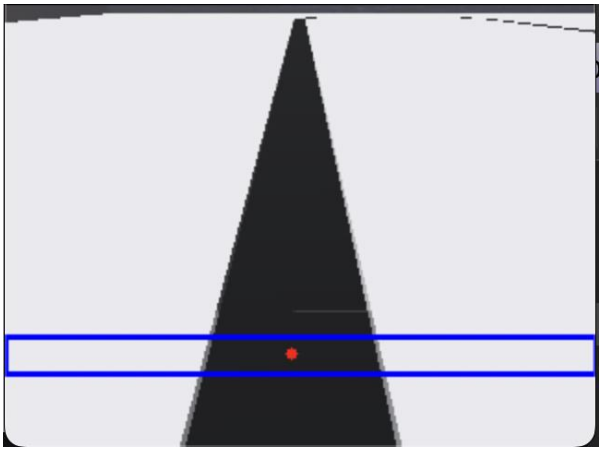


Gambar 12. Hasil thresholding

Langkah terakhir adalah mengembalikan hasil deteksi ke gambar asli untuk verifikasi visual dan penyesuaian lebih lanjut. **Gambar 13** menunjukkan gambar asli dengan tambahan hasil deteksi berupa titik merah dan garis pembatas. Penambahan ini memungkinkan sistem untuk memverifikasi apakah deteksi jalur hitam sudah akurat dan apakah robot sudah berada di jalur yang benar.

Gambar 13. Hasil moment untuk mendapatkan set point (tiitk berwarna merah)

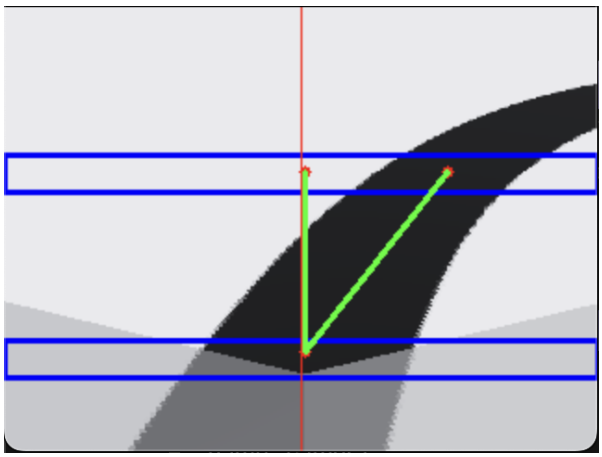
Dengan menggabungkan hasil deteksi ke gambar asli, operator dapat melakukan penyesuaian pada algoritma pengendalian robot jika diperlukan. Keseluruhan proses ini menunjukkan bagaimana teknik pengolahan citra digunakan untuk meningkatkan kemampuan navigasi robot *line follower* dalam mengikuti jalur hitam dengan presisi yang tinggi yang terlihat hasilnya pada **Gambar 14**.



Gambar 14. Hasil deteksi garis

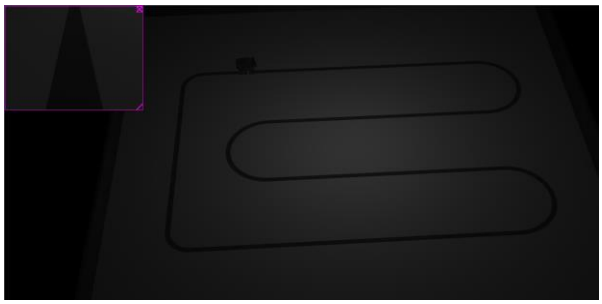
Hasil perhitungan *error* sudut (*angle error*) dan *error* jarak (*distance error*) ditampilkan pada gambar kedua. Dalam gambar tersebut, titik-titik merah menunjukkan lokasi pusat jalur hitam yang terdeteksi pada dua posisi berbeda. Garis hijau menunjukkan deviasi sudut yang terjadi antara jalur ideal dan jalur aktual yang diikuti oleh robot. Garis vertikal merah menandai jalur ideal yang seharusnya diikuti oleh robot. Perhitungan menunjukkan bahwa terdapat deviasi sudut yang cukup signifikan, yang menyebabkan robot tidak mengikuti jalur hitam dengan presisi yang diinginkan. *Error* jarak juga diukur sebagai jarak lateral antara pusat jalur ideal dan pusat jalur yang terdeteksi. Hasil ini ditunjukkan pada **Gambar 15** yang kemudian digunakan untuk melakukan penyesuaian pada kontroler robot, dengan tujuan

mengurangi *error* dan memastikan robot dapat mengikuti jalur hitam dengan akurasi yang lebih tinggi.

