JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 12 No. 3, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v12i3.4718

PENERAPAN OPENCV DAN FUZZY LOGIC CONTROLLER UNTUK LINE FOLLOWER BERBASIS KAMREA PADA SIMULASI ROBOT E-PUCK DI WEBOTS

Sheisya Rhieyanetta Divanny[1*], Zaki Nugraha[2*], Muhammad Ifan Ghaffar[3], Muhammad Rizqi Dwi A[4], Syahril Dwi Saputra Wardana[5], Ardy Seto Priambodo[6]

[1,2,3,4,5,6]Universitas Negeri Yogyakarta; Jl. Mandung, Serut, Pengasih, Kec. Wates, Kabupaten Kulon Progo, Daerah Istimewa Yogyakarta, 55651; Telp. (0274) 586168

Received: 1 Juli 2024
Accepted: 31 Juli 2024
Published: 7 Agustus 2024

Keywords:
Robot Pengikut Garis;
E-puck;
Fuzzy Logic;
Sistem Kamera;
Open CV.

Corespondent Email:
sheisyarhieyanetta.2022@student.uny.ac.id;
zakinugraha.2021@student.uny.ac.id

Abstrak. Perkembangan teknologi berkembang ke berbagai bidang termasuk industri, dan robot seperti robot line-following merupakan salah satu teknologi yang berkembang pesat untuk meningkatkan efisiensi produksi. Penelitian ini bertujuan untuk mengembangkan sistem kendali logika fuzzy pada robot e-puck yang dilengkapi dengan kamera pelacak garis. Penelitian ini menggunakan teknik observasi literatur terkait robot pelacak garis, perancangan robot dan kamera, perancangan algoritma pengolahan citra dan perancangan algoritma *fuzzy logic controller*. Kamera E-Puck digunakan untuk mendeteksi garis, data gambar diolah menggunakan algoritma OpenCV, dan logika fuzzy diimplementasikan untuk mengendalikan pergerakan robot. Robot ditentukan berdasarkan nilai error dan delta error yang ditentukan dari pengukuran kamera. Hasil pengujian menunjukkan bahwa robot dapat melacak lintasan dalam berbagai bentuk, dengan nilai error rata-rata 85 piksel dan nilai error delta rata-rata 0,1 piksel, serta motor dapat menggerakkan robot sepanjang gerak lintasan yang terdeteksi dapat merespons dengan tepat. Berkontribusi pada pengembangan sistem robot pelacak garis yang andal menggunakan kombinasi kamera dan logika fuzzy.

Abstract. *Technological developments are expanding into various fields including industry, and robots such as line-following robots are one technology that is developing rapidly to increase production efficiency. This research aims to develop a fuzzy logic control system for an e-puck robot equipped with a line tracking camera. This research uses literature observation techniques related to line tracking robots, robot and camera design, image processing algorithm design and fuzzy logic controller algorithm design. The E-Puck camera is used to detect lines, image data is processed using the OpenCV algorithm, and fuzzy logic is implemented to control the robot's movement. The robot is determined based on the error and delta error values determined from camera measurements. The test results show that the robot can track trajectories in various shapes, with an average error value of 85 pixels and an average delta error value of 0.1 pixels, and the motor can move the robot along the detected trajectory motion and can respond appropriately. Contribute to the development of a reliable line tracking robot system using a combination of cameras and fuzzy logic.*

1. PENDAHULUAN

Dalam beberapa dekade terakhir, perkembangan teknologi robotika telah mengalami kemajuan yang pesat terutama dalam bidang sistem kendali robot. Salah satunya yaitu robot *line follower*. Robot ini umumnya digunakan dalam berbagai kompetisi dan aplikasi industri untuk melakukan navigasi otomatis dengan mengikuti jalur yang telah ditentukan [1]. Dalam penelitian terbaru, berbagai pendekatan telah diimplementasikan untuk meningkatkan kinerja robot *line follower*, termasuk penggunaan sensor inframerah, sensor ultrasonik, dan kamera [2][3][4]. Teknologi kamera telah digunakan untuk menggantikan sensor inframerah karena menawarkan keuntungan seperti yang telah dijelaskan dalam [3].

Algoritma dan metode kendali bagi line follower telah banyak dikembangkan. Misalnya, algoritma PID *(Proportional-Integral-Derivative)* yang cukup sederhana dan efisien dalam mengoreksi deviasi dari jalur yang diinginkan [5]. Selain PID, *Fuzzy Logic Controller* (FLC) juga banyak digunakan karena lebih adaptif dalam menangani ketidakpastian dan non-linearitas yang sering terjadi di dunia nyata [6].

Agusma, dkk [7] melakukan penelitian pada robot line follower menggunakan sensor IR dengan algoritma *Fuzzy Logic* dan metode mamdani menghasilkan waktu respon yang cukup lambat (2,5 detik untuk memperbaiki *error* sudut sebesar 1,7 derajat).

Kemudian Yudha, dkk [8] mengembangkan robot jetbot *line follower* berbasis kamera dengan menggunakan metode *Fuzzy Logic* melalui simulasi pada Webots dan mendapatkan hasil respon robot yang lebih cepat (0,6 hingga 0,7 detik). Namun, penulis tidak menjelaskan secara rinci proses konversi dari PWM menjadi kecepatan motor.

