SISTEM KOORDINASI UNTUK PENGAMBILAN KEPUTUSAN PADA ROBOT SEPAKBOLA BERODA

Zaky Wafa Albahari [1], Akhmad Musafa[2], Sujono[3], dan Indra Riyanto[4]
Program Studi Teknik Elektro, Fakultas Teknik Universitas Budi Luhur Jakarta, Indonesia
[1]zakywafa99@gmail.com, [[2]akhmad.musafa;[3]sujono, [4]indra.riyanto]@budiluhur.ac.id

ABSTRAK

Pertandingan robot sepakbola beroda memiliki permasalahan yang salah satu diantaranya adalah kemampuan berkoordinasi. Karena dalam pertandingan robot sepakbola beroda satu tim terdiri dari beberapa robot, maka kekompakan merupakan faktor penentu untuk menghasilkan permainan yang baik. Untuk menghasilkan kekompakan yang baik, diperlukan koordinasi antar robot. Penelitian ini dilakukan dengan mensimulasikan tiga unit robot yaitu satu unit robot koordinator dan dua unit robot anggota, dengan tambahan 3 robot lawan yang diam. Konsep utama penelitian ini adalah tindakan yang dilakukan oleh setiap robot dalam tim merupakan hasil keputusan dari robot koordinator. Robot anggota memiliki dua pekerjaan dalam tim, yaitu mengirim semua data situasi pertandingan yang terbaca oleh sensornya ke robot koordinator, dan menjalankan perintah untuk masing-masing robot yang ditentukan oleh robot koordinator. Robot koordinator memberikan perintah kepada masing-masing robot berdasarkan semua data kondisi pertandingan yang dikumpulkan oleh semua robot. Untuk menghindari robot lain dalam tim saat bergerak, digunakan aturan Sin dan Cos. Sedangkan Dijkstra's Algorithm digunakan untuk menghindari robot lawan sekaligus path planning. Hasil dari penelitian ini robot berhasil bergerak secara terkoordinir yaitu hanya menggerakkan robot yang paling cocok untuk malakukan aksi. Sedangkan robot yang lain cukup bersiaga menunggu kondisi yang cocok baginya untuk melakukan aksi. Selain itu robot juga dapat bergerak tanpa menabrak robot lain baik robot dalam tim maupun robot lawan. Pada berbagai kasus yang dilakukan dalam pengujian, robot memerlukan waktu berkisar dari 3 sampai 5 detik untuk melakukan tugasnya.

Kata kunci: robot sepakbola beroda; sistem koordinasi; kendali terpusat; penggabungan data; pengambilan keputusan; dijkstra's algorithm;

ABSTRACT

Wheeled robot soccer matches have problems, one of which is the ability to coordinate. Because in a one-wheeled soccer robot match a team consists of several robots, then cohesiveness is the deciding factor to produce a good game. To produce good cohesiveness, coordination between robots is needed. In this final project a coordination system for decision making in a wheeled soccer robot team is designed. This research was conducted by simulating three robot units, one coordinator robot unit and two member robot units, with the addition of 3 silent robot opponents. The main concept of this research is the action taken by each robot in the team is the result of the decision of the coordinating robot. Robot members have two jobs in the team, namely sending all match situation data read by the sensor to the coordinating robot, and executing commands for each robot that is determined by the coordinating robot. The coordinating robot gives commands to each robot based on all match condition data collected by all robots. To avoid other robots in the team when moving, sin and cos rules are used. Meanwhile, to avoid opposing robots as well as path planning using Dijkstra's Algorithm. The results of this study the robot managed to move in a coordinated manner that is only moving the robot that is most suitable for action. While the other robots are prepared to wait for conditions suitable for him to take action. Besides robots can also move without crashing into other robots both robots in the team and the robot opponent. In various cases carried out in testing, the robot takes from 3 to 5 seconds to do its job.

Keywords— wheeled soccer robot; coordination system; centralized control; data integration; decision making; dijkstra's algorithm

I. PENDAHULUAN

Pertandingan robot sepakbola beroda merupakan pertandingan antara dua tim robot yang menggunakan peraturan mirip dengan peraturan permainan sepakbola pada umumnya. Satu tim robot sepakbola terdiri dari beberapa robot sepakbola beroda yang terdiri dari beberapa penyerang dan pemain belakang dan satu kiper. Dalam jalannya pertandingan, data situasi pertandingan yang dapat diambil oleh robot melalui sensor cukup terbatas. Hal ini dapat disebabkan karena kurangnya cakupan sensor dan juga karena robot saling menghalangi pandangan sensor satu dama lain. Sehingga jika

masing-masing robot mengambil keputusan hanya berdasarkan dari data kondisi lingkungan pertandingan yang didapatkannya sendiri secara masing-masing, maka akan sulit menghadapi kondisi lingkungan pertandingan yang tidak dilihatnya. Selain itu, robot juga akan sulit melakukan koordinasi dengan baik. Untuk dapat melakukan koordinasi dengan baik maka robot harus dapat saling berkomunikasi satu sama lain.

Pada penelitian yang telah dilakukan tentang sistem untuk merencanakan rute pergerakan sebuah robot sepakbola beroda dengan membuat sebuah simulasi sistem dan diuji pada sistem yang sebenarnya. Hasilnya rute pergerakan yang dihasilkan oleh sistem aslinya lebih kecil ukurannya jika dibandingkan dengan rute pergerakan yang dihasilkan oleh sistem simulasi [1]. Pada paper yang membahas tentang algoritma perencanaan gerakan robot sepakbola beroda, telah dibahas metode pembuatan rencana jalur gerakan dengan menggunakan metode Theta*. Penelitian dilakukan melalui simulasi secara tiga dimensi, hasilnya menunjukkan bahwa metode Theta* dapat menghasilkan path planning yang efektif [2]. Pada jurnal yang membahas tentang algoritma perencanaan ganda pada robot sepakbola telah dibahas kendali robot sepakbola menggunakan algoritma if-then rules dan robot menghasilkan perintah output secara masing masing meskipun telah mendapatkan semua data lingkungan [3].Pada jurnal yang berjudul "Robot soccer control using behaviour trees and fuzzy logic" telah dibahas mengenai pengambilan keputusan robot sepakbola dengan metode fuzzy logic yang dilakukan oleh masing masing robot dengan saling bertukar informasi melalui jaringan wifi [4].Pada paper yang membahas tentang pemrograman pusat kendali pada robot sepakbola beroda telah dibahas tentang penggunaan pemrograman dengan bahasa python untuk membuat pusat kendali dari tiga robot sepakbola beroda. Pusat kendali ini digunakan untuk menyalurkan perintah yang diberikan oleh wasit [5]. Pada paper yang membahas tentang sistem koordinasi berbasis perilaku pada robot sepakbola beroda, telah dibahas penggunaan Finite State Machine dengan bantuan fuzzy logic untuk membantu dalam pengambilan keputusan untuk berkoordinasi antar subsistem. Penelitian ini juga merupakan kelanjutan dari sebuah paper yang berjudul WiFi Data Communication System Design for Wheeled Soccer Robot Controller [6]. Paper ini membahas tentang pembentukan sistem komunikasi WiFi untuk pengendali robot sepak bola beroda. Penelitian ini mengimplementasikan kemampuan sebuah regu robot sepak bola yang dapat saling berkomunikasi dengan menggunakan komunikasiWiFi untuk melakukan proses pengendalian secara terpusat.