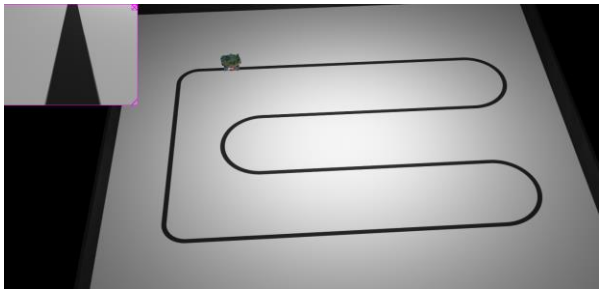


Gambar 15. Hasil deteksi *angle error* dan *distance error*

Pengujian pembacaan hasil deteksi garis juga dilakukan pada kondisi pencahayaan dengan nilai intensitas 0.1 hingga 1.0. Gambaran pencahayaan yang digunakan terlihat pada **Gambar 16** dan **Gambar 17**. Terlihat perbedaan ketika mendapatkan intensitas pencahayaan yang berbeda dimana intensitas cahaya 0.1 lebih gelap dibandingkan intensitas cahaya 1.0.

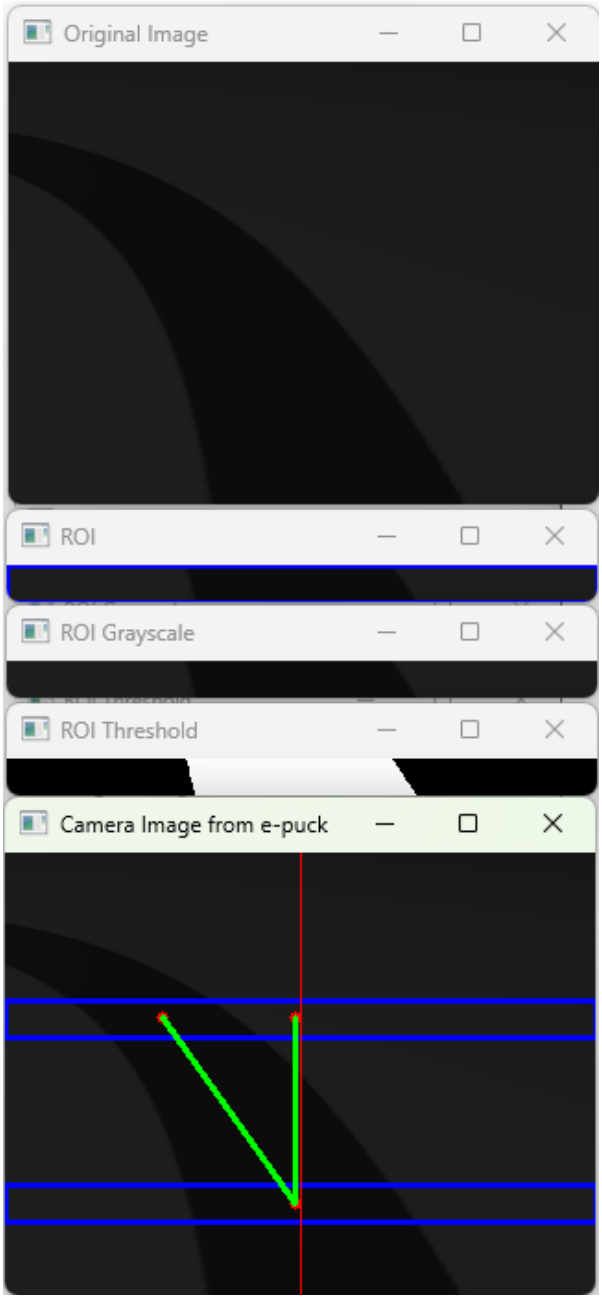


Gambar 16. Pengujian dengan Intensitas Cahaya 0.1

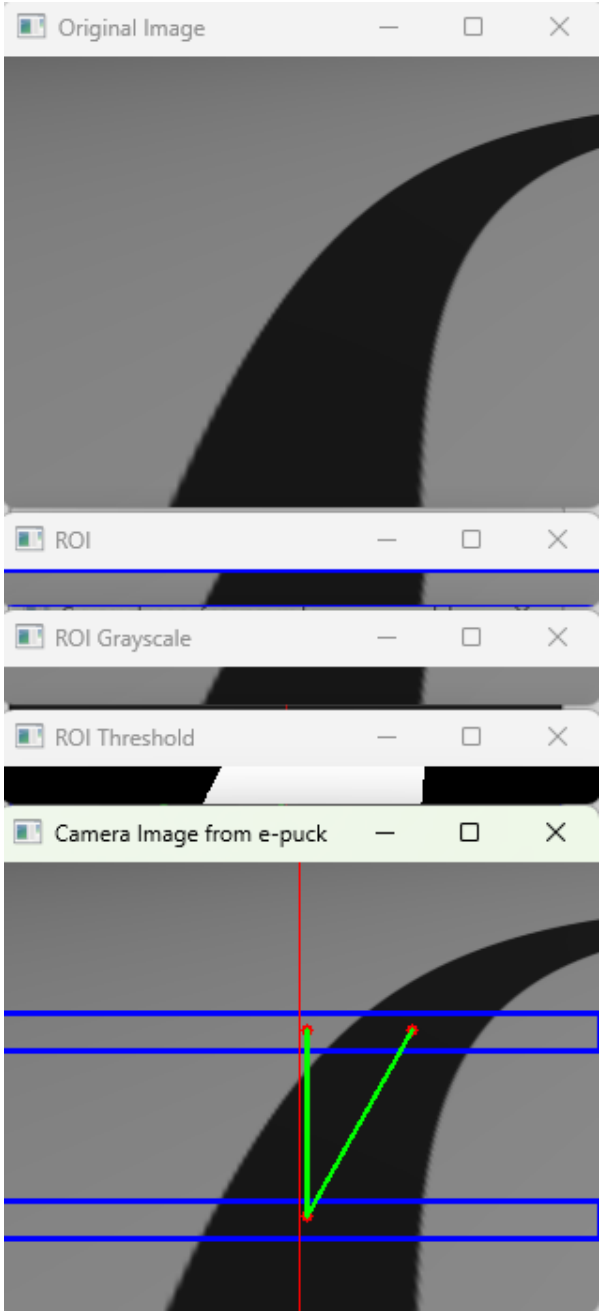


Gambar 17. Pengujian dengan Intensitas Cahaya 1.0

Pengujian pembacaan hasil deteksi garis dengan variasi nilai dari intensitas cahaya terlihat pada **Gambar 18** dan **Gambar 19**. Dari gambar tersebut terlihat bahwa dengan penggunaan *Otsu binarization* untuk penentuan nilai *threshold* dapat mendeteksi garis hitam dengan baik pada kondisi pencahayaan 0.1 hingga 1.0 terbukti pada 2 gambar tersebut. Terlihat pada Gambar 18 dan Gambar 19, meskipun terdapat variasi signifikan dalam intensitas pencahayaan, algoritma deteksi jalur tetap mampu mengidentifikasi jalur hitam secara akurat. Hal ini menunjukkan bahwa penggunaan *Otsu binarization* memberikan kemampuan adaptasi terhadap perubahan pencahayaan, menjaga konsistensi deteksi jalur hitam. Pusat massa jalur hitam tetap terdeteksi dengan baik, yang selanjutnya digunakan untuk perhitungan *error* dan umpan balik ke sistem kendali robot. Dengan demikian, teknik *Otsu binarization* menjadi solusi yang efektif dalam lingkungan dengan kondisi pencahayaan yang bervariasi.