Setelah membandingkan beberapa penelitian sebelumnya, dapat disimpulkan bahwa penggunaan sensor IR dalam efisiensi waktu yang kurang efisien dengan akurasi kondisi kecepatan penggerak motor yang berbeda, sementara itu penggunaan kamera dianggap sebagai pilihan yang lebih efisien dan akurat dibandingkan dengan sensor IR berdasarkan analisis penelitian yang ada. Meskipun kamera memiliki kelebihan dalam efisiensi waktu, tetapi dalam penelitian Yudha, dkk tidak disebutkan secara rinci proses konversi dari sinyal PWM menjadi kecepatan motor.

Penelitian ini bertujuan untuk mengatasi kesenjangan yang ada dengan fokus peningkatan efisiensi dan kecepatan respons serta transparansi proses kendali, termasuk konversi sinyal dan penyesuaian kecepatan motor. Penulis memilih robot e-puck yang dilengkapi dengan sensor kamera untuk mendeteksi garis, menggunakan simulasi pada webots.

2. TINJAUAN PUSTAKA

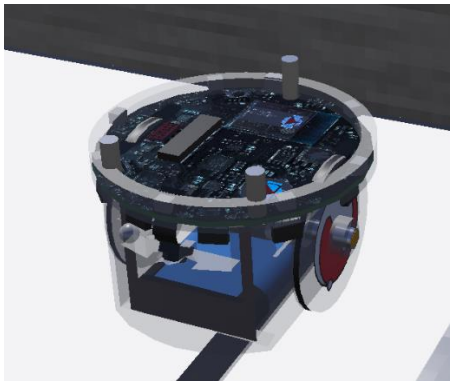
2.1. Webots

Webots adalah sebuah aplikasi simulasi robot 3D yang dikembangkan oleh *Cyberbotics Ltd.* sejak tahun 1998. Aplikasi Webots telah banyak digunakan dalam bidang industri, edukasi, dan pelitian [9].

Mulanya, aplikasi Webots adalah aplikasi berlisensi hak milik (*proprietary*). Kemudian pada 18 Desember 2018, Cyberbotics Ltd. selaku pemegang lisensi tersebut mengumumkan bahwa untuk Webots versi R2019a dan selanjutnya akan berubah menjadi aplikasi *open-source* dibawah lisensi *Apache* 2.0 [10].

2.2. E-puck

E-puck adalah miniatur robot bergerak yang dikembangkan oleh EPFL (*École polytechnique fédérale de Lausanne*) dan ditujukan untuk proses pembelajaran oleh para perancang robot Khepera yang telah sukses sebelumnya. Robot e-puck ini dari segi *hardware* dan *software* secara keseluruhan bersifat *open-source*. Selain itu, robot e-puck juga menawarkan akses level terendah untuk setiap perangkatnya dan memungkinkan ekstensi yang tidak terbatas seperti motor roda diferensial, sensor infrared sebagai sensor jarak dan cahaya, akselerometer, kamera, 8 LED, fitur komunikasi *Bluetooth*, dan sensor tanah [11].



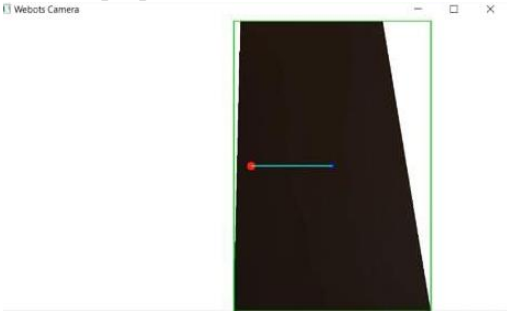
Gambar 1. Robot E-puck

2.3. Fuzzy Logic

Fuzzy Logic atau logika fuzzy diperkenalkan oleh Lothfi Zadeh pada tahun 1965 [12], merupakan perluasan dari logika Boolean klasik yang mempertimbangkan konsep kebenaran parsial. Berbeda dengan logika Boolean yang hanya mengenal nilai benar (1) dan salah (0), logika fuzzy memungkinkan nilai kebenaran di antara keduanya, sehingga dapat merepresentasikan ketidakpastian dan ambiguitas yang seringkali hadir di dunia nyata [13].

2.4. OpenCV

Open Computer Vision (OpenCV) adalah sebuah pustaka perangkat lunak *open-source* yang dikenalkan oleh Intel untuk kebutuhan pengolahan citra digital seperti analisis gambar dan video [14].



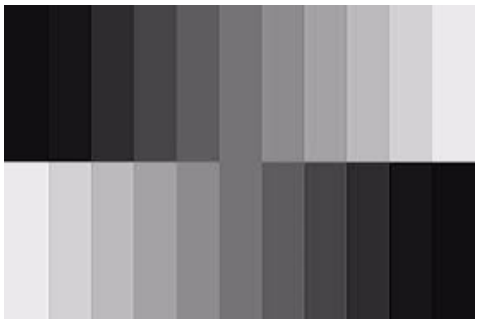
Gambar 2. Tampilan Kamera dengan OpenCV

2.4.1. Pengolahan Citra Digital (Digital Image Processing).

Pengolahan citra digital (*Digital Image Processing*) merupakan proses menganalisis citra (gambar atau video) dengan menggabungkan sistem *computer vision* sehingga mendapatkan informasi yang diperlukan dan dapat diaplikasikan dalam berbagai bidang [15]. Citra digital merupakan representasi visual dari dunia nyata yang terdiri dari elemen-elemen titik yang disebut piksel dalam bentuk baris dan kolom. Setiap piksel dapat merepresentasikan satu nilai (tingkat keabuan) atau tiga nilai (*Red-Green-Blue*) [16].