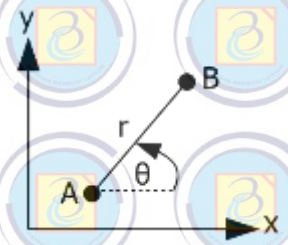
Dalam penelitian ini driancang metode pengambilan keputusan robot sepakbola berdasarkan pembacaan input yang terbaca oleh masing-masing robot dalam tim, lalu diolah oleh robot koordinator untuk menghasilkan perintah untuk masing-masing robot dalam tim.

II. LANDASAN TEORI

Pada bab ini, diuraikan secara ringkas dan jelas kajian teori dasar yang digunakan dalam pembahasan sistem dan dalam pengujian dan analisa data yang dilakukan. Teori yang digunakan dalam pembuatan algoritma dalam sistem ini meliputi aturan sin dan cos sebagai dasar pengolahan titik koordinat dan Dijkstra's Algorithm sebagai pengolah data titik koordinat yang didapatkan. Perangkat lunak yang diperlukan dalam sistem ini meliputi Gazebo Simulator dan Robot Operating System (ROS). Gazebo Simulator berperan sebagai sensor, aktuator, fisik serta lingkungan bagi robot. Sedangkan Robot Operating System berperan sebagai pengendali bagi robot.

A. Aturan Sin Dan Cos

Aturan sin dan cos diperlukan untuk mencari titik koordinat akhir dari sebuah garis dengan diketahui koordinat titik awal garis, sudut garis, dan panjang garis [7]. Untuk mencari titik koordinat akhir dari sebuah garis dengan diketahui koordinat titik awal garis, sudut garis, dan panjang garis seperti ditunjukkan pada Gambar 1.



Gambar 1. Garis yang dibentuk oleh dua titik pada diagram kartesian

Jika koordinat A, panjang r dan sudut θ diketahui, maka koordinat B dapat dihitung dengan rumus:

$$B(x) = A(x) + r * cos\,\theta \qquad (1)$$

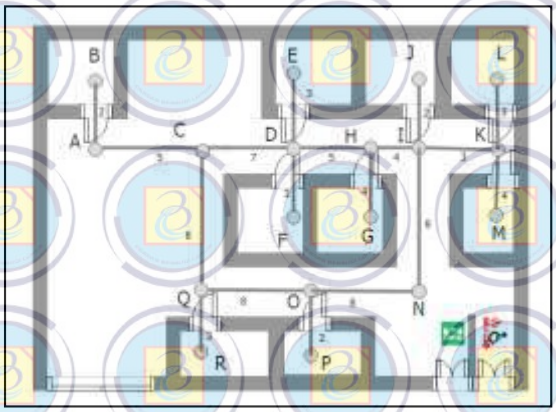
$$B(y) = A(y) + r * sin\,\theta \qquad (2)$$

dengan:

A(x,y) = koordinat awal garis
B(x,y) = koordinat akhir garis
r = panjang garis
θ = sudut yang ingin dibentuk

B. Dijkstra's Algorithm (Algoritma Dijkstra)

Dijkstra's algorithm digunakan untuk mencari rute terpendek untuk mencapai titik target dengan rute yang berupa titik-titik yang dapat dilalui. [8]. Pernah dilakukan penelitian yang menggunakan Dijkstra's algorithm untuk mencari rute terpendek antara ruangan satu dengan ruangan lainnya [9]. Penelitian ini diilustsrasikan pada Gambar 2.



Gambar 2. Ilustrasi penggunaan Dijkstra's algorithm untuk mencari rute terpendek antara ruangan satu dengan ruangan lainnya [9]

Dari sebuah ilustrasi yang digambarkan pada Gambar 2, dapat dilihat untuk mencari rute terpendek untuk berpindah dari suatu ruangan ke ruangan lainnya perlu ada titik-titik koordinat yang dapat dilalui. Kemudian titik-titik ini dibuat jalurnya untuk membentuk sebuah peta rute yang dapat dilalui. Rute yang dipilih adalah rute yang memiliki jarak terdekat bagi posisi awal untuk mencapai tujuannya.

C. Integerasi Gazebo Simulator Dengan Robot Operating System (ROS)

Sistem ini dibentuk dan dijalankan dalam perangkat lunak yaitu Gazebo Simulator yang diintegerasikan dengan Robot Operating System (ROS) [10].

1) Gazebo Simulator

Gazebo Simulator merupakan sebuah perangkat lunak yang dikhususkan untuk melakukan simulasi robot. Gazebo Simulator biasanya diintegerasikan dengan perangkat lunak lain yang berfungsi sebagai pengendali bagi robot [11].

2) Robot Operating System (ROS)

Robot Operating System merupakan sebuah perangkat lunak yang digunakan untuk membuat perangkat lunak pengoprasian robot. Robot Operating System bertujuan membuat perangkat lunak pengoprasian robot yang kuat dan bisa digunakan untuk segala jenis robot [12].

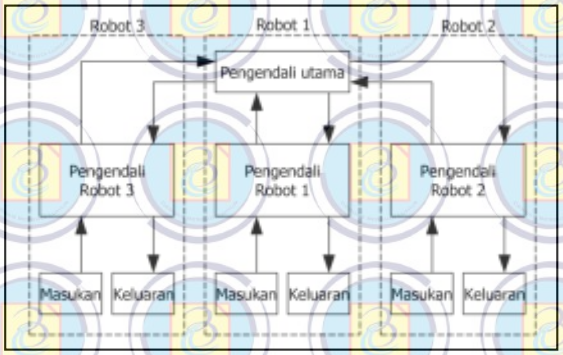
III. PERANCANGAN SISTEM

Pada bab ini menjelaskan mengenai struktur sistem simulasi dan algoritma yang ditanamkan dalam sistem koordinasi robot. Untuk mengaplikasikan sistem ini diperlukan tiga bagian utama.

1. Sistem akuisisi data yang dapat cukup cepat dalam ranah pemrosesan data. Data yang perlu didapatkan yaitu posisi serta orientasi masing-masing robot, posisi bola, dan robot posisi lawan.
2. Sistem komunikasi data yang cukup cepat dalam ranah pemrosesan data. Karena semua data masukan dan data keluaran semuanya harus dikirim antar robot dengan robot koordinator sebagai pusat komunikasinya.
3. Sistem pemrosesan data untuk pengolahan data semua robot yang dikumpulkan untuk menghasilkan perintah bagi masing-masing robot.