Gambar 18. Hasil pembacaan garis dengan intensitas Cahaya 0.1

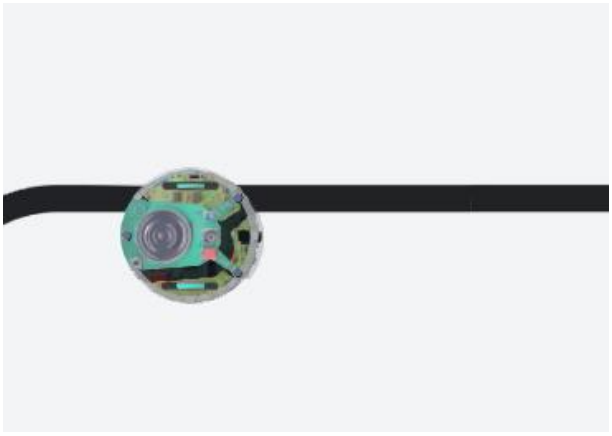


Gambar 19. Hasil pembacaan garis dengan intensitas Cahaya 1.0

B. Penentuan Parameter Kendali PID *Base speed* dan *Delta speed* dan Karakteristik Respon Transiennya

Penentuan parameter kendali PID (*Proportional-Integral-Derivative*) sangat penting dalam mengoptimalkan kinerja sistem kendali robot, khususnya untuk menjaga jalur hitam secara akurat. Dalam penelitian ini, parameter PID ditentukan melalui serangkaian eksperimen untuk menemukan kombinasi terbaik yang menghasilkan respon sistem yang stabil dan cepat.

Pada awal pengujian, seperti yang terlihat pada **Gambar 20**, posisi awal robot sengaja digeser ke kanan dari jalur untuk menguji kemampuan sistem kendali dalam mengembalikan robot ke jalur yang benar. Grafik pada **Gambar 21** menunjukkan perbandingan respon sistem dengan tiga set parameter PID yang berbeda, yaitu Kp=0.01, Ki=0, Kd=0; Kp=0.03, Ki=0, Kd=0.005; dan Kp=0.035, Ki=0.001, Kd=0.004. Dari hasil eksperimen ini, ditemukan bahwa kombinasi Kp=0.035, Ki=0.001, dan Kd=0.004 memberikan respon terbaik dengan *error* yang paling cepat kembali ke nilai nol dan stabil dalam waktu yang singkat.

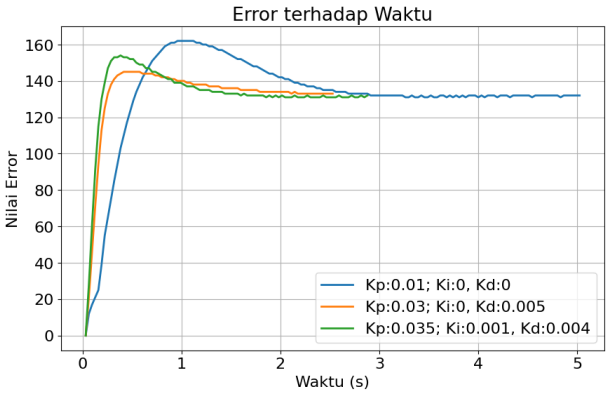


Gambar 20. Posisi awal robot terhadap garis untuk pengujian parameter kendali PID

Parameter Kp (Proportional) berfungsi untuk mengatur besarnya aksi korektif proporsional terhadap *error* yang terjadi. Nilai Kp yang lebih tinggi (0.035) memberikan respon yang lebih cepat terhadap *error*, namun jika terlalu tinggi dapat menyebabkan *overshoot* yang berlebihan. Nilai Ki (Integral) digunakan untuk menghilangkan *error* steady-state, yaitu *error* yang masih tersisa setelah sistem mencapai kestabilan. Penambahan nilai Ki sebesar 0.001 membantu sistem untuk menghilangkan *error* residu dengan lebih efisien.

Komponen Kd (*Derivative*) berperan dalam meredam respon sistem agar tidak terlalu cepat berubah yang dapat menyebabkan osilasi. Nilai Kd yang digunakan adalah 0.004, yang membantu untuk meredam *overshoot* yang dihasilkan oleh aksi proporsional yang tinggi. Kombinasi parameter Kp=0.035, Ki=0.001, dan Kd=0.004 menghasilkan karakteristik respon transien terbaik, di mana sistem mencapai nilai *error* nol dengan cepat dan stabil tanpa osilasi berlebihan. Dari grafik respon *error* terhadap waktu, dapat dilihat bahwa kombinasi ini memberikan waktu pemulihan yang lebih singkat dan *error* yang lebih kecil dibandingkan dengan dua set parameter lainnya. Hasil ini membuktikan bahwa pemilihan

parameter PID yang tepat sangat krusial untuk kinerja optimal sistem kendali robot.