2.4.2. Tingkat Keabuan (Grayscale)

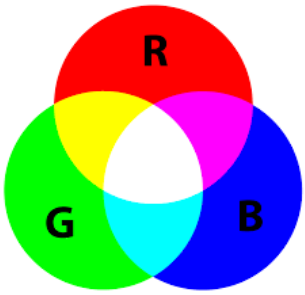
Citra *Grayscale* adalah citra yang setiap pikselnya hanya memiliki satu nilai yaitu nilai tingkat keabuan atau biasa disebut dengan monokromatik [16]. Nilai maksimum dari *grayscale* adalah 255 (putih) dan minimumnya adalah 0 (hitam). Jumlah nilai tersebut disesuaikan dengan jumlah *bit* yang digunakan seperti yang dijelaskan dalam penelitian yang dilakukan oleh RD. Kusumanto dan A.N. Tompunu [17].



Gambar 3. Warna *Grayscale*.

2.4.3. RGB (Red-Green-Blue)

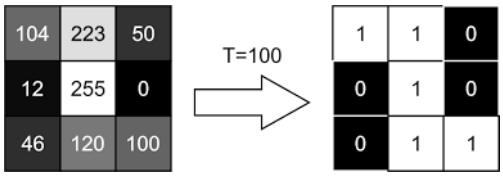
Citra warna adalah citra yang setiap pikselnya memiliki tiga nilai numerik, *Red* (Merah), *Green* (Hijau), dan *Blue* (Biru) sebagai warna utama [16][18]. Nilai *Red* (Merah), *Green* (Hijau), dan *Blue* (Biru) berkisar antara 0-255. Dalam piksel, nilai tersebut dituliskan dengan format [R, G, B]. Misalnya, [0,0,255] akan menghasilkan nilai *Blue* (Biru). Sedangkan [255,0,255] akan menghasilkan warna *Magenta* (Ungu) [19].



Gambar 4. Warna RGB

2.4.4. Ambang Batas (Thresholding)

Thresholding (Ambang Batas) adalah konversi citra hitam putih menjadi citra biner. Hal ini dilakukan dengan mengelompokkan nilai skala abu-abu setiap piksel ke dalam dua kelas. Hitam (0) direpresentasikan dalam biner dengan bit 0, dan putih (255) direpresentasikan dalam biner dengan bit 1 [20].



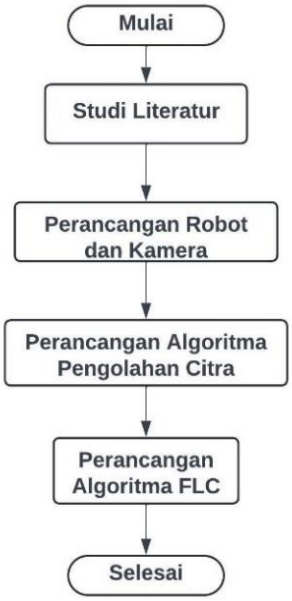
Gambar 5. Thresholding

2.4.5. Kontur (Contour)

Kontur dalam pengolahan citra mengacu pada garis atau kurva yang menandai batas atau tepi suatu objek dalam suatu gambar. Biasanya, tepi tertentu mengacu pada piksel batas yang memiliki warna dan intensitas yang sama. Proses deteksi tepi memainkan peran penting dalam mengidentifikasi, mengukur, dan memisahkan objek secara akurat dalam gambar [21].

3. METODE PENELITIAN

Dalam penelitian ini, terdapat beberapa tahapan yang dilakukan. Dimulai dari Studi Literatur, kemudian perancangan robot dan kamera, perancangan algoritma pengolahan citra, dan perancangan algoritma logika fuzzy kontroler.



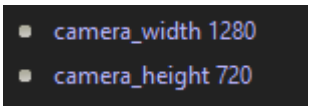
Gambar 6. Metode Penelitian

3.1. Studi Literatur

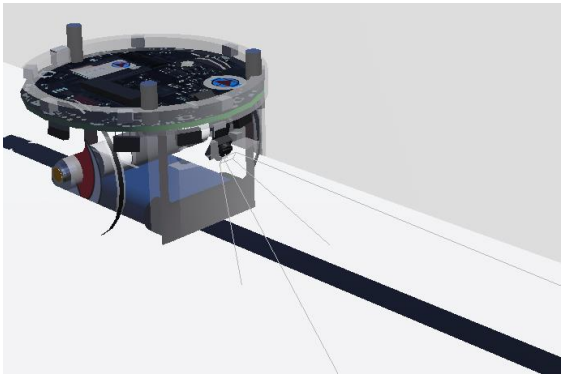
Studi literatur dilakukan dengan mengumpulkan data dari penelitian yang sudah dilakukan sebelumnya tentang *Line Follower* Robot kemudian menganalisis dan mengkombinasikan sehingga dapat dijadikan referensi dalam penelitian ini.

