Jurnal Elkolind Volume 12, Nomor 2, Juli 2025
DOI: http://dx.doi.org/10.33795/elkolind.v12i2/4014

Robot Beroda Pemindah Barang Otomatis Dengan Metode *Path Planning*

Wahid Hasyim[1], Indrazno Siradjuddin[2], Leonardo Kamajaya[3]
e-mail: wahidnda07@gmail.com, indrazno@polinema.ac.id, leonardo42@polinema.ac.id
[1,2,3]Jurusan Teknik Elektro, Politeknik Negeri Malang, Jalan Soekarno Hatta No.9 Malang, Indonesia

Informasi Artikel

Riwayat Artikel

Diterima 24 Agustus 2023
Direvisi 4 Maret 2024
Diterbitkan 31 Juli 2025

Kata kunci:

Path planning
Algoritma A*
Robot Pemindah Barang

ABSTRAK

Seringkali, memindahkan barang dari satu tempat ke tempat lain memerlukan waktu, proses pemindahan ini juga bisa menimbulkan risiko kecelakaan kerja dan memakan waktu yang cukup lama. Oleh karena itu, diperlukan langkah-langkah untuk meningkatkan efisiensi waktu dalam proses pemindahan barang agar lebih cepat, efektif, dan mengurangi risiko yang mungkin timbul. Oleh karena itu, penggunaan robot sebagai alternatif untuk tenaga manusia dapat meningkatkan efisiensi dan memudahkan proses pemindahan barang. Tujuan dari penelitian ini adalah untuk mengembangkan sebuah robot yang mampu memindahkan barang secara otomatis dengan menggunakan metode *path planning* A*. Algoritma A* digunakan untuk merencanakan jalur terpendek antara titik awal dan titik tujuan dengan mempertimbangkan rute yang ada di sekitar robot. Penelitian ini mencakup topik-topik seperti desain robot beroda, pemrograman Arduino dan *python,* implementasi algoritma A* pada robot, dan juga pengujian serta evaluasi performa robot dalam memindahkan barang. Pengujian dilakukan dengan mendeteksi *aruco marker* dari kamera sebagai *pose estimation,* titik tujuan dan *obstacles* pada robot. Berdasarkan hasil pengujian, robot dapat bergerak memindahkan barang sampai ke tujuan dengan rata-rata error 0.214, dikarenakan robot hanya mengontrol posisi, tidak mengontrol kecepatan motornya juga.

Keywords:

Path planning
algorithm A*
goods moving robots

ABSTRACT

Often, moving goods from one place to another takes time, this moving process can also pose a risk of work accidents and take quite a long time. Therefore, steps are needed to increase time efficiency in the process of moving goods so that it is faster, more effective and reduces risks that may arise. Therefore, using robots as an alternative to human labor can increase efficiency and facilitate the process of moving goods. The aim of this research is to develop a robot that is capable of moving goods automatically using the A path planning method. The A* algorithm is used to plan the shortest path between the starting point and the destination point by considering the existing routes around the robot. This research covers topics such as wheeled robot design, Arduino and Python programming, implementation of the A* algorithm on robots, and also testing and evaluating robot performance in moving goods. Testing is carried out by detecting Aruco markers from the camera as pose estimation, goal points and obstacles on the robot. Based on the test results, the robot can move goods to its destination with an average error of 0.214, because the robot only controls the position, not the speed of the motor.*

Penulis Korespondensi:
Wahid Hasyim,
Jurusan Teknik Elektro,
Politeknik Negeri Malang,
Jl. Soekarno Hatta No. 9, Malang, Jawa Timur, Indonesia, 65141.
Email: wahidnda07@gmail.com
Nomor HP/WA aktif: +6285606339923

p-ISSN: 2356-0533; e-ISSN: 2355-9195

1. PENDAHULUAN

Mobile robot atau yang biasa disebut robot beroda telah diterapkan secara luas untuk membantu pekerjaan manusia. Mobile robot telah banyak digunakan di dunia industri, militer, pertanian, dan masih banyak lainnya yang dapat membantu manusia [1] . Dengan menggunakan algoritma Path Planning dapat digunakan untuk menentukan arah pergerakan dari Mobile Robot dan melakukan perancangan grid lintasan yang sesuai dengan tujuan [2] . Algoritma A* (dibaca "A STAR") adalah algoritma *Best First Search* yang merupakan perpaduan antara *Uniform Cost Search* yang memilih jarak paling kecil dari simpul awal ke simpul berikutnya dan *Greedy-Best First Search* yang menggunakan heuristik atau nilai perkiraan untuk menentukan simpul berikutnya. Algoritma A* ini akan menemukan rute yang *complete* (selalu menemukan solusi (jika memang ada solusinya)) dan optimal [3] .