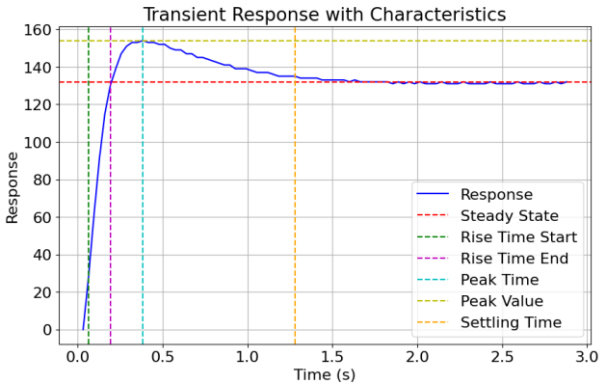


Gambar 21. Pengujian beberapa nilai parameter PID untuk *Delta speed*

Analisis karakteristik respon dari sistem kendali PID dengan parameter Kp=0.035, Ki=0.001, dan Kd=0.004 menunjukkan performa yang cukup baik dalam mengatur sistem menuju keadaan steady-state. Dari grafik yang dihasilkan, waktu naik (rise time) sebesar 0.128 detik menunjukkan bahwa sistem dapat mencapai 90% dari nilai *steady state* dengan cepat. Waktu naik yang cepat ini menunjukkan bahwa nilai Kp yang cukup tinggi memberikan respon awal yang agresif, memungkinkan sistem untuk dengan cepat mendekati nilai targetnya.

Waktu puncak (*peak time*) sebesar 0.384 detik menunjukkan bahwa sistem mencapai nilai maksimum relatif cepat setelah awal transien. Namun, adanya *overshoot* sebesar 16.67% mengindikasikan bahwa respon awal yang agresif dari Kp menyebabkan sistem melewati nilai *steady state* sebelum stabil kembali. Meskipun *overshoot* ini dapat dianggap cukup besar, penambahan komponen derivatif (Kd) sebesar 0.004 membantu meredam osilasi lebih lanjut, sehingga sistem tidak mengalami *overshoot* yang berulang-ulang dan cepat kembali mendekati steady-state.

Sesuai dengan hasil nilai karakteritik dari respon yang terlihat pada **Gambar 22** dan **Tabel II**, *settling time* sebesar 1.28 detik menunjukkan bahwa sistem membutuhkan waktu sekitar 1.28 detik untuk berada dalam batas toleransi 2% dari nilai *steady state* tanpa mengalami deviasi yang signifikan. Hal ini mencerminkan bahwa kombinasi dari nilai Kp, Ki, dan Kd yang digunakan cukup efektif dalam mencapai keseimbangan antara kecepatan respon dan stabilitas sistem. Nilai Ki yang kecil (0.001) membantu dalam menghilangkan *error* steady-state, memastikan bahwa sistem tetap stabil dalam jangka panjang tanpa memperkenalkan osilasi yang signifikan. Parameter kendali PID untuk *Base speed* juga dicari dengan metode yang yang serupa dan didapatkan parameter Kp = 0.05, Ki = 0, dan Kd = 0.001. Secara keseluruhan, kombinasi parameter ini menunjukkan kinerja yang baik dalam hal kecepatan respon dan stabilitas, meskipun ada sedikit *overshoot* yang masih dapat dioptimalkan lebih lanjut.



Gambar 22. Karakteristik Respon Transien dengan parameter PID Kp=0.035, Ki=0.001, dan Kd=0.004 untuk *Delta speed*

Tabel II. Tabel hasil pengujian

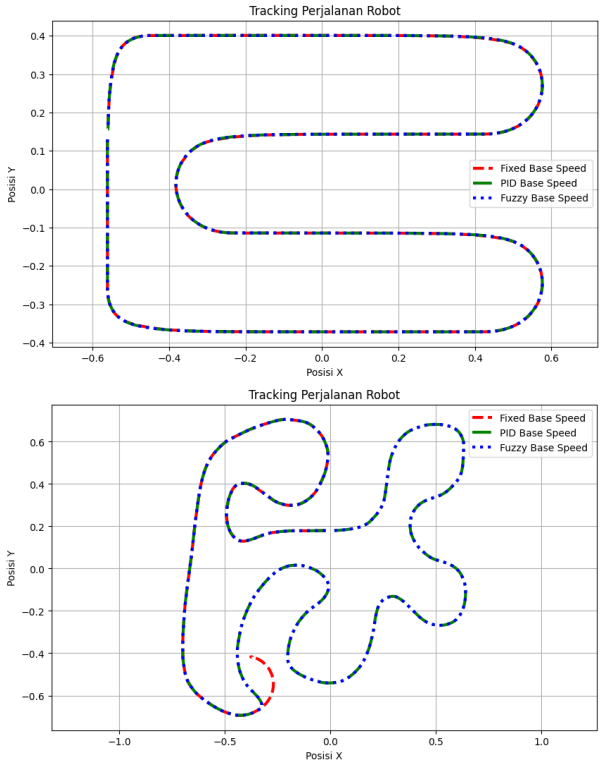
Karakteristik	**Nilai**
Rise Time	0.128 detik
Peak time	0.384 detik
Percentage *Overshoot*	16.67 %
Settling time	1.28 detik

