ANALISIS SIMULASI PENERAPAN ALGORITMA D* PADA SLAM UNTUK EKSPLORASI BERBASIS MULTI-ROBOT

*SIMULATION ANALYSIS OF D * ALGORITHM APPLICATION IN SLAM FOR MULTI-ROBOT-BASED EXPLORATION*

Naufal Rif'at Najib[1], Agung Nugroho Jati S.T.,M.T.[2] Muhammad Faris Ruriawan S.T.,M.T.[3]
[1,2,3]Prodi S1 Teknik Komputer, Fakultas Teknik Elektro, Universitas Telkom
[1]rifatnajib@student.telkomuniversity.ac.id. [2]agungnj@gmail.com,
[3]muhammadfaris@telkomuniversity.ac.id

Abstrak

Sistem navigasi yang biasa dilakukan terkait dengan proses pemetaan atau pembuatan peta sebagai titik acuan. Saat ini sistem navigasi secara konseptual dikenal sebagai SLAM (Simultan Lokalisasi dan Pemetaan), yaitu proses pemetaan sekaligus pelokalan. Selanjutnya SLAM berkembang kembali dan memiliki sistem pencarian jalur eksplorasi untuk mendukung proses pembuatan peta (pemetaan) maupun lokalisasi. Sistem pencarian jalur eksplorasi dibuat dengan menggunakan algoritma tertentu dan terdapat banyak algoritma pencarian jalur eksplorasi yang berhasil dikembangkan.

Proses pencarian jalur yang dimiliki algoritma dikhususkan untuk pencarian jalur eksplorasi berdasarkan informasi yang sudah didapat seperti sketsa kasar suatu daerah atau menggunakan LiDAR pada Mobile Robot. Salah satu algoritma pencarian jalur adalah algoritma D *. Proses pencarian jalur dalam algoritma D * masih sangat beragam dalam hasil dan diperlukan aspek tambahan untuk meningkatkan keakuratannya dalam banyak penelitian.

Hasil dari penelitian ini adalah membuktikan bahwa algoritma D * mampu mendukung proses pembuatan jalur pada sistem Mobile Robot. Hal ini juga didukung dengan eksplorasi multi robot, setiap robot mampu saling mendukung dengan melanjutkan kegiatan eksplorasi jika salah satu robot berhenti bekerja, hal ini didukung dengan hasil beberapa pengujian simulasi multi robot mampu mengunjungi minimal satu robot. titik tujuan. Simulasi multi robot juga dapat mengunjungi lebih banyak titik tujuan dengan presentasi dua kali lebih banyak daripada simulasi robot tunggal. Dari semua pengujian yang dilakukan, 87% di antaranya membuktikan bahwa area eksplorasi yang berhasil dicapai dalam simulasi multi robot ternyata lebih luas. Sayangnya, simulasi multi robot ini terbebani oleh durasi waktu simulasi yang lebih lama dibandingkan simulasi robot tunggal, yaitu sekitar 69,3%. Algoritma ini mampu mendukung proses pencarian jalur menuju titik tujuan dalam kegiatan eksplorasi.

Kata kunci : Algoritma D*, SLAM, Mobile Robot, Navigasi, informasi, pencarian

Abstract

The usual navigation system is associated with the process of mapping or making maps as a reference point. Nowadays the navigation system is conceptually known as SLAM (Simultaneous Localization and Mapping), which is the process of mapping simultaneously localizing. Furthermore, SLAM developed again and had an exploration path search system to support the process of making maps (mapping) as well as localization. Exploration path search systems are created using certain algorithms and there are many exploration path search algorithms successfully developed.

*The process of searching for paths owned by the algorithm is specifically for searching for exploration pathways based on information already obtained such as rough sketches of an area or using LiDAR on Mobile Robot. One of the path search algorithms is D * algorithm. The path search process in the D * algorithm is still very diverse in results and additional aspects are needed to improve its accuracy in many studies.*

*The result of this research is to prove that the D * algorithm is able to support the path creation process in the Mobile Robot system. This is also supported by multi-robot exploration, each robot is able to support each other by continuing exploration activities if one of the robots*

stops working, this is supported by the results of several tests on multi-robot simulations capable of visiting at least one destination point. Multi robot simulation can also visit more points of destination with twice as much presentation than single robot simulation. From all the tests carried out, 87% of them proved that the area of exploration that was successfully reached in the multi robot simulation proved to be wider. Unfortunately, the multi-robot simulation is burdened by the longer simulation time duration than the single robot simulation, which is around 69.3%. This algorithm is able to support the process of finding a path to the destination point in exploration activities.