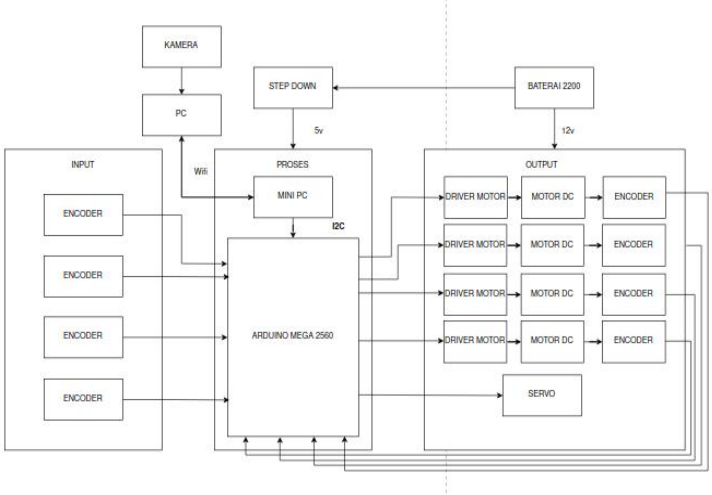
Terdapat beberapa penelitian tentang Algoritma A* yang telah dilakukan, seperti navigasi robot berbasis *path planning* dengan pemulihan jalur otomatis yang menggunakan sensor kompas untuk mengetahui posisi robot [4]. Kemudian terdapat juga penelitian tentang pembacaan jarak dan kecepatan dengan *ArUco Marker* pada sistem koper *follow me* beroda. Koper bergerak otomatis apabila kamera mendeteksi adanya pendanda khusus *(markers)* dari pemiliknya [5] . Dari penelitian tersebut penulis membandingkan keakuratan antara pembacaan posisi dari sensor kompas dengan pendeteksian posisi dari kamera dimana menggunakan penanda khusus pada robot untuk pendeteksiannya.

Sehingga pada penelitian *mobile robot* ini menggunakan kamera untuk mendeteksi *aruco marker* sebagai *pose estimation* robot, *obstacles*, dan titik tujuan robot pada koordinat X dan Y [6] . Berdasarkan latar belakang ini maka penulis mengangkat permasalah robot beroda pemindah barang otomatis dengan metode *path planning*. Pengujian dari penelitian ini dilakukan dengan pembacaan kamera untuk mendeteksi dan merencanakan jalur secara otomatis supaya robot dapat bergerak untuk memindahkan barang ke titik tujuan. Pembahasan dalam penelitian ini terbagi menjadi beberapa bab, bab 2 yang membahas metodologi penelitian yang melibatkan perancangan untuk memperoleh kombinasi nilai dari input sensor dalam sistem, bab 3 yang menyajikan hasil dan analisis dari pengujian yang telah dilakukan, serta bab 4 yang mencakup kesimpulan yang ditarik dari hasil penelitian dan saran untuk penelitian yang akan datang.

2. METODE PENELITIAN

2.1 Diagram Blok Sistem

Diagram blok sistem adalah gambaran visual yang menunjukkan bagaimana setiap komponen dalam suatu sistem berfungsi sebagai input, proses, dan kemudian menghasilkan output.



Gambar 1**:** Diagram Blok Sistem Penelitian

Gambar 1 pada diagram blok menunjukkan bahwa baterai digunakan sebagai sumber daya motor DC, *mini pc* serta arduino mega yang di turunkan tegangannya terlebih dahulu menjadi 5V menggunakan *step down.* Kemudian terdapat kamera yang terhubung dengan pc/laptop berfungsi untuk mendeteksi posisi dan pergerakan

robot menggunan *aruco marker* yang kemudian data dari kamera dikirim ke *mini pc* melalui komunikasi ROS (*robot operating system*) yang terhubung dengan satu alamat IP. Pada input terdapat 4 *rotary encoder* yang berfungsi untuk menghitung jumlah putaran roda pada sistem yang bekerja dengan memanfaatkan 2 *channel output* A (CLK) dan B (DT) untuk menggerakkan roda yang memiliki logika output 0 dan 1, dimana logika tersebut digunakan untuk mengetahui arah putaran dari roda apakah searah jarum jam (*Clockwise*) yang memiliki *output* berbeda A = 1 dan B = 0 dan selalu bergantian tiap gerakan menjadi A = 0 dan B = 1 dan setersunya jika digerakan kembali lagi ke awal *output* dari *channel* A = 1 dan B = 0, atau berlawan jarum jam (*Counter Clockwise*) yang memiliki *output* A = 0 dan B = 0 dan selalu bergantian tiap gerakan menjadi A = 1 dan dan B = 1 dan seterusnya kembali ke awal yaitu A = 0 dan B = 0 ketika digerakkan [7] . Pada tahap proses arduino mega memproses data dari mini pc dengan komunikasi *I2C* yang kemudian arduino mega memberikan sinyal PWM ke motor agar robot bergerak ke titik tujuan. Pada *output* terdapat 4 motor DC sebagai penggerak robot dan juga servo sebagai pencapit barang.

2.2 Sensor *Rotary Encoder*

Rotary encoder pada robot ini berfungsi sebagai membaca nilai putaran kecepatan roda atau RPM (*Rotatition Per Minute*) untuk mengetahui kecepatan perubahan posisi robot. *Encoder* yang digunakan adalah *incremental encoder*, dengan memiliki dua *channel* yaitu *channel* A dan *channel* B. Kedua *channel* ini menghasilkan pulsa, dan keduanya memiliki perbedaan fasa sebesar 90°. Perbedaan fasa ini dimanfaatkan untuk menentukan arah putaran roda, apabila *rotary encoder* bernilai positif maka motor bergerak searah jarum jam (CW) atau *rotary encoder* bernilai *negative* maka motor bergerak berlawanan arah jarum jam (CCW).