A. Struktur Sistem Simulasi

Sistem simulasi terdiri dari simulator kondisi visual robot beserta lingkungan yang akan dilaksanakan oleh Gazebo Simulator dan wadah jalannya proses pengendalian robot yang akan dilaksanakan oleh Robot Operating System. Sistem simulasi dibuat dengan melanjutkan serta memodifikasi yang pernah dibuat oleh [13]. Secara umum, diagram blok sistem pada penelitian ini digambarkan pada Gambar 3.

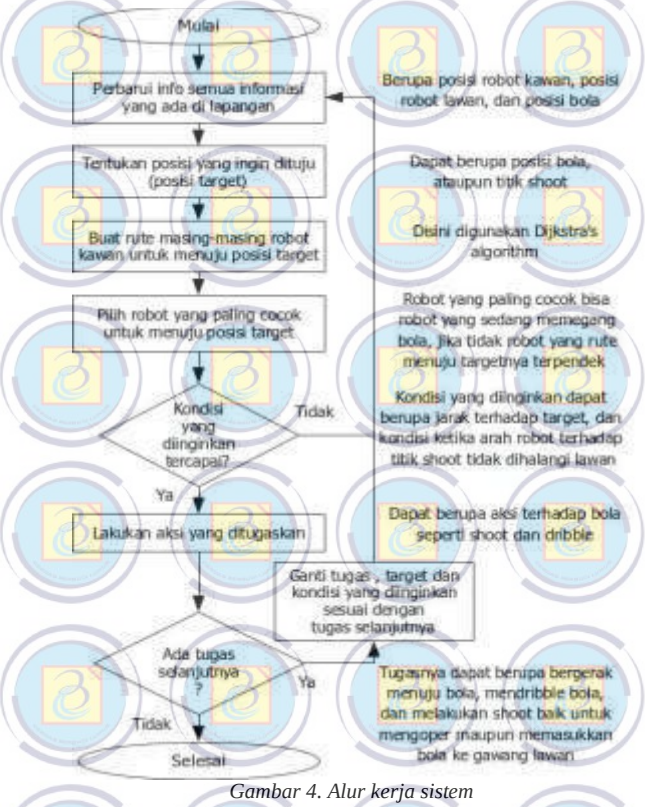


Gambar 3. Diagram blok sistem robot secara umum

Dari Gambar 3, dapat dilihat bahwa pengendali utama dapat mengendalikan masukan dan keluaran robot dengan saling berkomunikasi dengan masing-masing pengendali pada robot. Masing-masing pengendali robot terdiri dari antarmuka komunikasi, serta pengendali masukan dan keluaran. Masukan yang dibaca oleh robot adalah pembacaan posisi serta orientasi robot, posisi halangan (robot lawan) dan posisi bola. Sedangkan keluaran yang dikendalikan robot adalah kendali posisi robot (pergerakan robot), dan kendali bola (dribble dan shoot bola).

B. Algoritma Yang Ditanamkan Dalam Sistem

Algoritma sistem dijalankan oleh Robot Operating System dan dibuat dalam bahasa pemrograman C++. Alur kerja yang ditanamkan pada sistem ini ditampilkan pada Gambar 4.



Gambar 4. Alur kerja sistem

1) Bergerak Menuju Koordinat Target

Algoritma pergerakan robot menuju target dibuat dengan membuat sebuah fungsi :

bool move2target (nomorRobot , koordinat_target)

Sehingga, robot koordinator dapat lebih mudah dalam memerintahkan robot untuk bergerak. Fungsi ini memerlukan nomor robot yang ingin digerakkan dan koordinat target. Kedua parameter ini diperlukan untuk menentukan arah dan kecepatan pergerakan. Arah pergerakan yang diambil langsung lurus menuju target. Sedangkan kecepatan yang dihasilkan berbanding lurus dengan besarnya jarak antara posisi robot yang ingin digerakkan dengan targetnya.

2) Mengoper Bola

Algoritma pengoperan bola dibuat dengan membuat sebuah fungsi :

bool pengoperan (robot_pengoper,robot_penerima)

Sehingga, robot koordinator dapat lebih mudah dalam memerintahkan robot untuk mengoper bola. Selain itu fungsi ini juga menghasilkan nilai boolean sebagai penanda selesainya pengoperan untuk membantu proses pengendalian.

3) Bergerak Menuju Target Tanpa Bertabrakan Dengan Kawan

Algorima pergerakan menuju target dilakukan dengan memodifikasi fungsi pergerakan robot menuju target dengan menyisipkan fungsi lain yaitu:

int kedeketan_sama_kawan (nomorRobot , koordinat_target , batasJarak)

Fungsi ini menghasilkan nilai desimal yang merupakan nomor robot mana yang terdekat oleh robot_yang_bergerak. Namun jika tidak ada robot yang terlalu dekat dengannya maka fungsi ini akan menghasilkan nilai 0 sebagai penanda karena nilai ini tidak digunakan sebagai nomor robot. Jika fungsi ini menghasilkan nilai selain 0 yang berarti ada robot yang terlalu dekat dengan robot ini, maka dilakukan pengalihan target yang dibuat dengan fungsi:

DPoint alihkan (target,nomor_robot_yang_gerak, nomor_robot_yang_ngalangin)

Fungsi ini menghasilkan koordinat alihan dengan acuan koordinat target, posisi robot yang bergerak, dan posisi robot yang menghalangi. Pengalihan koordinat dilakukan dengan menggunakan aturan sin dan aturan cos agar robot yang ingin bergerak menuju posisi targetnya dialihkan sementara posisi targetnya agar robot ini bergerak mengitari robot yang sedang menghalangi rutenya.

4) Dijkstra's Algorithm

Dijkstra's Algorithm memerlukan titik - titik koordinat yang dapat dilalui sebagai pilihan jalur yang dapat dilalui oleh robot, yang pada penelitian ini disingkat menjadi DJpoint. Pada sistem ini DJpoint dihasilkan secara variabel (berubah-ubah) berdasarkan posisi lawan. hal ini dilakukan karena pada kasus pertandingan sepakbola beroda yang menghalangi target adalah robot musuh.