3.2. Perancangan Robot dan Kamera

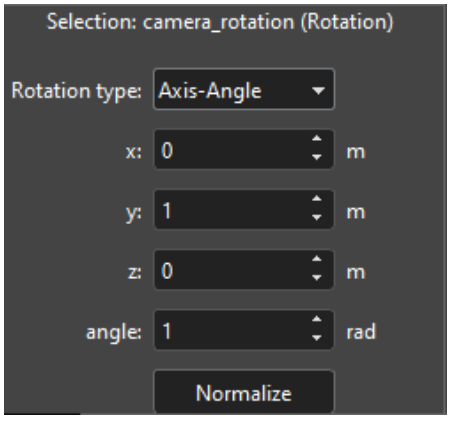
Perancangan kamera pada robot e-puck bertujuan untuk mengembangkan sistem yang dapat digunakan untuk mendeteksi garis. Dalam aplikasi Webots, penulis menggunakan kamera yang telah terpasang pada robot e-puck dengan resolusi 1280x720 piksel dan konfigurasi sudut $x = 0$, $y = 1$, dan $z = 0$ dengan sudut kamera 1 radian.



Gambar 7. Resolusi Kamera Pada Webots

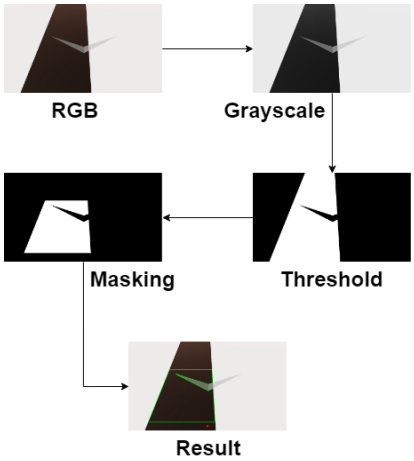


Gambar 8. Sudut Kamera Pada Robot



Gambar 9. Sudut Kamera Pada Robot

3.3. Perancangan Algoritma Pengolahan Citra



Gambar 10. Sudut Kamera Pada Robot

Metode utama yang digunakan robot dalam mendeteksi garis pada penelitian ini yaitu menggunakan algoritma kontur yang secara bawaan telah disediakan oleh OpenCV. Dalam aplikasi Webots, hasil gambar yang ditangkap oleh modul kamera disimpan ke dalam *buffer*. Kemudian nilai dalam *buffer* tersebut diolah menjadi gambar menggunakan pustaka *Numpy* dengan format warna BGR. Hasil dari format warna BGR ini dirubah kedalam format warna RGB. Selanjutnya format warna tersebut dirubah menjadi format warna *grayscale*. Tujuannya adalah untuk meringankan proses komputasi karena piksel dari gambar yang diolah hanya memiliki satu jenis nilai saja, yaitu tingkat keabuan (*grayscale*). Setelah menggunakan nilai *threshold* untuk menghasilkan masker biner dari gambar *grayscale*, di mana nilai threshold 100 mengubah piksel dengan intensitas di atas 100 menjadi hitam (0) dan di bawah 100 diubah menjadi putih (255), langkah berikutnya adalah mengekstraksi garis yang dianggap sebagai jalur yang harus diikuti oleh robot. Di tengah gambar, sebuah lingkaran dengan radius 10 dan warna merah (0,0,255) digambar untuk menandai titik tengah. Garis utama yang harus diikuti, direpresentasikan oleh kontur terbesar dari semua yang ditemukan, ditandai dengan titik tengah berwarna hijau (0,255,0). Selanjutnya, sebuah titik biru digunakan untuk menunjukkan pusat dari *bounding rectangle* yang di gunakan untuk menentukan area terkecil yang dapat menutupi kontur dalam gambar dan garis kuning (255,255,0) digambar dari pusat bounding rectangle ke titik tengah gambar, dihitung dengan persamaan (1) dan (2).

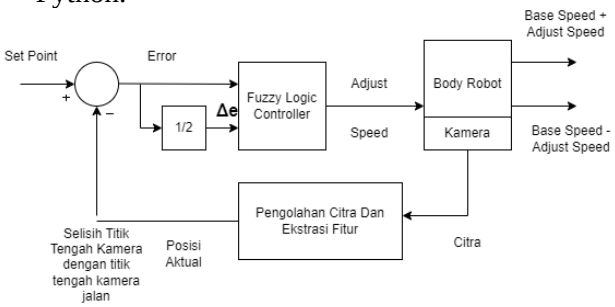
$$center\_x = x + \frac{w}{2} \quad (1)$$

$$center\_y = y\_pot \quad (2)$$

Dengan demikian, proses ini secara visual membantu navigasi robot dengan menyoroti rute yang harus diambil dalam gambar yang ditangkap dari kamera.

3.4. Perancangan Algoritma FLC

Penulis menggunakan algoritma FLC (*Fuzzy Logic Controller*) sebagai algoritma kendali robot. Algoritma FLC ini dipilih karena dapat merepresentasikan ketidakpastian yang terjadi dan tidak memerlukan model matemtis dalam proses kendalinya. Dalam pembuatan logika fuzzy, penulis menggunakan pustaka pada *scikit-fuzzy* dengan bahasa pemograman Python.



Gambar 11. Diagram Blok Sistem FLC

Seperti yang ditampilkan dalam diagram blok diatas, input yang digunakan dalam algoritma logika fuzzy adalah input dari error dan delta error. Nilai error didapatkan dari selisih antara titik tengah kamera dengan titik tengah jalan atau *Center of Line* (*CoL*). Sedangkan delta error didapatkan dari selisih error saat ini (*present error*) dengan error sebelumnya (*last error*).