C. Pengujian Perbandingan *Base speed* Tetap, Kendali PID dan Kendali Fuzzy

Pada pengujian Arena 1, yang memiliki jalur sederhana, hasilnya menunjukkan bahwa baik dengan menggunakan *Base speed* Tetap, Kendali PID, maupun Kendali Fuzzy, robot dapat menyelesaikan arena dengan baik. Garis-garis pada **Gambar 23** menunjukkan jalur yang dilalui oleh robot dengan ketiga metode kendali tersebut. Warna merah menunjukkan jalur dengan *Base speed* Tetap, warna hijau menunjukkan jalur dengan Kendali PID, dan warna biru menunjukkan jalur dengan Kendali Fuzzy. Dari gambar tersebut dapat dilihat bahwa ketiga metode menghasilkan jalur yang hampir sama, dengan sedikit variasi pada kelokan-kelokan tajam.

Pada pengujian Arena 2, yang memiliki jalur lebih kompleks dengan banyak kelokan tajam, hasilnya menunjukkan perbedaan yang signifikan antara ketiga metode kendali. Robot dengan *Base speed* Tetap (garis merah) gagal menyelesaikan arena, terlihat pada bagian garis merah yang keluar dari jalur dan terputus. Hal ini menunjukkan bahwa kecepatan tetap tidak mampu mengatasi kelokan tajam dan variasi jalur yang ada. Sementara itu, robot dengan Kendali PID (garis hijau) dan

Kendali Fuzzy (garis biru) berhasil menyelesaikan arena hingga selesai. Kedua metode ini mampu menyesuaikan kecepatan dan arah gerak robot sesuai dengan kondisi jalur, sehingga menghasilkan jalur yang lebih halus dan stabil.



Gambar 23. Tracking jalur robot *line follower* pada (a) Arena 1 dan (b) Arena 2 dengan *Base speed* Fixed, PID dan Fuzzy

Tabel III menunjukkan perbandingan waktu selesai antara *Base speed* Tetap, Kendali PID, dan Kendali Fuzzy di kedua arena. Pada Arena 1, waktu selesai dengan *Base speed* Tetap adalah 60.67 detik, sedangkan dengan Kendali PID dan Kendali Fuzzy masing-masing adalah 50.272 detik dan 52.608 detik. Hal ini menunjukkan bahwa Kendali PID lebih efisien dalam menyelesaikan jalur sederhana dibandingkan dengan kedua metode lainnya. Pada Arena 2, pengujian menunjukkan bahwa *Base speed* Tetap gagal menyelesaikan jalur, sedangkan Kendali PID dan Kendali Fuzzy berhasil. Waktu selesai dengan Kendali PID adalah 91.04 detik dan dengan Kendali Fuzzy adalah 94.048 detik. Meskipun Kendali PID lebih cepat, kedua metode ini menunjukkan kemampuan adaptasi yang baik terhadap jalur yang lebih kompleks dibandingkan dengan *Base speed* Tetap.

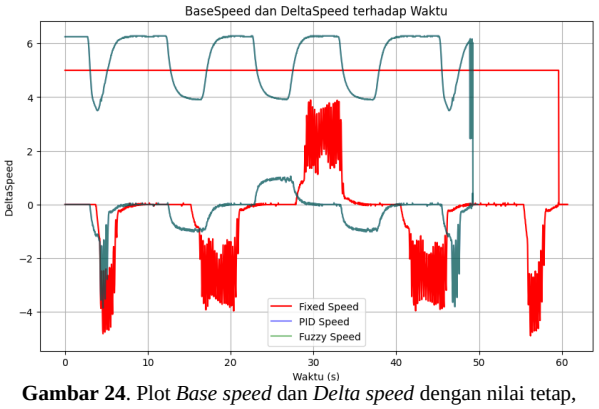
Pada **Gambar 24** terkait plot *Base speed* dan *Delta speed* terhadap waktu di Arena 1, terlihat perbandingan antara *Base speed* Tetap, Kendali PID, dan Kendali Fuzzy. Grafik menunjukkan bagaimana perubahan *Delta speed* (sumbu Y) berhubungan dengan waktu (sumbu X) untuk ketiga metode tersebut. *Base speed* Tetap (garis merah) menunjukkan perubahan *Delta speed* yang cukup signifikan dan sering. Perubahan yang berlebihan ini menandakan bahwa robot dengan *Base speed* Tetap mengalami banyak koreksi kecepatan untuk menyesuaikan dengan jalur. Hal ini menyebabkan gerakan yang kurang halus dan lebih tidak stabil, yang pada akhirnya mempengaruhi efisiensi waktu dalam menyelesaikan arena.