Arduino Mega mengolah sinyal pulsa yang diterima dari rotary encoder menjadi kecepatan putaran dalam satuan RPM. Proses pengolahan tersebut dilakukan berdasarkan akumulasi pulsa *rotary encoder* dengan seiring berjalanya waktu. Arduino mega menghitung pulsa dari *rotary encoder* yang melakukan transisi dari logika "0" ke logika "1" atau dapat disebut transisi *rising*. Untuk menghitung jumlah sinyal pulsa dari *rotary encoder* dapat dihitung dengan rumus berikut.

$$N = PPR_{Motor}\ x\ Gear_{Ratio\ Motor} \tag{1}$$

Jika dilihat dari datasheet motor JGA25-370, spesifikasinya adalah:

1. *Gear Ratio* = 1 : 21.3
2. *Speed* = 220 – 280 RPM
3. *Encoder Out* = 11 PPR

Maka:

$$N = 11\ x\ 21.3$$
$$N = 234.3$$

Untuk perhitungan sinyal pulsa rotary encoder menjadi kecepatan RPM adalah sebagai berikut.

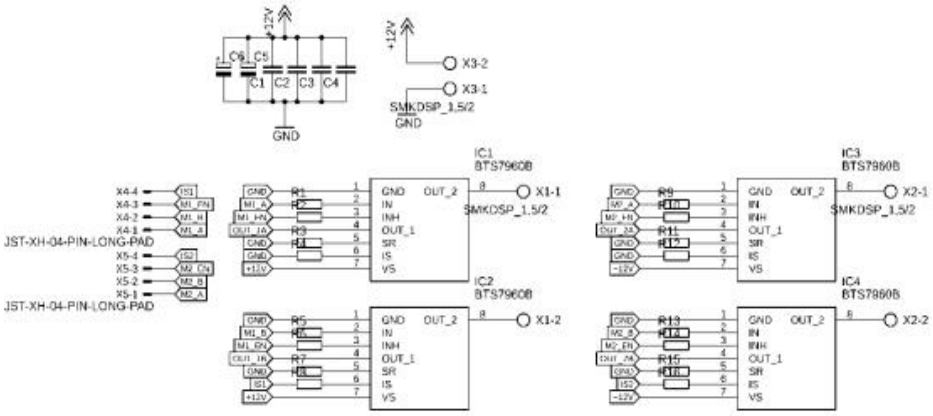
$$RPM = \frac{\sum Pulsa\ x \frac{60}{\Delta t}}{PPR} \tag{2}$$

Yang kemudian arduino mega akan mengirimkan hasil perhitungan kecepatan RPM tersebut ke raspberry pi untuk dijadikan kecepatan linear robot (v) menggunakan persamaan berikut.

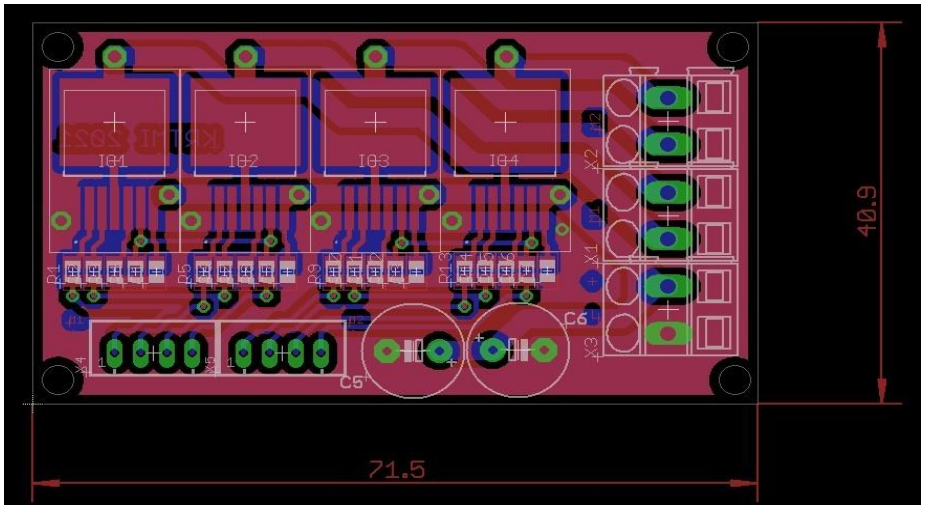
$$v = RPM\ x\ \frac{2\pi r}{60} \tag{3}$$

2.3 Rancangan Driver Motor BTS7960

Driver motor BTS7960 merupakan suatu perangkat elektronik yang digunakan untuk mengontrol motor DC. BTS7960 adalah driver motor H-bridge yang dilengkapi dengan dua *channel*, menggunakan empat IC BTS7960 di mana setiap IC telah menyatukan 1 MOSFET-P dan 1 MOSFET-N. Sesuai dengan informasi pada datasheet BTS7960, frekuensi gelombang PWM diperoleh dengan mempertimbangkan waktu *rise time* atau *fall time* dari BTS7960 saat melakukan *switching*. Durasi *rise time* atau *fall time* dapat diatur dengan memasang resistor yang memiliki nilai resistansi sesuai dengan petunjuk *datasheet*, terhubung pada pin SR (*slew rate*) yang terhubung dengan *ground*. Penyetelan *slew rate* berfungsi untuk mengoptimalkan disipasi daya MOSFET ketika MOSFET melakukan *switching*.