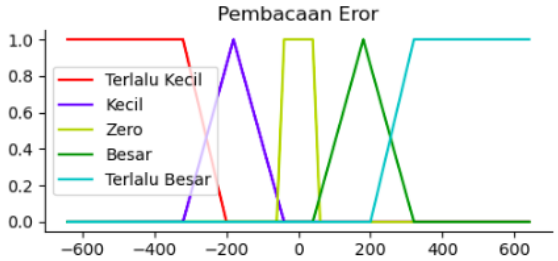
$$error = \left(\frac{cam\ height}{2}\right) - CoL \quad (3)$$

$$\Delta error = present\ error - last\ error \quad (4)$$

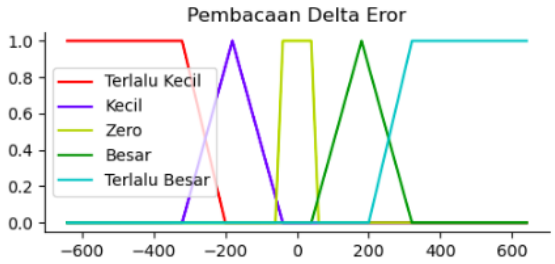
3.4.1. Fuzzifikasi

Pada proses fuzzifikasi, nilai input, yaitu *error* dan *delta error* dimasukkan ke dalam fungsi membership (*membership function*),

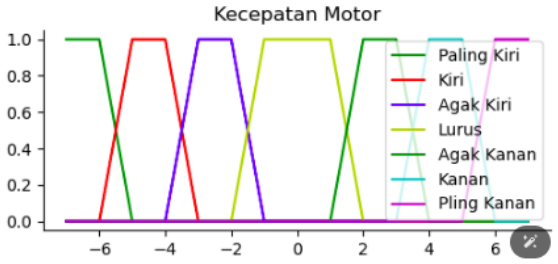
penulis menggunakan 5 fungsi keanggotaan pada nilai input (*error* dan *delta error*) dan 7 fungsi keanggotaan pada nilai output (*adjustment error*).



Gambar 12. Fungsi Keanggotaan *Error*



Gambar 13. Fungsi Keanggotaan *Delta Eror*



Gambar 14. Fungsi Keanggotaan *Output*

Untuk rentang keanggotaan *error* yang ditunjukkan pada Gambar 12 dijelaskan dalam persamaan (5) hingga (9).

$$\mu\ \text{Terlalu Kecil}\ (x) = \begin{cases} Y = 1,\ X \le -640 \\ 0 < Y < 1,\ -320 \le X \le -200 \\ Y = 0,\ X \ge -200 \end{cases} \quad (5)$$

$$\mu\ \text{Kecil}\ (x) = \begin{cases} Y = 0,\ X \le -320 \text{ atau } X \ge -40 \\ 0 < Y < 1,\ -200 \le X \le -40 \\ Y = 1,\ -40 \le X \le -20 \end{cases} \quad (6)$$

$$\mu\ \text{Zero}\ (x) = \begin{cases} Y = 0,\ X \le -60 \\ 0 < Y < 1,\ -60 \le X \le -40 \\ Y = 1,\ X \le 60 \\ 1 < Y < 0,\ 40 \le X \le 60 \end{cases} \quad (7)$$

$$\mu\ \text{Besar}\ (x) = \begin{cases} Y = 0,\ X \le 320 \text{ atau } X \ge 40 \\ 0 < Y < 1,\ 200 \le X \le 40 \\ Y = 1,\ 40 \le X \le 20 \end{cases} \quad (8)$$

$$\mu\ \text{Terlalu Besar}\ (x) = \begin{cases} Y = 1,\ X \le 640 \\ 0 < Y < 1,\ 200 \le X \le 320 \\ Y = 0,\ X \ge 200 \end{cases} \quad (9)$$

Sedangkan rentang keanggotaan *delta error* yang ditunjukkan pada Gambar 13 dijelaskan dalam persamaan (10) hingga (14).

$$\mu\ \text{Terlalu Kecil}\ (x) = \begin{cases} Y = 1,\ X \le -640 \\ 0 < Y < 1,\ -320 \le X \le -200 \\ Y = 0,\ X \ge -200 \end{cases} \quad (10)$$

$$\mu\ \text{Kecil}\ (x) = \begin{cases} Y = 0,\ X \le -320 \text{ atau } X \ge -40 \\ 0 < Y < 1,\ -200 \le X \le -40 \\ Y = 1,\ -40 \le X \le -20 \end{cases} \quad (11)$$

$$\mu\ \text{Zero}\ (x) = \begin{cases} Y = 0,\ X \le -60 \\ 0 < Y < 1,\ -60 \le X \le -40 \\ Y = 1,\ X \le 60 \\ 1 < Y < 0,\ 40 \le X \le 60 \end{cases} \quad (12)$$

$$\mu\ \text{Besar}\ (x) = \begin{cases} Y = 0,\ X \le 320 \text{ atau } X \ge 40 \\ 0 < Y < 1,\ 200 \le X \le 40 \\ Y = 1,\ 40 \le X \le 20 \end{cases} \quad (13)$$

$$\mu\ \text{Terlalu Besar}\ (x) = \begin{cases} Y = 1,\ X \le 640 \\ 0 < Y < 1,\ 200 \le X \le 320 \\ Y = 0,\ X \ge 200 \end{cases} \quad (14)$$