Tabel III. Perbandingan waktu selesai antara *Base speed* Tetap, Kendali PID dan Kendali Fuzzy

Arena	***Base speed* Tetap & *Delta speed* PID**	***Base speed* PID & *Delta speed* PID**	***Base speed* Fuzzy & *Delta speed* Fuzzy**
Arena 1	60.67 detik	50.272 detik	52.608 detik
Arena 2	Gagal	91.04 detik	94.048 detik

Sementara itu, Kendali PID (garis hijau) dan Kendali Fuzzy (garis biru) menunjukkan pola perubahan *Delta speed* yang lebih halus dan stabil. Kedua garis tersebut terlihat berhimpit, yang mengindikasikan bahwa kedua metode ini memiliki performa yang mirip dalam hal penyesuaian kecepatan dan arah gerak robot. Hal ini menunjukkan bahwa baik Kendali PID maupun Kendali Fuzzy mampu menjaga kecepatan robot tetap optimal dan stabil, sehingga memungkinkan robot untuk menyelesaikan jalur dengan lebih efisien dan dengan waktu yang lebih singkat dibandingkan dengan *Base speed* Tetap.

Analisis ini menggarisbawahi bahwa penggunaan Kendali PID dan Kendali Fuzzy lebih optimal dibandingkan dengan *Base speed* Tetap dalam pengaturan kecepatan dinamis untuk robot *line follower*. Kedua metode ini berhasil menjaga stabilitas gerakan robot dan mengurangi frekuensi perubahan *Delta speed* yang berlebihan, yang pada akhirnya meningkatkan efisiensi dan efektivitas dalam penyelesaian arena.



Gambar 24. Plot *Base speed* dan *Delta speed* dengan nilai tetap, Kendali PID dan Kendali Fuzzy pada Arena 1

IV. KESIMPULAN

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat disimpulkan bahwa integrasi teknologi *computer vision* dengan kendali PID dan fuzzy dalam sistem kendali kecepatan dinamis pada robot *line follower* memberikan peningkatan signifikan dalam stabilitas dan efisiensi gerakan robot. Pertama, penggunaan *base speed* dinamis dengan kendali PID dan fuzzy menunjukkan kemampuan adaptasi yang lebih baik terhadap perubahan jalur yang kompleks dibandingkan dengan *base speed* tetap. Hal ini terlihat dari keberhasilan robot menyelesaikan Arena 2 yang lebih kompleks, di mana *base speed* tetap gagal.

Kedua, sistem kendali dengan *base speed* dinamis mampu mengurangi *error* posisi dan sudut secara signifikan, sebagaimana ditunjukkan dalam distribusi *error* yang lebih terpusat di sekitar nol. Pengurangan *error* ini mencerminkan peningkatan akurasi dalam tracking jalur, yang sangat penting dalam menjaga robot tetap pada jalur yang diinginkan. Ketiga, hasil pengujian waktu tempuh menunjukkan bahwa kendali PID lebih efisien dalam menyelesaikan lintasan dibandingkan dengan kendali fuzzy dan *base speed* tetap. Waktu tempuh untuk kendali PID pada Arena 1 dan Arena 2 masing-masing adalah 50.272 detik dan 91.04 detik, lebih singkat dibandingkan dengan kendali fuzzy yang masing-masing membutuhkan 52.608 detik dan 94.048 detik. Ini menunjukkan bahwa sistem kendali PID tidak hanya meningkatkan akurasi tetapi juga efisiensi operasional robot, memberikan waktu tempuh yang lebih cepat dan stabil.

Keempat, pengujian dalam berbagai kondisi pencahayaan menunjukkan bahwa teknik *Otsu* binary thresholding mampu mengatasi masalah deteksi garis dengan baik. Dengan menggunakan metode ini, robot mampu mengenali jalur dengan akurat meskipun ada variasi intensitas pencahayaan, sehingga meningkatkan robustitas sistem secara keseluruhan.