Dalam penelitian ini, setiap robot musuh akan menghasilkan 4 DJpoint yang ditempatkan di sekitar robot musuh dengan berdasarkan jarak dari tiap DJpoint ke robot musuh yang ditetapkan dan sudut antar titik yang dibagi rata. Pembuatan DJpoint dilakukan dengan menggunakan fungsi:

void buatDJpoint (DPoint *DJP)

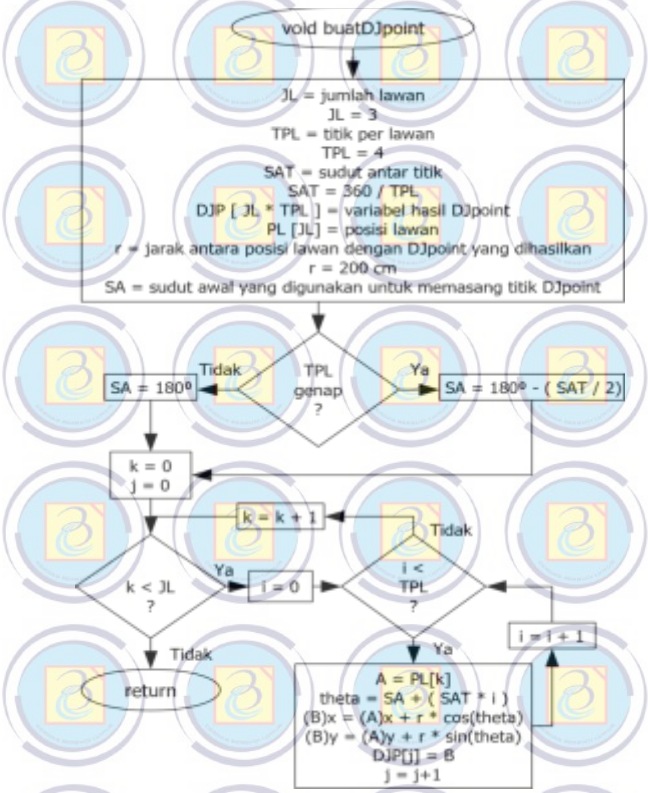
Fungsi ini menggunakan posisi lawan untuk menghasilkan DJpoint yang akan disimpan pada variabel array DJP. Cara perhitungan DJpoint dilakukan dengan menggunakan aturan sin dan aturan cos.

theta = ST + (SAT * i)
A = PL[k]
B(x) = A(x) + r * cos (theta)
B(y) = A(y) + r * sin (theta)
DJP[i] = B

dengan:

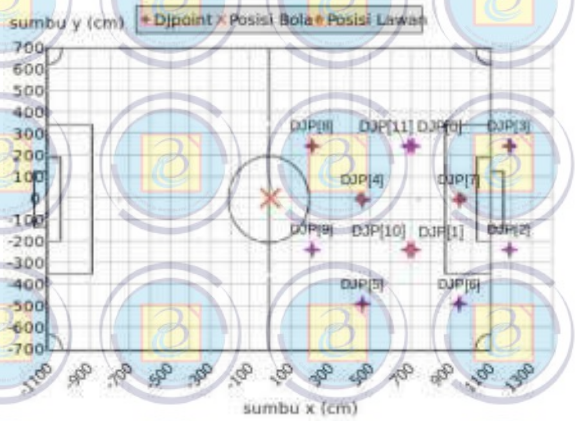
DJP[i] = koordinat DJpoint ke-i
i = 0,1,2,...(Jumlah lawan * Titik per lawan)
PL[k] = posisi lawan ke-k
k = 0,1,2,...(Jumlah lawan)
r = jarak antara posisi lawan dengan DJpoint yang dihasilkan
ST = sudut yang terakhir digunakan untuk membuat DJpoint
SAT = 360 / Titik per lawan

Flowchart fungsi ini ditunjukkan pada Gambar 5.



Gambar 5. Flowchart buat DJpoint

Seperti yang ditunjukkan pada Gambar 5, fungsi buat DJpoint menghitung DJpoint berdasarkan jumlah titik yang disebar pada tiap lawan dan juga jumlah lawannya. Hasil DJpoint diisi pada variabel array yang diberikan kepada fungsi ini. Contoh hasil pengujian fungsi ini ditunjukkan pada Gambar 6.



Gambar 6. Contoh kasus pembuatan DJpoint

Dari Gambar 6, dapat dilihat bahwa masing-masing posisi lawan akan menghasilkan 4 DJpoint yang diletakkan disekitar masing-masing posisi lawan. Namun DJpoint yang dihasilkan sebaiknya disaring terlebih dahulu seperti DJpoint yang berada di luar lapangan maupun berada terlalu dekat dengan robot lawan lainnya. Untuk menyaring DJpint yang dapat dilalui, dapat dilakukan dengan fungsi:

```
void        filterDJpoint( DPoint *DJP , bool* DJF )
```

Fungsi ini akan memeriksa apakah DJpoint tersebut dapat dilalui atau tidak. Status DJpoint dapat dilalui atau tidak akan disimpan pada variabel boolean array DJF. Variabel array boolean flag DJpoint akan diberi nilai FALSE jika DJpoint berada terlalu dekat dengan robot lawan atau berada di luar lapangan. Setelah mendapatkan flag DJpoint yang aman untuk dilalui, maka perlu dihitung jumlah DJpoint yang aman dengan menggunakan fungsi:

```
int getJumlahTitik (bool* DJF , int jumlahDJF)
```

Fungsi ini menghasilkan nilai jumlah dari isi array DJF yang bernilai TRUE. Nilai ini diperlukan untuk melakukan deklarasi variabel DJpoint aman dan juga dapat digunakan untuk melakukan perulangan dalam pengolahan data. Setelah mendapatkan jumlah DJpoint yang aman untuk dilalui, selanjutnya dapat dilakukan penyusunan DJpoint aman dengan menggunakan fungsi:

```
void      isiDJpointAman ( bool* DJF , int
jumlahDJP ,  DPoint *DJP , DPoint *DJA)
```

Fungsi ini menyusun DJpoint dan DJpoint flag menjadi DJpointAman. Setelah mendapatkan DJpointAman, dapat dilakukan pembuatan rute serta jarak tempuh bagi tiap DJpointAman. Proses ini dilakukan dengan cara

1. Periksa setiap DJpoint yang dapat langsung menuju target, lalu catat jarak tempuhnya.
2. Periksa setiap DJpoint sisanya yang dapat menuju DJpoint yang baru saja terjalur pada langkah sebelumnya. lalu catat jaraknya ditambah dengan jarak tempuh milik DJpoint tersebut. Pilih kombinasi yang menghasilkan jarak tempuh terkecil.
3. Lakukan lagi seperti pada langkah No. 2, sampai tidak ada DJpoint yang tersisa.

Setelah DJpoint memiliki jalur serta jarak tempuh menuju target, maka DJpoint dapat digunakan untuk menentukan rute bagi robot untuk mencapai target.