Rentang keanggotaan pada output yang ditunjukkan pada Gambar 14 dijelaskan dalam persamaan (15) hingga (21).

$$\mu\ \text{Paling Kiri}\ (x) = \begin{cases} Y = 1,\ X \le -7 \\ 0 < Y < 1,\ -6 \le X \le -5 \\ Y = 0,\ X \ge -5 \end{cases} \quad (15)$$

$$\mu\ \text{Kiri}\ (x) = \begin{cases} Y = 0,\ X \le -6 \\ 0 < Y < 1,\ -6 \le X \le -5 \\ Y = 1,\ X \le -3 \\ 1 < Y < 0,\ -3 \le X \le -4 \end{cases} \quad (16)$$

$$\mu\ \text{Agak Kiri}\ (x) = \begin{cases} Y = 0,\ X \le -4 \\ 0 < Y < 1,\ -4 \le X \le -3 \\ Y = 1,\ X \le -1 \\ 1 < Y < 0,\ -1 \le X \le -2 \end{cases} \quad (17)$$

$$\mu\ \text{Lurus}\ (x) = \begin{cases} Y = 0,\ X \le -2 \\ 0 < Y < 1,\ -2 \le X \le -1 \\ Y = 1,\ X \le 2 \\ 1 < Y < 0,\ 1 \le X \le 2 \end{cases} \quad (18)$$

$$\mu\ \text{Agak Kanan}\ (x) = \begin{cases} Y = 0,\ X \le 1 \\ 0 < Y < 1,\ 1 \le X \le 2 \\ Y = 1,\ X \le 4 \\ 1 < Y < 0,\ 3 \le X \le 4 \end{cases} \quad (19)$$

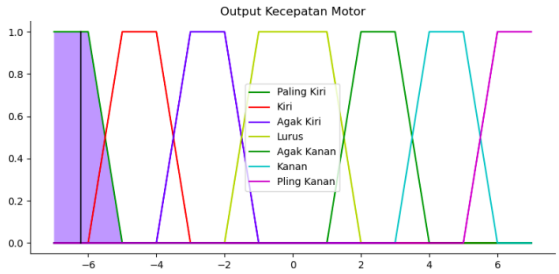
$$\mu\ \text{Kanan}\ (x) = \begin{cases} Y = 0,\ X \le 3 \\ 0 < Y < 1,\ 3 \le X \le 4 \\ Y = 1,\ X \le 6 \\ 1 < Y < 0,\ 5 \le X \le 6 \end{cases} \quad (20)$$

$$\mu\ \text{Paling Kanan}\ (x) = \begin{cases} Y = 1,\ X \le 7 \\ 0 < Y < 1,\ 5 \le X \le 6 \\ Y = 0,\ X \ge 5 \end{cases} \quad (21)$$

3.4.2. Defuzzifikasi

Setelah input dimasukkan dalam aturan fuzzy, selanjutnya adalah proses inferensi. Pada proses inferensi, penulis meggunakan metode Mamdani karena metode madani merupakan metode bawaan dalam pustaka *scikit-fuzzy*. Hasil dari proses inferensi tersebut di *defuzzifikasi* menggunakan metode *centroid*. Tujuan menggunakan metode *centroid* dalam

proses *defuzzifikasi* ini karena untuk menemukan pusat area di bawah kurva fungsi keanggotaan fuzzy yang mewakili hasil keseluruhan dari berbagai aturan fuzzy yang diterapkan. Garis hitam vertikal yang terdapat pada Gambar 15 adalah hasil *defuzzifikasi* menggunakan metode *centroid*.

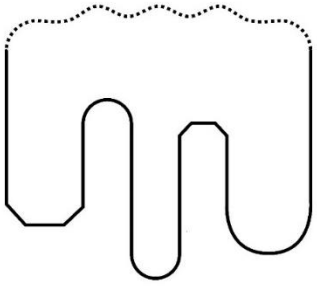


Gambar 15. Defuzzifikasi

4. HASIL DAN PEMBAHASAN

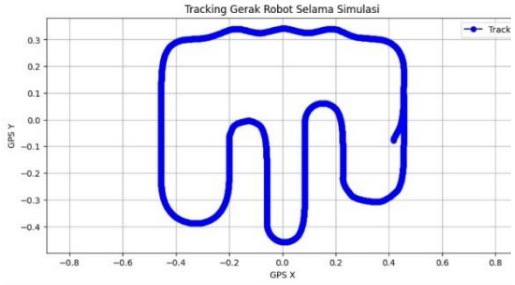
Pengujian dilakukan menggunakan jalur yang memiliki beberapa tikungan, garis dengan sudut tertentu, dan garis putus-putus. Jalur ini ditampilkan pada Gambar 16, yang menunjukkan posisi dan bentuk arena secara jelas. Untuk memberi informasi kepada robot mengenai keberadaan tikungan, garis dengan sudut tertentu, dan garis putus-putus, sebuah garis melintang ditempatkan tepat sebelum setiap elemen tersebut. Fungsi garis ini adalah memberikan sinyal kepada robot bahwa tikungan telah dicapai, sehingga robot dapat mempersiapkan manuver yang diperlukan.