Dengan demikian, penelitian ini menunjukkan bahwa penggunaan teknologi *computer vision* dan kendali PID dalam pengaturan kecepatan dinamis pada robot *line follower* dapat mengatasi masalah yang terkait dengan tikungan tajam dan variasi pencahayaan dengan lebih efektif dibandingkan dengan kendali fuzzy. Penelitian lanjutan disarankan untuk mengoptimalkan parameter kendali PID dan menguji sistem dalam skenario dunia nyata untuk memastikan robustitas dan adaptabilitas yang lebih tinggi.

DAFTAR PUSTAKA

[1] F. McLeay, V. S. Osburg, V. Yoganathan, and A. Patterson, “Replaced by a Robot: Service Implications in the Age of the Machine,” Journal of Service Research, vol. 24, no. 1, hlm. 104–121, 2021

[2] N. Gu, D. Wang, Z. Peng, J. Wang, and Q.-L. Han, “Advances in line-of-sight guidance for path following of autonomous marine vehicles: An overview,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 53, no. 1, hlm. 12–28, 2022

[3] F. Jannah, S. Fuada, H. Putri, F. Zanah, and W. Pratiwi, “Teaching analog Line-Follower (LF) robot concept through simulation for elementary students,” in Journal of Physics: Conference Series, 2021, vol. 1987, no. 1, p. 012046

[4] H. F. Rahman, M. N. Janardhanan, and P. Nielsen, “An integrated approach for line balancing and AGV scheduling towards smart assembly systems,” Assembly Automation, vol. 40, no. 2, hlm. 219–234, 2020

[5] R. Szeliski, *Computer vision*: algorithms and applications. Springer Nature, 2022

[6] D. Babunski, J. Berisha, E. Zaev, and X. Bajrami, “Application of *fuzzy logic* and PID controller for mobile robot navigation,” in 2020 9th Mediterranean Conference on Embedded Computing (MECO), 2020, hlm. 1–4

[7] A. Priambodo, F. Arifin, A. Nasuha, and A. Winursito, “Face tracking for flying robot quadcopter based on haar cascade classifier and PID controller,” in Journal of Physics: Conference Series, 2021, vol. 2111, no. 1, p. 012046

[8] Gajjar, H., Sanyal, S., & Shah, M. (2023). A comprehensive study on lane detecting autonomous car using *computer vision*. Expert Systems with Applications, 233, 120929.

[9] Sridhar, R., Baskar, S., Shaisundaram, V., Karunakaran, K., Ruban, M., & Raja, S. J. I. (2021). Design and development of material behavior of *line follower* automated vehicle. Materials Today: Proceedings, 37, hlm. 2193–2195.

[10] Adam, Y. M., Sariff, N. B., & Algeelani, N. A. (2021). E-puck mobile robot obstacles avoidance controller using the *fuzzy logic* approach. 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), hlm. 107–112.

[11] Amoriya, A., Kumar, C., & Kumari, S. (2024). Counting Number of Objects in Images Using Machine Learning. 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT), 5, hlm. 852–856.

[12] Farley, A., Wang, J., & Marshall, J. A. (2022). How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion. Simulation Modelling Practice and Theory, 120, 102629.

[13] Sezgin, A., & Çetin, Ö. (2020). Autonomous *line follower* robot with fuzzy based hybrid controller. Journal of Intelligent & Fuzzy Systems, 39(5), 6021–6031.

[14] Ramadhan, M. I. A. (2021). Robot *Line follower* dengan Kamera Pengolah Citra Menggunakan Metode Deteksi Kontur dan Perbandingan Intensitas Gambar.

[15] Thamrin, I., Leman, Z. A., Utami, N. P. E., Arrashid, H., & Agustio, L. (2024). *Line follower* mobile robots with adaptive PID control utilizing kinematic model. AIP Conference Proceedings, 2991(1), 050067

[16] Bendimrad, A., El Amrani, A., & El Amrani, B. (2020). Design and implementation of line follower and obstacle detection robot. International Journal of Power Electronics and Drive Systems, 11(1), 160.

[17] Castillo, O., Cortés-Antonio, P., Melin, P., & Valdez, F. (2020). Type-2 fuzzy control for line following using line detection images. Journal of Intelligent & Fuzzy Systems, 39(5), 6089–6097.