Gambar 2**:** Desain skematik driver motor BTS7960



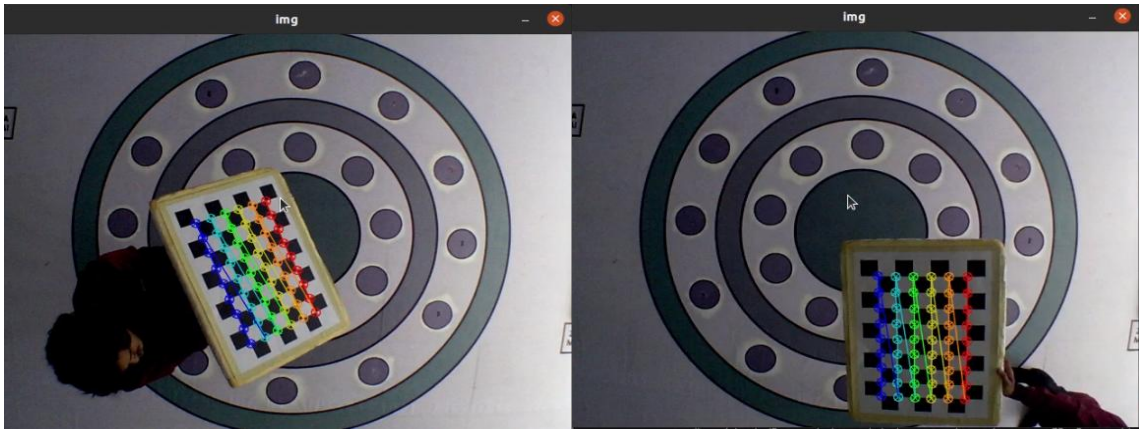
Gambar 3**:** Desain *layout* PCB driver motor BTS7960

Gambar 2 dan gambar 3 merupakan desain skematik dan desain *layout* PCB *driver motor* BTS7960. Kontrol Motor DC *High Current* pada *driver* motor DC ini dapat mengeluarkan arus hingga 43A dengan rangkaian *H-Bridge* sebagai pengendali arah putar motor DC. BTS7960 tidak memisahkan *power ground* dan *logic ground*. Oleh karena itu dianjurkan untuk memastikan laju perubahan tegangan antara koneksi *ground* dengan resistor. Jika BTS 7960 digunakan pada *H-bridge*, *offset* tegangan antara pin GND dari perangkat yang berbeda juga harus berukuran kecil. Kapasitor keramik dari VS ke GND yang dekat dengan setiap perangkat disarankan untuk disediakan arus untuk fase peralihan melalui jalur induktansi rendah supaya mengurangi kebisingan dan pantulan *ground*. Pada BTS7960 sudah

terdapat pengaman bila terjadi *error* berupa perlindungan arus balik. Pengaman ini berupa MOSFET dengan penyearah dioda.

2.4 Kalibrasi Kamera

Kalibrasi kamera dilakukan untuk memahami hubungan antara citra yang dihasilkan oleh kamera dengan objek di dunia nyata dengan menentukan parameter intrinsik dan ekstrinsik dari kamera sehingga dapat digunakan untuk mengoreksi distorsi dan mengkonversi koordinat dalam citra menjadi koordinat dunia nyata (3D).



Gambar 4: Contoh Pengambilan Sampel

Pada contoh gambar diatas merupakan proses kalibrasi yang dilakukan dengan mengambil sampel dari pola papan catur yang dilakukan dibeberapa sisi dan sudut kamera dimana pola papan catur harus terlihat jelas semua. Semakin banyak gambar yang diambil, maka akan semakin bagus hasil kalibrasi yang akan didapatkan. Selanjutnya dalam proses kalibrasi digunakan pseudocode sebagai berikut.

```
if img is None:
    print("error cali5")
else:
    h,  w = img.shape[:2]
    newCameraMatrix, roi = cv.getOptimalNewCameraMatrix(
    cameraMatrix, dist, (w,h), 1, (w,h))
    dst = cv.undistort(img, cameraMatrix, dist, None,
    newCameraMatrix)
    x, y, w, h = roi
    dst = dst[y:y+h, x:x+w]
    cv.imwrite('caliResult1.png', dst)
    mapx, mapy = cv.initUndistortRectifyMap(cameraMatrix,
    dist, None, newCameraMatrix, (w,h), 5)
    dst = cv.remap(img, mapx, mapy, cv.INTER_LINEAR)
    x, y, w, h = roi
    dst = dst[y:y+h, x:x+w]
    cv.imwrite('caliResult2.png', dst)
```

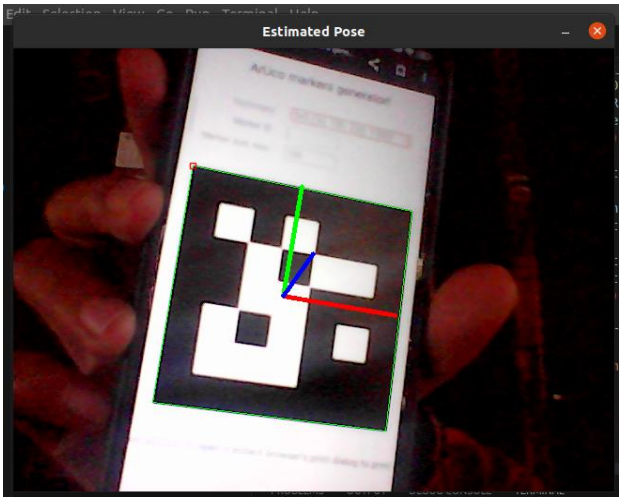
Gambar 5: Contoh Program Kalibrasi

Dari potongan program diatas, program akan memproses semua sampel foto kalibrasi untuk menghitung matriks kamera dan koefisien distorsi menggunakan citra dari papan kalibrasi catur. Program akan membaca citra dari folder 'images' yang mengandung citra-citra papan kalibrasi, mengubah citra ke skala abu-abu (*gray*) untuk

mendeteksi sudut titik kalibrasi menggunakan cv.findChessboardCorners(). Jika sudut titik kalibrasi ditemukan di citra, maka objek titik kalibrasi dan sudut titik kalibrasi akan ditambahkan ke dalam objpoints dan imgpoints masing-masing untuk mendapatkan matriks kamera dan koefisien distorsi. yang terakhir menghitung kesalahan reproyeksi sebagai evaluasi kualitas kalibrasi. Kesalahan reproyeksi adalah perbedaan antara titik sudut kalibrasi yang diestimasi kembali dengan titik sudut yang sebenarnya pada citra.