5) Menentukan Rute Masing-masing DJpoint Menuju Posisi Target

Menentukan rute masing-masing DJpoint menuju target dapat dilakukan dengan cara:

1. Periksa apakah DJpoint dapat langsung menuju target dan rutenya tidak terlalu dekat dengan robot lawan. Jika benar, maka posisi target dijadikan posisi target pada DJpoint ini dan atur jarak tempuh menuju targetnya menjadi jarak antara DJpoint ke target. Jika salah maka lakukan langkah ke-2.
2. Periksa apakah DJpoint dapat langsung menuju DJpoint lain yang berhasil memiliki rute menuju target dan rutenya tidak terlalu dekat dengan robot lawan. ika benar, maka posisi DJpoint yang lain ini dijadikan posisi target pada DJpoint ini dan atur jarak tempuh menuju targetnya menjadi jarak antara DJpoint ke DJpoint yang lain ini. Jika ada lebih dari satu DJpoint lain yang dapat dicapai, maka pilih DJpoint yang menghasilkan rute menuju target nya yang terpendek. Jika tidak ada DJpoint lain yang

dapat dicapai, maka DJpoint ini tidak memiliki rute dan atur jarak tempuh menuju targetnya menjadi 999999 cm.

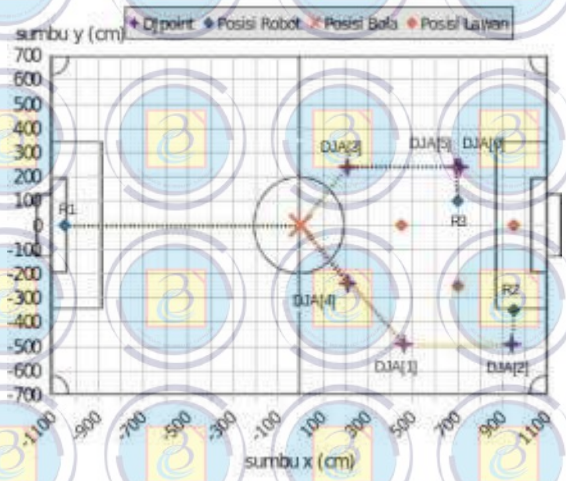
Setiap DJpoint yang dapat langsung menuju target dan rutenya tidak terlalu dekat dengan robot lawan akan menggunakan posisi target sebagai posisi targetnya. Sedangkan DJpoint yang tidak dapat langsung menuju target karena rutenya terlalu dekat dengan robot lawan akan menargetkan DJpoint lain yang memiliki jarak tempuh menuju targetnya terdekat dan dapat dicapai oleh DJpoint tersebut.

6) Menentukan Rute Robot Menuju Target Berdasarkan DJpoint Yang Dihasilkan

Menentukan rute masing-masing robot menuju target berdasarkan DJpoint yang dihasilkan dapat dilakukan dengan cara:

1. Periksa apakah robot dapat langsung menuju target dan rutenya tidak terlalu dekat dengan robot lawan. Jika benar, maka posisi target dijadikan posisi target pada robot ini dan atur jarak tempuh menuju targetnya menjadi jarak antara robot ini ke target. Jika salah maka lakukan langkah ke-2.
2. Periksa apakah robot dapat langsung menuju DJpoint memiliki rute menuju target dan rutenya tidak terlalu dekat dengan robot lawan. ika benar, maka posisi DJpoint ini dijadikan posisi target pada robot ini dan atur jarak tempuh menuju targetnya menjadi jarak antara robot ke DJpoint ini. Jika ada lebih dari satu DJpoint yang dapat dicapai, maka pilih DJpoint yang menghasilkan rute menuju target nya yang terpendek. Jika tidak ada DJpoint lain yang dapat dicapai, maka robot ini tidak memiliki rute dan atur jarak tempuh menuju targetnya menjadi 999999 cm.

Contoh kasus menentukan rute masing-masing robot menuju target ditunjukkan pada Gambar 7.



Gambar 7. Contoh kasus menentukan rute masing-masing robot menuju target

7) Menentukan Robot Mana Yang Harus Bergerak Dengan Acuan Rute Yang Dihasilkan Oleh Dijkstra's Algorithm Dari Setiap Robot

Penentuan robot mana yang harus bergerak adalah robot yang rutenya terdekat dalam mencapai targetnya. Jika mengharuskan untuk membawa bola ke target, maka robot yang rutenya terdekat dengan bola akan diutamakan.

8) Menentukan Posisi Yang Cocok Bagi Robot Yang Sedang Memegang Bola Untuk Menembak Dengan Dijkstra's Algorithm

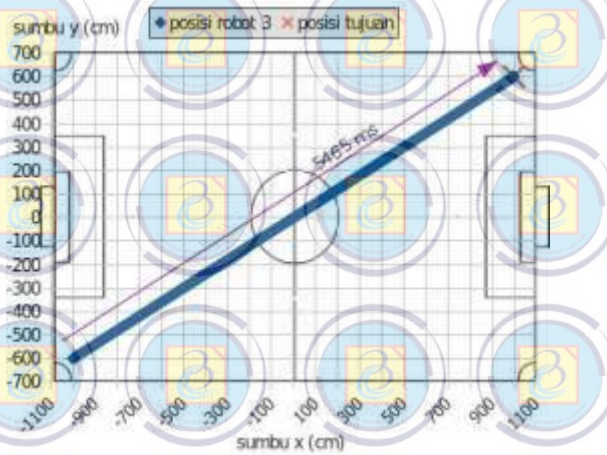
Posisi yang cocok bagi robot yang sedang memegang bola dilakukan dengan cara:

1. Tentukan posisi gawang lawan yang celahnya paling lebar, lalu jadikan tengah tengah dari celah tersebut sebagai posisi tujuan
2. Lakukan pergerakan menuju posisi tujuan dengan menggunakan Dijkstra's Algorithm
3. Jika arah robot menuju target tidak dihalangi robot lawan dan jaraknya cukup untuk melakukan shoot, maka arahkan rotasi robot sampai mengarah tepat pada tujuan
4. Posisi koordinat dan rotasi robot saat ini adalah posisi yang cocok untuk melakukan shoot.

IV. PENGUJIAN PERILAKU ROBOT

A. *Robot Bergerak Menuju Posisi Target*

Pada pengujian ini, robot yang bergerak adalah robot 3. Robot 3 bergerak dengan posisi awal (-1000x , -600y) dan posisi target (1000x , 600y). Posisi target berada pada sumbu x dan sumbu y yang berbeda terhadap posisi robot 3. Kondisi posisi selama pergerakan robot 3 menuju targetnya ditunjukkan pada Gambar 8.



Gambar 8. Kondisi pergerakan robot 3 menuju posisi target pada pengujian A.2.

Dari Gambar 8, dapat dilihat bahwa robot 3 bergerak lurus langsung menuju posisi target. Waktu yang diperlukan untuk mencapai posisi targetnya yang berjarak 23.3238075794 meter adalah 5465 ms.

Dari data hasil pengujian, dapat disimpulkan bahwa robot berhasil mencapai targetnya dengan cara bergerak lurus langsung menuju targetnya.