Dengan mendeteksi garis ini, robot dapat mulai menghitung nilai error dan delta error setelah memasuki tikungan, garis dengan sudut tertentu, dan garis putus-putus. Hal ini memungkinkan pengukuran kinerja robot yang akurat dalam menghadapi tikungan, garis dengan sudut tertentu, dan garis putus-putus dengan berbagai radius, sehingga dapat dioptimalkan lebih lanjut sesuai dengan kebutuhan pengujian. Berikut adalah Gambar 16 untuk tampilan lintasan yang digunakan.



Gambar 16. Lintasan yang digunakan

Selain itu, untuk melacak pergerakan robot secara lebih akurat, data dari GPS dapat digunakan untuk menunjukkan jalur yang ditempuh robot. Dengan menggabungkan citra yang diambil oleh kamera dan data posisi dari GPS, jalur yang diikuti robot dapat divisualisasikan, memberikan gambaran yang jelas tentang performa robot dalam mengikuti garis. Ini memungkinkan analisis lebih lanjut dan penyesuaian algoritma kontrol untuk meningkatkan akurasi dan efisiensi robot line follower. Dengan demikian, penggunaan kamera dan GPS secara bersamaan. Berikut adalah Gambar 17 untuk tampilan tracking gerak robot selama simulasi.



Gambar 17. Tracking Gerak Robot Selama Simulasi

Gambar di atas menunjukkan jalur yang menyimpang dari lintasan ideal karena kemiringan yang disengaja pada peletakan robot selama uji coba. Tujuan dari gambar ini adalah untuk mengukur kemampuan robot untuk menemukan jalur yang benar.

Data yang dikumpulkan pada setiap putaran dalam pengujian ini mencakup nilai error maksimum, minimum, dan rata-rata serta delta error yang dialami robot saat melewati setiap tikungan, garis dengan sudut tertentu, dan garis putus-putus. Nilai rata-rata diperoleh dari nilai eror total dan nilai delta error total selama pengujian, sedangkan nilai minimum dan maksimum diperoleh saat robot melewati lintasan tersebut. Pengujian dilakukan lima kali putaran. Nilai data dari semua data yang dikumpulkan selalu identik dengan yang ditunjukkan pada Tabel 2 dan 3.

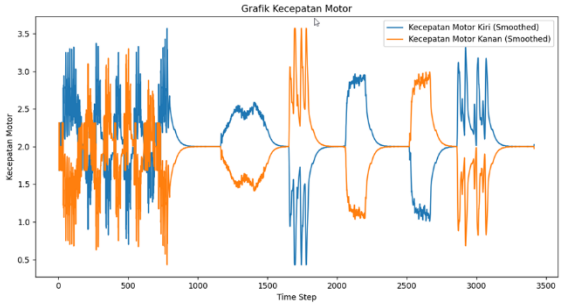
Tabel 2. Pengujian Error

LAP	Eror		
	Min	Max	Rata - Rata
Putaran	- 486 piksel	468 piksel	-85.5921 piksel

Tabel 3. Pengujian Delta Error

LAP	Delta Eror		
	Min	Max	Rata - Rata
Putaran	-436 piksel	429 piksel	-0.11013 piksel

Dengan demikian, setelah menjalankan robot melewati lintasan, diperoleh grafik kecepatan motor. Grafik ini ditunjukkan setelah dilakukan proses smoothing pada Gambar 18.



Gambar 18. Grafik Kecepatan Motor

Grafik diatas adalah grafik menunjukkan dua garis yang mewakili dua set data yang berbeda yaitu Kecepatan Motor Kiri (Smoothed) berwarna biru dan Kecepatan Motor Kanan (Smoothed) berwarna oranye.

Sumbu x berlabel Time Step dengan rentang dari 0 hingga sekitar 3500, sedangkan sumbu y berlabel Kecepatan Motor dengan rentang dari 0,5 hingga 3,5. Garis-garis tersebut menunjukkan kecepatan motor dari waktu ke waktu, dengan data yang sudah disempurnakan seperti yang tertera pada label, yang berarti bahwa beberapa bentuk rata-rata filter telah diterapkan untuk mengurangi noise dan membuat grafik lebih jelas.

Grafik tersebut menunjukkan fluktuasi dalam kecepatan motor, dengan beberapa puncak dan lembah yang menunjukkan perubahan kecepatan dari waktu ke waktu. Ada juga periode di mana garis-garis tersebut datar, menunjukkan kecepatan motor yang stabil selama interval tersebut.

5. KESIMPULAN

a. Robot yang kami kembangkan akan mampu melacak dan mengikuti jalur dengan berbagai macam bentuk, antara lain tikungan, garis dengan sudut tertentu, dan garis putus-putus.
b. Pengolahan gambar digital menggunakan perpustakaan OpenCV memungkinkan robot mengenali garis dengan akurasi tinggi. Lalu kendali logika fuzzy yang diterapkan dapat mengoptimalkan pergerakan robot mengikuti lintasan yang terdeteksi.
c. Berdasarkan pengujian yang dilakukan, robot bekerja dengan baik dengan rata-rata nilai error 85 piksel dan rata-rata nilai delta error 0,1 piksel. Selain itu, motor merespons dengan lancar kecepatan robot saat bergerak sepanjang lintasan tertentu. Pengulangan pengujian menunjukkan bahwa nilai data yang diperoleh konsisten dan sangat akurat.