2.5 Pendeteksian Aruco

Aruco marker merupakan tanda pengenal visual yang digunakan dalam pengenalan objek dan penglihatan komputer. Terdapat 3 jenis aruco yang akan digunakan pada penelitian ini. Pertama *aruco* dengan *dictionary* 4x4 berfungsi untuk mengetahui titik tujuan robot. Kedua *aruco* dengan *dictionary* 7x7 berfungsi sebagai *pose estimation* robot. Ketiga *aruco* dengan *dictionary* 6x6 berfungsi sebagai *obstacles.*



Gambar 6: Contoh Pendeteksian Aruco

Gambar 6 merupakan salah satu contoh pendeteksian kamera terhadap aruco *dictionary* 7x7 sebagai *pose* robot dengan menghitung posisi dan orientasi objek dalam ruang 3D, serta menampilkan informasi tersebut di atas gambar.

2.6 Perencanaan Algoritma A*

Pembuatan algoritma A* dilakukan setelah kalibrasi kamera dan pendeteksian aruco selesai, kalibrasi kamera untuk mengetahui data *intrinsik* dan *distorsi* kamera yang menjadi *input* dari fungsi aruco *pose* robot. *Frame* kamera akan menampilkan sebuah *path*(jalur) ke titik tujuan apabila kamera mendeteksi aruco *pose* robot dan aruco titik tujuan. Apabila kamera mendeteksi sebuah *obstacles* maka algoritma A* akan merubah *path* (jalur) lain yang terpendek untuk sampai ke titik tujuan dengan menghindari aruco *obstacles*. Perencanaan path menggunakan algoritma A* dapat dihitung dengan rumus:

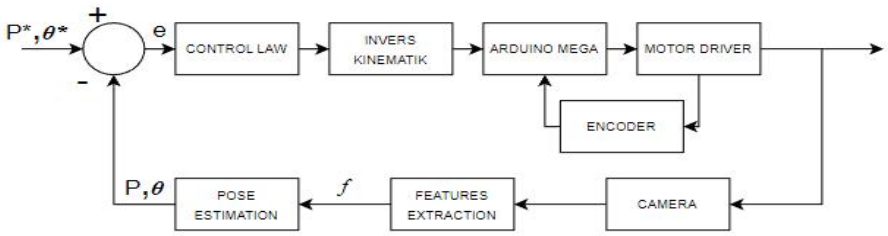
$$f(n) = g(n) + h(n) \tag{4}$$

$$g(n) = |(x_{node} - x)| + |(y_{node} - y)| \tag{5}$$

$$h(n) = |(x_{goal} - x)| + |(y_{goal} - y)| \tag{6}$$

dimana:
f(n) = Akumulasi dari titik tengah ke titik tujuan
g(n) = Jarak dari titik selanjutnya
h(n) = Jarak dari titik tujuan

2.7 Diagram Blok Kontrol

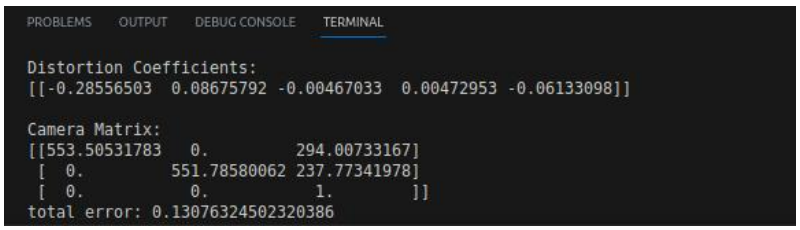


Gambar 7: Diagram Blok Kontrol

Dalam diagram blok control diatas, terdapat *control law* yang digunakan sebagai pendekatan eksponensial dengan proporsi konversi yang kemudian diteruskan ke *inverse* kinematika. Pada *inverse* kinematika mengirimkan sinyal ke arduino mega untuk memproses dan mengirimkan sinyal PWM ke driver motor. Setelah motor bergerak, pergerakan robot terdeteksi oleh *encoder* sejauh berapa derajat putaran (RPM) roda yang kemudian dikembalikan lagi ke Arduino mega. kamera mendeteksi *aruco marker* yang berada di robot sebagai *pose estimation* robot. P dan θ merupakan posisi aktual robot saat ini, sedangkan p* dan θ* merupakan titik tujuan dar robot. Nilai-nilai tersebut digunakan sebagai umpan balik dan dibandingkan dengan *setpoint*, yang kemudian menghasilkan nilai kesalahan (*error*).