B. *Mengoper Bola*

Pengujian mengoper bola dilakukan dengan robot 3 sebagai pengoper dan robot 2 sebagai penerima. Hasil pengujian dibuat dalam dalam

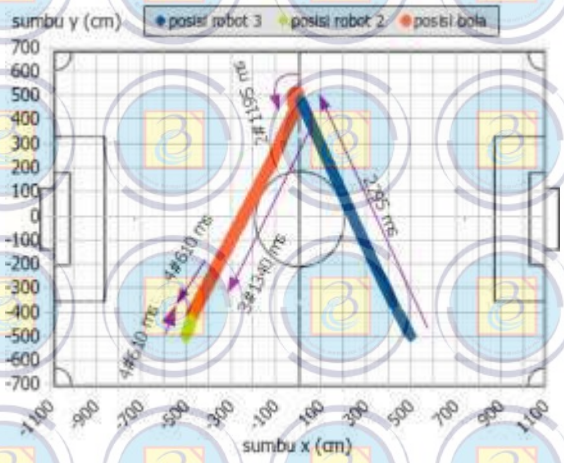
grafik-grafik posisi robot beserta dengan bola. Pada saat kondisi awal pengujian, robot 3 berada pada posisi (500x , -500y), robot 2 berada pada posisi (-500x , -500y), posisi bola berada pada posisi (0x , 500y).

Selama robot 3 bergerak unruk mengambil bola, robot 3 bergerak lurus menuju bola. Waktu yang diperlukan robot 3 untuk mencapai bola yang berjarak 11.1803398875 meter adalah 2295 ms.

Setelah robot 3 mengambil bola, robot 3 mulai memposisikan arahnya sendiri yang sedang memegang bola. Robot 3 berputar menghadap ke arah robot 2 sambil membawa bola dengan memerlukan waktu sebesar 1195 ms.

Setelah robot 3 menghadap ke robot 2, robot 3 mulai melakukan shoot. Shoot yang dilakukan robot 3 hampir tepat mengarah kepada robot 2. Bola mencapai posisi ini saat 1340 ms setelah dishoot oleh robot 3.

Setelah bola berada pada jarak yang dekat dengan robot 2, robot 2 mulai bergerak mengambil bola. Ketika bola mulai berada di dekat robot 2, robot 2 juga bergerak ke posisi bola walaupun bola masih dalam kondisi begerak. Dari posisi bola berada dekat dengan bola sampai bola diambil oleh robot 2 waktu berjalan selama 610 Setelah robot 2 mengambil bola, pengoperan telah selesai.



Gambar 9. Data pada pengujian B

Dari hasil pengujian ini, dapat disimpulkan bahwa pengoperan berhasil dilakukan. Meskipun robot tidak sedang memegang bola, yang penting dipilih robot mana yang harus mengoper dan robot mana yang harus menerima. Maka robot pengoper akan mengambil bola dan mengopernya kepada robot penerima, lalu robot penerima akan menerima bola operan dari robot pengoper. Pengoperan dalam pengujian ini memerlukan waktu selama 5455 ms.

C. Bergerak Menuju Target Tanpa Bertabrakan Dengan Kawan

Pengujian pergerakan robot menuju target tanpa bertabrakan dengan kawan dilakukan sebanyak 2 kali dengan posisi awal robot serta posisi tujuan yang berbeda ditambah dengan posisi robot lain yang diam namun menghalangi robot yang ingin bergerak ke posisi tujuannya. Dalam pengujian ini, robot yang bergerak adalah robot 3, dan robot yang diam adalah robot 2.

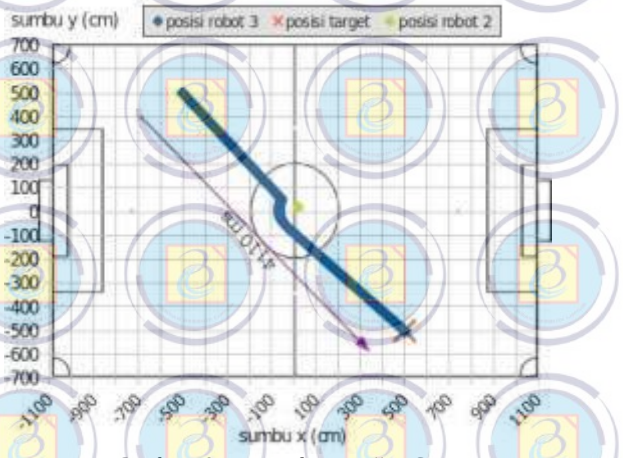
1) Skenario 1

Robot 3 bergerak dengan posisi awal (-500x , 500y) dan posisi target (500x , -500y), sedangkan robot 2 diam pada posisi (20x , 20y). Untuk mencapai posisi target, robot 3 akan dihalangi oleh robot 2 nantinya.

Robot 3 bergerak menuju target sampai robot 3 terlalu dekat dengan robot 2 dan posisi robot 2 agak condong ke kirinya robot 3 daripada posisi target. Pergerakan ini dilakukan selama 1265 ms.

Ketika robot 3 menghindari robot 2 yang terlalu dekat dengannya, robot 3 mengitari robot 2 ke arah kanannya robot 3 terhadap posisi target. Hal ini disebabkan karena posisi robot 2 agak condong ke kirinya robot 3 daripada posisi target. Robot 3 mengitari robot 2 selama 235 ms.

Ketika robot 3 bergerak setelah menghindari robot 2 menuju target, robot 3 tidak dihalangi robot 2 untuk mencapai posisi target, maka robot 3 bergerak lurus langsung menuju posisi target dengan menghabiskan waktu sebesar 2610 ms. Data posisi jika dilihat dari awal sampai akhir ditampilkan pada Gambar 10.



Gambar 10. Data pada pengujian C.1

Dari Gambar 10, dapat dilihat bahwa robot 3 bergerak lurus menuju posisi target, namun melakukan belokan ketika dihalangi oleh robot 2. Dalam skenario ini, robot 3 memerlukan waktu sebesar 4110 ms untuk mencapai posisi targetnya.

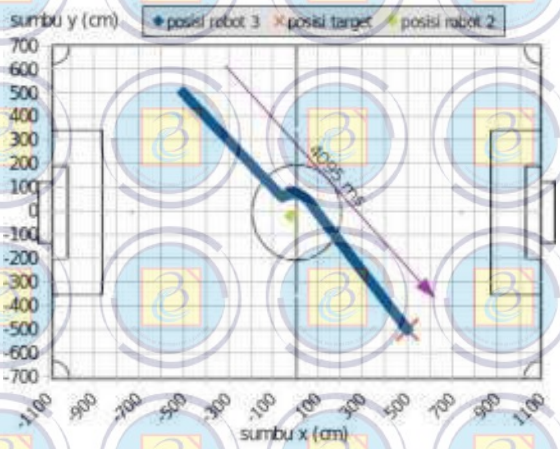
2) Skenario 2

Robot 3 bergerak dengan posisi awal (-500x , 500y) dan posisi target (500x , -500y), sedangkan robot 2 diam pada posisi (-20x , -20y). Untuk mencapai posisi target, robot 3 akan dihalangi oleh robot 2 nantinya. Posisi awal pengujian 4.3.2 ini memeang mirip dengan posisi awal pengujian 4.3.1, namun bedanya adalah posisi robot 2 yang dibuat supaya condong ke kanan dari robot 3 terhadap posisi target.