DAFTAR PUSTAKA

[1] A. Pathak, R. Khan Pathan, A. Tutul, N. Tousi, A. Rubaba, and N. Bithi, “Line Follower Robot for Industrial Manufacturing Process,” *International Journal of Engineering Inventions*, vol. 6, no. 10, pp. 10–17, 2017,

[2] K. M. Hasan, Abdullah-Al-Nahid, and A. Al Mamun, “Implementation of autonomous line follower robot,” in *IEEE Xplore*, May 01, 2012. https://ieeexplore.ieee.org/abstract/document/6317486.

[3] M. A. Nugraha, D. Syauqi, dan R. R. M. Putri, “Perancangan dan Implementasi Robot Line Follower Menggunakan Avoid Obstacle dengan Metode Wall Following”, *J-PTIIK*, vol. 8, no. 3, Mar 2024.

[4] A. Kondakor, Z. Torcsvari, A. Nagy, and I. Vajk, “A Line Tracking Algorithm Based on Image Processing,” in 2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), May 2018, doi: https://doi.org/10.1109/saci.2018.8440975.

[5] K. Joni, M. Ulum, and Z. Abidin, “Robot Line Follower Berbasis Kendali Proportional-Integral-Derivative (PID) Untuk Lintasan Dengan Sudut Ekstrim”, *JURNAL INFOTEL - Informatika Telekomunikasi Elektronika*, vol. 8, no. 2, p. 138, Nov. 2016, doi: https://doi.org/10.20895/infotel.v8i2.129.

[6] B. Aji and S. Sutikno, “Fuzzy Logic Algorithm of Sugeno Method for Controlling Line Follower Mobile Robot,” *ILKOM Jurnal Ilmiah*, vol. 15, no. 2, pp. 283–289, Aug. 2023.

[7] I Putu Adinata Mas Pratama, I Nengah Suweden, I.B. Alit Swamardika "Sistem Kontrol Pergerakan Pada Robot Line Follower Berbasis Hybrid PID-Fuzzy Logic" 14-15 November 2013 [Online] Available: https://ojs.unud.ac.id/index.php/prosidingcsgteis2013/article/download/7244/5493.

[8] Yudha Febrian, Rasyid Ammary Yahya, M. Ibrah Dzaluli, Ardy Seto Priambodo, ST., M.Eng, "Implementasi Fuzzy Logic dengan sistem Visual Camera pada Robot Jetbot sebagai Line Follower," *Electrician*, vol. 17, no. 3, pp. 287–291, Sep. 2023, doi: https://doi.org/10.23960/elc.v17n3.2491.

[9] Ltd, Cyberbotics. "Cyberbotics: Robotics Simulation with Webots." *Www.cyberbotics.com*,https://www.cyberbotics.com/#webots

[10] Norton, Tom, and Cyberbotics Ltd. "Webots Webots-2019." *Www.cyberbotics.com*, 18 Dec. 2018, www.cyberbotics.com/doc/blog/Webots-2019-a-release.

[11] "Cyberbotics": ,*www.cyberbotics.com*. https://www.cyberbotics.com/doc/guide/epuck?version=cyberbotics:R2019a. Accessed 18 June 2024.

[12] L. Zadeh, "Fuzzy logic," *Scholarpedia*, vol. 3, no. 3, p 1766, 2008, doi: https://doi.org/10.4249/scholarpedia.1766 .

[13] V. Novák, I. Perfilieva, and J. Mockor, *Mathematical Principles of Fuzzy Logic*. Springer Science & Business Media, 2012.

[14] M. Z. Andrekha and Y. Huda, "Deteksi Warna Manggis Menggunakan Pengolahan Citra dengan Opencv Python," *Voteteknika (Vocational Teknik Elektronika dan Informatika)*, vol. 9, no. 4, p. 27, Dec. 2021, doi: https://doi.org/10.24036/voteteknika.v9i4.114251.

[15] D. Putra, *Pengolahan Citra Digital*. Yogyakarta: Penerbit Andi, 2010.

[16] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. New York, Ny: Pearson, 2018.

[17] Rd. Kusumanto and Alan Novi Tompunu, "Pengolahan Citra Digital Untuk Mendeteksi Obyek Menggunakan Pengolahan Warna Model Normalisasi Rgb," *Semantik*, vol. 1, no. 1, Apr. 2011.

[18] M. Loesdau, S. Chabrier, and A. Gabillon, "Hue and Saturation in the RGB Color Space," *Lecture Notes in Computer Science*, pp. 203–212, 2014, doi: https://doi.org/10.1007/978-3-319-07998-1_23.

[19] A. Koschan and M. Abidi, *Digital Color Image Processing*. John Wiley & Sons, 2008.

[20] Ahmad Fashiha Hastawan, Risma Septiana, and Yudi Eko Windarto, "Perbaikan Hasil Segmentasi HSV Pada Citra Digital Menggunakan Metode Segmentasi RGB Grayscale," *Edu Komputika Journal*, vol. 6, no. 1, pp. 32–37, Jun. 2019, doi: https://doi.org/10.15294/edukomputika.v6i1.23025

[21] Contour Detection using OpenCV (Python/C++)," *LearnOpenCV*, Mar. 29, 2021 [Online]Available:https://learnopencv.com/contour-detection-using-opencv-python-c/