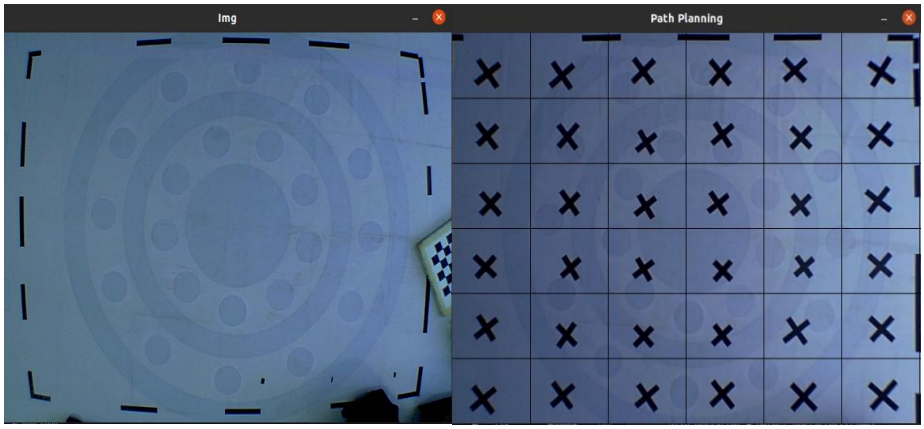
3. HASIL DAN PEMBAHASAN

3.1 Pengujian Kalibrasi Kamera



Gambar 8: Hasil dari Kalibrasi Kamera

Dari hasil pengujian kalibrasi kamera diatas didapatkan nilai koeffisien distorsi dan nilai matriks kamera adalah [-0.28556503, 0.08675792, -0.00467033, 0.00472953, -0.06133098] sebagai nilai koeffisien distorsi dan [[553.50531783, 0, 294.00733167], [0, 551.78580062, 237.77341978], [0, 0, 1]] sebagai nilai matriks kamera dengan total error 0.1307632.

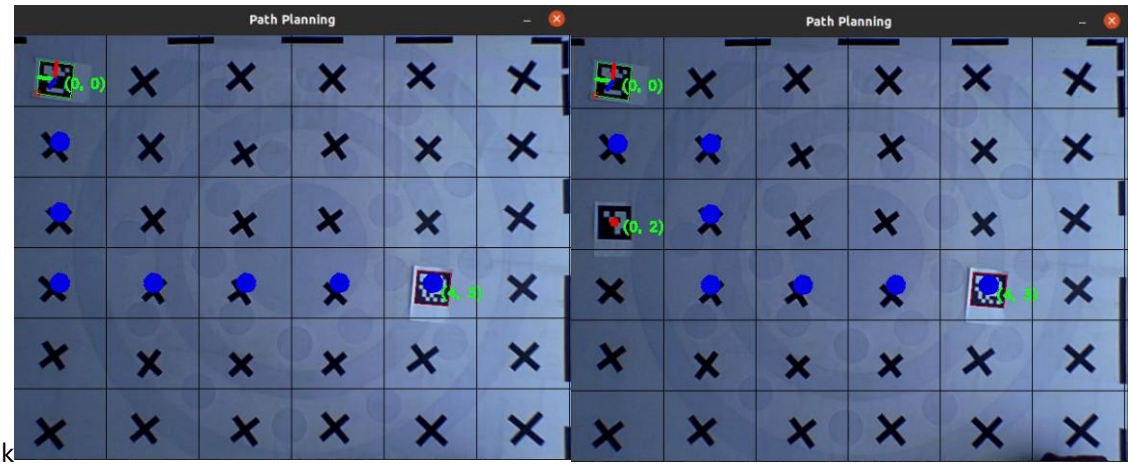


Gambar 9: gambar sebelum dan sesudah kamera dikalibrasi

Pada gambar diatas merupakan perbandingan gambar antara kamera sebelum terkalibrasi (kiri) dengan kamera yang sudah terkalibrasi (kanan). Untuk mengubah frame menjadi linier diperlukan nilai distorsi koefffisien dan nilai kamera matrix dari hasil kalibrasi, sehingga kamera yang awalnya melengkung menjadi linier. Untuk mendapatkan nilai-nilai tersebut dibutuhkan beberapa foto sampel dari papan catur untuk mengoreksi distorsinya. Gambar yang belum terkalibrasi terdapat sedikit bentuk melengkung pada frame kamera, dan akan terlihat lurus ketika telah dikalibrasi. Dari hasil pengujian yang dilakukan, dapat disimpulkan bahwa nilai koeffisien distorsi dan nilai matriks kamera yang didapatkan dari proses kalibrasi sudah tepat dengan nilai akurasi error dibawah 0,2% untuk membuat gambar asli yang masih melengkung menjadi gambar yang lurus.

3.2 Pengujian Pencarian *Path* Otomatis

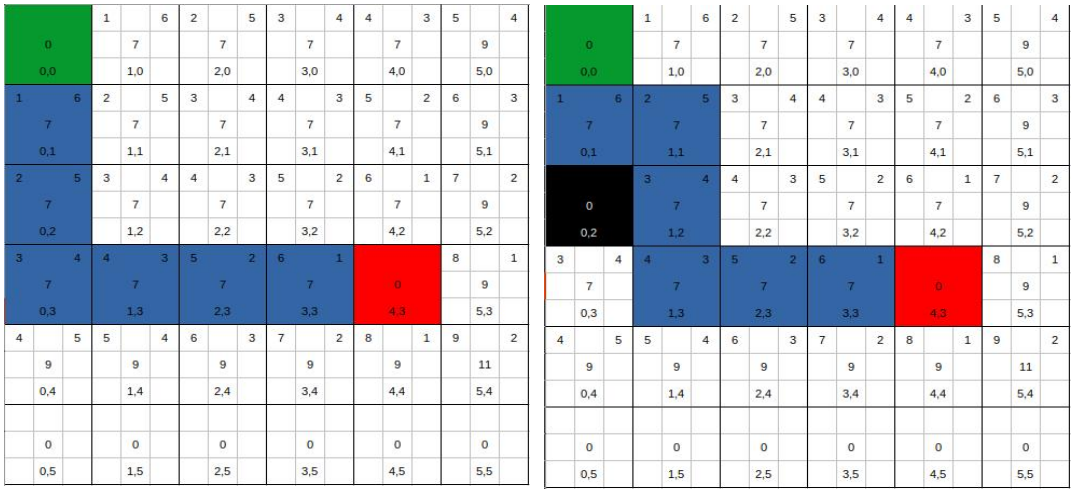
Pengujian ini dilakukan untuk menentukan *path* terpendek ke titik tujuan secara otomatis. *Path* akan muncul apabila kamera mendeteksi aruco pose robot dan aruco titik tujuan robot. Apabila kamera mendeteksi adanya *obstacles, algoritma A* akan merubah path* untuk menghindari *obstacles* tersebut. Berikut contoh pencarian *path* otomatis oleh algoritma A*:



Gambar 10: Contoh Pencarian *Path*

p-ISSN: 2356-0533; e-ISSN: 2355-9195

Dari rumus perencanaan *path menggunakan algoritma A** diatas, didapatkan perhitungan *path* pada gambar dibawah ini.



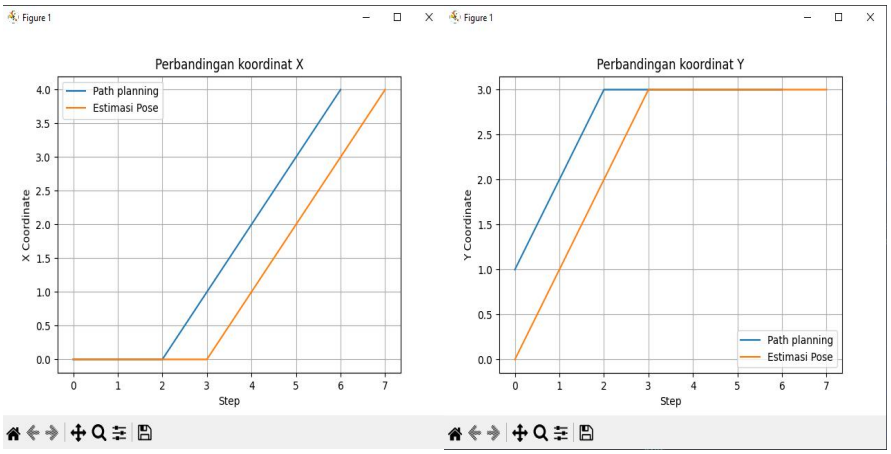
Gambar 11: Contoh Perhitungan *path*

Dari hasil perhitungan diatas, dapat diketahui nilai-nilai f(n) yang terkecil untuk menjadi *list path* yang harus di tempuh. Perencanaan *path* mendahulukan *error* y sampai 0 terlebih dahulu kemudian menghitung *error* x sampai ke titik tujuan. Oleh sebab itu kenapa pada contoh perencanaan *path* diatas tidak memilih jalur yang horizontal terlebih dahulu ketika terdapat *obstacles.*

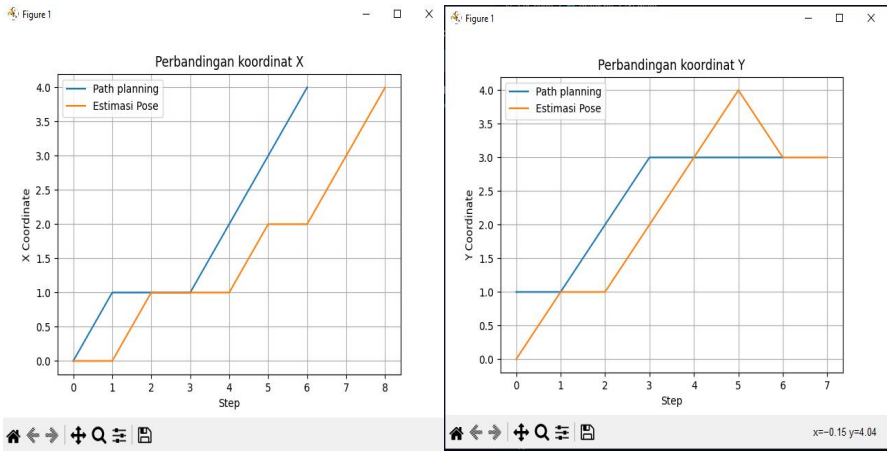
Dalam algoritma A* dipilih berdasarkan prinsip bahwa algoritma ini berusaha untuk menemukan *path* dengan total biaya (nilai f(n)) yang paling rendah dari titik awal ke titik tujuan. Path yang dipilih meminimalkan total biaya dengan mempertimbangkan nilai g(n) (biaya sejauh ini) dan h(n) (estimasi biaya sisa ke titik tujuan). *Path* tersebut dipilih karena pada setiap langkahnya, algoritma A* memilih langkah dengan nilai f(n) terkecil yang tersedia. Setiap langkah menambahkan 1 ke g(n) (karena kita bergerak satu langkah dalam grid) dan menghitung h(n) menggunakan *heuristic* jarak Manhattan dari posisi saat ini ke titik tujuan.