Robot 3 bergerak menuju target sampai robot 3 terlalu dekat dengan robot 2 dan posisi robot 2 agak condong ke kanannya robot 3 daripada posisi target. Pergerakan ini dilakukan selama 1260 ms.

Ketika robot 3 menghindari robot 2 yang terlalu dekat dengannya robot 3 mengitari robot 2 ke arah kirinya robot 3 terhadap posisi target. Hal ini disebabkan karena posisi robot 2 agak condong ke kanannya robot 3 daripada posisi target. Robot 3 mengitari robot 2 selama 240 ms.

Ketika robot 3 bergerak setelah menghindari robot 2 menuju target, robot 3 tidak dihalangi robot 2 untuk mencapai posisi target, maka robot 3 bergerak lurus langsung menuju posisi target dengan menghabiskan waktu sebesar 2595 ms. Data posisi jika dilihat dari awal sampai akhir ditampilkan pada Gambar 11.



Gambar 11. Data pada pengujian C.2.

Dari Gambar 11, dapat dilihat bahwa robot 3 bergerak lurus menuju posisi target, namun melakukan belokan ketika dihalangi oleh robot 2. Dalam skenario ini, robot 3 memerlukan waktu sebesar 4095 ms untuk mencapai posisi targetnya.

D. Menentukan Rute Menuju Bola Dengan Dijkstra's Algorithm Dan Menjalankan Robot Yang Memiliki Rute Terdekat Untuk Mengambil Bola Lalu Menentukan Posisi Yang Cocok Untuk Menembak Dengan Dijkstra's Algorithm

Pengujian ini dilakukan sebanyak 2 kali dengan posisi awal robot serta posisi bola yang berbeda namun dengan posisi robot lawan yang sama. Tugas yang dilakukan sama yaitu menjalankan robot yang memiliki rute terdekat untuk mengambil bola, lalu robot tersebut menggiring bola unruk melakukan shoot dengan posisi robot lawan menghalangi shoot jika robot tidak menggiring bola ke tempat yang tidak terhalang oleh robot lawan.

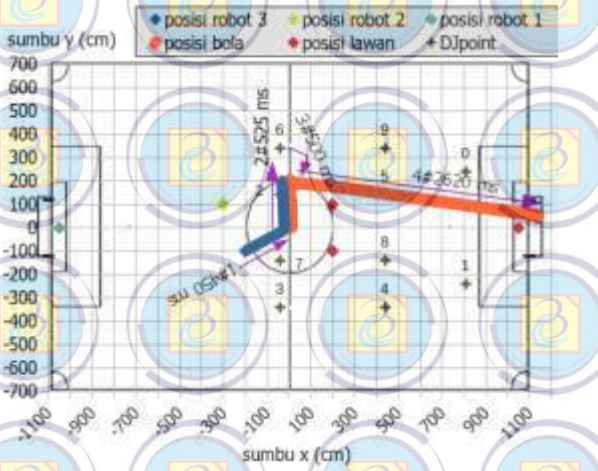
1) Skenario 1

Skenario kali ini menjauhkan posisi robot 2 dari bola. Hal ini dimaksudkan agar jelas bahwa robot 3 merupakan robot yang terdekat dengan bola. Selain itu, pada skenario ini bola diposisikan agar tidak terhalang oleh musuh ketika robot ingin mengambilnya.

Robot 3 bergerak untuk mengambil bola. Hal ini dianggap sesuai karena cukup jelas terlihat pada Gambar 12 bahwa robot 3 merupakan robot yang terdekat dengan bola. Robot 3 dapat langsung menuju bola tanpa harus melalui DJpoint. Waktu yang diperlukan bagi robot 3 untuk mengambil bola adalah 450 ms dengan jarak bola sejauh 2.2360679775 meter.

Selanjutnya ketika robot yang mengambil bola menbawa bola untuk melakukan shoot robot 3 membawa bola menuju DJpoint 6 namun berhenti ketika mencapai posisi yang tidak dihalangi musuh baginya untuk dapat melakukan shoot. Pergerakan ini berlangsung selama 525 ms.

Selanjutnya ketika robot berada di posisi shoot, bola yang dibawa oleh robot 3 diarahkan ke titik shoot. Rotasi ini dilakukan selama 500 ms. Selanjutnya ketika robot yang membawa bola melakukan shoot. Shoot dilakukan oleh robot 3 mengarah ke gawang, lalu bola berhasil masuk ke gawang lawan. Bola bergerak selma 2620 ms setelah shoot yang dilakukan oleh robot 3.



Gambar 12. Data pada pengujian D.1

Total waktu yang diperlukan untuk skenario ini adalah sebesar 4095 ms. Berdasarkan data yang didapatkan dari pengujian D.1, dapat disimpulkan pengujian dengan skenario ini berhasil melaksanakan tugas pada pengujian ini.

2) Skenario 2

Skenario kali ini menjauhkan posisi robot 3 dari bola. Hal ini dimaksudkan agar jelas bahwa robot 2 merupakan robot yang terdekat dengan bola. Selain itu, pada skenario ini bola diposisikan agar terhalang oleh musuh ketika robot ingin mengambilnya.

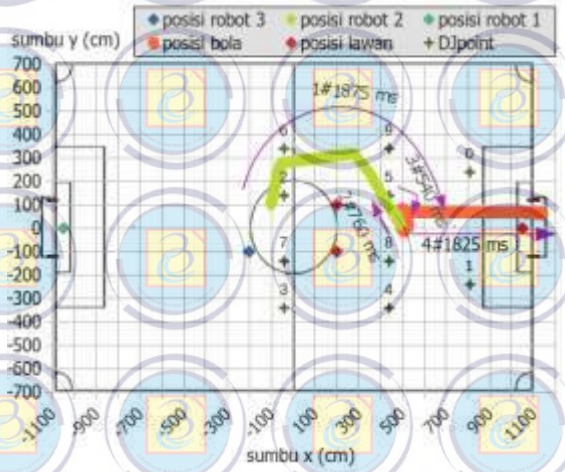
Robot 2 bergerak untuk mengambil bola. Hal ini dianggap sesuai karena cukup jelas terlihat pada Gambar 13 bahwa robot 2 merupakan robot yang terdekat dengan bola. Untuk mengambil bola, robot 2 bergerak mengarah ke DJpoint 6, lalu mengarah ke DJpoint 9, lalu mengarah ke DJpoint 5, lalu barulah dapat langsung menuju bola. Waktu yang diperlukan bagi robot 2 untuk mengambil bola adalah 1875 ms dengan jarak bola sejauh 6.0827625303 meter dan

adanya robot lawan yang menghalangi rutenya menuju bola.