3.3 Pengujian Pergerakan Robot

Pada pengujian ini dilakukan untuk mengetahui apakah pergerakan robot sudah sesuai dengan data koordinat *step by step* dari perencanaan path algoritma A*



Gambar 12: Error pergerakan (x,y) Tanpa *Obstacles*



Gambar 13: Error pergerakan (x,y) Dengan 1 *Obstacles*

Error (x, y) diatas di dapatkan dari $E_x = X_{path} - X_{robot}$ dan $E_y = y_{path} - y_{robot}$ Dimana:

1. $E_x > 0$ robot bergerak maju
2. $E_x < 0$ robot bergerak mundur
3. $E_y > 0$ robot bergerak kanan
4. $E_y < 0$ robot bergerak kiri

Hasil dari data pergerakan robot diatas masih terdapat error pada saat uji coba adanya *obstacles* dengan rata-rata error 0.214 dikarenakan tidak mengontrol kecepatan motor. Robot hanya menggunakan kontrol posisi dari error (x, y), sehingga robot sering mengalami selip pada roda omni.

4. KESIMPULAN

Dari hasil pengujian diatas untuk kalibrasi sudah mendekati linier dengan error 0.13. Untuk perencanaan path perhitungan A* dengan perhitungan secara langsung hasilnya sama, hasil perencanaan path mendahulukan error y sampai 0 terlebih dahulu kemudian menghitung error x sampai ke titik tujuan. Tetapi untuk pergerakan robot terdapat error pada saat uji coba adanya *obstacles* dengan rata-rata error 0.214, karena robot hanya mengontrol error posisi saja tidak mengontrol kecepatan juga, akibatnya robot sering terjadi selip. Oleh karena itu untuk pengembangan lebih lanjut disarankan menggunakan kontrol kecepatannya juga, supaya pergerakan lebih bagus dan terarah.

DAFTAR PUSTAKA

[1] U. G. Mada, "Smoothed Gradient Descent A-Star Algorithm untuk Mobile Robot Path Planning SYAIFUL ARDY GUNAWAN, Dr. Eng. Adha Imam Cahyadi, S.T., M.Eng. ; Dr. Ir. Bondhan Winduratna, M.Eng.," pp. 13–14, 2019.

[2] D. F. Adryady, A. Prasetyo, H. P. Shatyaziamawan, and A. S. Priambodo, "Implementasi Algoritma Path Planning dan Mapping Arena pada Mobile Robot," *Jurnal Teknik Elektro dan Komputer TRIAC*, vol. 9, no. 2, pp. 41–45, 2022, [Online]. Available: https://journal.trunojoyo.ac.id/triac/article/view/13215

[3] D. Hermanto and S. Dermawan, "Penerapan Algoritma A-Star Sebagai Pencari Rute Terpendek pada Robot Hexapod," *Jurnal Nasional Teknik Elektro*, vol. 7, no. 2, p. 122, 2018, doi: 10.25077/jnte.v7n2.545.2018.

[4] A. Rafsanjani, "NAVIGASI ROBOT BERBASIS PATH PLANNING DENGAN PEMULIHAN JALUR OTOMATIS," 2017.

[5] A. Triwahyudin, H. K. Safitri, and M. Fauziyah, "Pembacaan Jarak dan Kecepatan dengan ArUco Marker pada Sistem Koper Follow Me Beroda," *Majalah Ilmiah Teknologi Elektro*, vol. 21, no. 1, p. 97, Jul. 2022, doi: 10.24843/mite.2022.v21i01.p14.

[6] W. A. Maulana, I. T. Winarno, I. Siradjuddin, and D. Ph, "Navigasi Pergerakan Robot Berdasarkan Rekam Data Sensor Odometry," vol. x, no. 9.

[7] T. D. PUTRI, W. SUGENG, and F. RAMDANI, "Sistem Pembayaran Elektronik pada Transportasi Angkutan Kota menggunakan Rotary Encoder," *MIND Journal*, vol. 6, no. 1, pp. 1–15, 2021, doi: 10.26760/mindjournal.v6i1.1-15.

[8] M. H. Fadhlurrahman, B. Dirgantara, and A. Mulyana, "Implementasi Dan Analisis Penggunaan Algoritma A-Star Dengan Prioritas Pada Pemilihan Rute Lintas Kendaraan Roda Dua," *Bandung : Universitas Telkom*, pp. 1–8, 2013.

[9] A. K. Guruji, H. Agarwal, and D. K. Parsediya, "Time-efficient A* Algorithm for Robot Path Planning," *Procedia Technology*, vol. 23, pp. 144–149, 2016, doi: https://doi.org/10.1016/j.protcy.2016.03.010.

[10] G. Klančar, A. Zdešar, S. Blažič, and I. Škrjanc, *Wheeled Mobile Robotics Wheeled Mobile*. 2017. [Online]. Available: https://www.sciencedirect.com/book/9780128042045/wheeled-mobile-robotics

[11] G. K. Nugraha, I. Setiawan, and H. Afrisal, "Perancangan Dan Pengendalian Differential Drive Robot Dengan Mengaplikasikan Metode a-Star," *Transient: Jurnal Ilmiah Teknik Elektro*, vol. 10, no. 4, pp. 559–565, 2021, doi: 10.14710/transient.v10i4.559-565.

[12] T. A. Putra, M. Muliady, and D. Setiadikarunia, "Navigasi Indoor Berbasis Peta pada Robot Beroda dengan Platform Robot Operating System," *Jetri : Jurnal Ilmiah Teknik Elektro*, vol. 17, no. 2, pp. 121–144, 2020, doi: 10.25105/jetri.v17i2.5447.

[13] I. Siradjuddin *et al.*, "DESAIN DAN PEMODELAN KONTROL KINEMATIK PERGERAKAN ROBOT BERODA DENGAN MENGGUNAKAN 6 RODA OMNI-WHEELS," vol. 18, no. 1, 2020, [Online]. Available: http://eltek.polinema.ac.id/index.php/eltek