Selanjutnya ketika robot yang mengambil bola menbawa bola untuk melakukan shoot, robot 2 membawa bola menuju DJpoint 5 namun berhenti ketika mencapai posisi yang tidak dihalangi musuh baginya untuk dapat melakukan shoot. Pergerakan ini berlangsung selama 760 ms.

Selanjutnya kondisi ketika robot yang berada di posisi shoot untuk merotasi dirinya bersama dengan bola ke arah titik shoot, bola yang dibawa oleh robot 2 diarahkan ke titik shoot. Rotasi ini dilakukan selama 540 ms.

Selanjutnya ketika robot yang membawa bola melakukan shoot. Shoot dilakukan oleh robot 2 mengarah ke gawang, lalu bola berhasil masuk ke gawang lawan. Bola bergerak selma 1825 ms setelah shoot yang dilakukan oleh robot 2.



Gambar 13. Data pada pengujian D.2.

Total waktu yang diperlukan untuk skenario ini adalah sebesar 5000 ms. Berdasarkan data yang didapatkan dari pengujian D.2, dapat disimpulkan pengujian dengan skenario ini berhasil melaksanakan tugas pada pengujian ini.

V. KESIMPULAN

Berdasarkan dari hasil pengujian yang telah dilakukan dalam penelitian ini, berikut ini merupakan poin-poin yang dapat disimpulkan.

1. Pengoperan bola, berhasil dilakukan dengan mementukan terlabih dahulu robot mana yang dijadikan pengoper dan robot mana yang dijadikan penerima. Hasil dari pengujian yang dilakukan, pengoperan memerlukan waktu selama 5455 ms dengan jarak bola dari robot pengoper sebesar 11.08 meter dan jarak bola dari robot penerima juga sebesar 11.08 meter.
2. Agar robot kawan tidak saling bertabrakan ketika robot bergerak menuju posisi targetnya karena robot yang sedang bergerak tetap memperhatikan posisi robot kawannya. Lalu ketika robot yang bergerak ke posisi target sedang berada terlalu dekat dengan robot kawan, posisi target dialihkan secara sementara menggunakan aturan sin dan cos ketika jalurnya menuju posisi target terhalang oleh robot kawan lainnya. Hasil yang dari pengujian yang dilakukan, robot memerlukan waktu sekitar 4100 ms untuk bergerak menuju posisi target yang berjarak 14.14 meter dengan adanya 1 robot kawan yang menghalangi di pertengahan jalurnya.
3. Robot yang jarak tempuhnya terdekat dengan posisi bola berhasil diperintahkan untuk mengambil bola karena dibuatnya rute masing-masing robot untuk menuju bola menggunakan Dijkstra's Algorithm. Hasil dari pengujian yang dilakukan pada skenario 1, robot berhasil mengambil bola dalam waktu 450 ms dengan jarak bola sejauh 2,236 meter. Sedangkan hasil yang dilakukan pada pengujian skenario 2 robot berhasil mengambil bola dalam waktu 1875 ms dengan jarak bola sejauh 6.083 meter dan adanya robot lawan yang menghalangi rutenya menuju bola.
4. Robot yang memegang bola berhasil berhasil menggiring bola dan melakukan shoot untuk memasukkan bola ke gawang lawan dengan membawa bola menggunakan Dijkstra's Algorithm sampai posisi yang jika diambil rutenya ke titik shoot tidak dihalangi oleh robot lawan, lalu melakukan shoot. Hasil dari pengujian yang dilakukan pada skenario 1, robot berhasil memasukkan bola ke gawang lawan dalam waktu 4095 ms. Sedangkan hasil dari pengujian yang dilakukan pada skenario 2, robot berhasil memasukkan bola ke gawang lawan dalam waktu 5000 ms.

REFERENSI

[1] A. A. Rachman, "Sistem Perencanaan Rute Gerak Pada Robot Sepakbola Beroda," Institut Teknologi Sepuluh Nopember, 2017.

[2] F. A. Rahman *et al.*, "Motion Planning in Dynamic Environment for Middle Size League using Theta* and Polynomial Trajectory Generator," in *5th Indonesian Symposium on Robotic Systems and Control*, 2017, pp. 113–117.

[3] I. P. Suryaningsih, A. Risal, H. Jaya, and A. R. Soccer, "Algoritma Multi Planning Pada Robot Soccer," *5th Indones. Symp. Robot. Syst. Control*, pp. 156–159, 2017.

[4] R. H. Abiyev, I. Günsel, N. Akkaya, E. Aytac, A. Çağman, and S. Abizada, "Robot Soccer Control Using Behaviour Trees and Fuzzy Logic," 2016, doi: 10.1016/j.procs.2016.09.430.

[5] Fitri, K. R. R, A. Rahmansyah, and W. Darwin, "Penggunaan Bahasa Pemrograman Python Pusat Kendali Pada Robot 10-D," in *5th Indonesian Symposium on Robotic Systems and Control*, 2017, pp. 23–26.

[6] R. A. Pamungkas, A. Maulana, D. P. Sya'ban, R. Ramdani, A. Musafa, and I. Riyanto, "WiFi Data

Communication System Design for Wheeled Soccer Robot Controller," *Adv. Sci. Lett.*, vol. 24, no. 11, pp. 8782–8786, 2018, doi: 10.1166/asl.2018.12345.

[7] M. Occhiogrosso, *Graphs of Trigonometric Functions: Trigonometry*. Milliken Publishing Company, 2007, 2007.

[8] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959, doi: 10.1007/BF01386390.

[9] S. A. Fadzli, S. I. Abdulkadir, M. Makhtar, and A. A. Jamal, "Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries," *2015 IEEE 2nd Int. Conf. InformationScience Secur. ICISS 2015*, 2016, doi: 10.1109/ICISSEC.2015.7371031.

[10] S. Pietrzik and B. Chandrasekaran, "Setting up and Using ROS-Kinetic and Gazebo for Educational Robotic Projects and Learning," in *Journal of Physics: Conference Series*, 2019, vol. 1207, no. 1, doi: 10.1088/1742-6596/1207/1/012019.

[11] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 3, pp. 2149–2154, 2004, doi: 10.1109/iros.2004.1389727.

[12] lentin joseph, "Robotic Operating System," *Introduction to ROS*. p. 385, 2017, [Online]. Available: https://www.ros.org.

[13] W. Yao, W. Dai, J. Xiao, H. Lu, and Z. Zheng, "A simulation system based on ROS and Gazebo for RoboCup Middle Size League," in *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, 2015, pp. 54–59, doi: 10.1109/ROBIO.2015.7414623.