Keywords: ***: D* Algorithm, SLAM, Mobile Robot, Navigation, information, search*.**

1. Pendahuluan

SLAM (*Simultaneous Localization And Mapping)* adalah salah satu konsep pembuatan peta otomatis dan secara simultan membuat sebuah pemetaan lokasi pembuat peta berdasarkan titik referensi (lokalisasi).

Perkembangan konsep navigasi ini didukung dengan munculnya banyak algoritma yang bertugas untuk menjadi mesin pencarian jalur pada SLAM agar proses eksplorasi menggunakan SLAM bisa lebih tertata. Pembahasan pada jurnal ini merupakan tentang penerapan salah satu algoritma pencarian jalur yang digunakan untuk mendukung aktivitas eksplorasi menggunakan SLAM yaitu Algoritma D*.

Algoritma D* merupakan hasil modifikasi dari Algoritma A* karena itu Algoritma D* memiliki beberapa perbedaan. Penulis pada jurnal ini mencoba untuk menguji karakteristik Algoritma D* tersebut pada sebuah simulasi. Simulasi ini akan diimplementasikan pada beberapa robot yang mengeksplorasi suatu area.

Dalam simulasi yang dilakukan, penulis mencoba untuk mengukur karakteristik Algoritma D* dengan beberapa aspek. Aspek tersebut adalah waktu eksplorasi, waktu simulasi, dan luas area eksplorasi yang berhasil dilakukan pada simulasi *multi* robot.

2. Dasar Teori

2.1 Multi-Robot

Multi-Robot adalah kumpulan dari beberapa robot yang bekerja sama sebagai satu tim atau komunitas [1]. Multi-Robot biasanya digunakan untuk mengeksplorasi area yang akan dipetakan secara detail pada area tertentu [2]. Multi-Robot merupakan pengembangan dari teknologi robot tunggal. Proses eksplorasi akan berjalan lebih efektif karena informasi yang diperoleh dapat berjalan secara paralel dengan jumlah yang lebih banyak karena informasi yang diperoleh dari robot yang bertugas berjalan secara bersamaan. Tingkat kesulitan dalam mengontrol Multi-Robot juga akan meningkat dibandingkan robot individu [3]. Hal tersebut dibuktikan dengan banyaknya konsep algoritma yang dibuat oleh banyak developer untuk mengoptimalkan kendali Multi Robot.

Ada jurnal oleh Shua D. Han dan Jingjin Yu yang membahas konsep algoritma pencarian jalur untuk Multi-Robot yang berfokus pada penggunaan nilai database heuristik. Mereka menggunakan algoritma DDM (Diversified path Database driven Multi-robot Path Planning) yang menggunakan berbagai nilai database heuristik. Hasil dari penelitian mereka adalah bahwa algoritma DDM yang mereka gunakan terbukti secara empiris mencapai kecepatan komputasi yang jauh lebih cepat sekaligus menghasilkan solusi berkualitas tinggi, baik untuk manajemen masalah tunggal maupun manajemen masalah dinamis [18].

2.2 SLAM (Simultaneous Localization And Mapping)

SLAM memiliki konsep mapmaker yang bertugas memetakan suatu area atau area, yang sekaligus membuat peta informasi posisi pembuat peta terhadap suatu titik acuan pada peta yang telah dibuat [4]. SLAM secara singkat berfungsi untuk membuat peta. Metode yang digunakan dalam pembuatan peta juga bermacam-macam. SLAM biasa digunakan untuk mengetahui informasi kondisi suatu daerah berdasarkan kondisi di atas permukaan tanah di daerah tersebut. Hasil pemetaan wilayah yang diperoleh melalui SLAM memiliki skala peta yang sama dengan wilayah yang dipetakan. Informasi ini sangat membantu para pembuat peta untuk membuat peta dengan informasi yang lebih aktual dan lebih banyak.

Jurnal yang dibuat oleh H. Jacky Chang dan kawan-kawan membahas tentang pengembangan konsep SLAM yang disebut P-SLAM yang bertujuan untuk meningkatkan akurasi yang diperoleh

robot yang menggunakan SLAM dalam kegiatan eksplorasi. Mereka mengembangkan P-SLAM dengan fungsi prediktif yang berguna bagi para pembuat peta dalam membuat peta wilayah yang mereka jelajahi. Hasilnya, robot pembuat peta mampu mengurangi waktu eksplorasi yang dilakukan dan mempersingkat waktu proses SLAM berlangsung. Hasil ini menyimpulkan bahwa P-SLAM sangat efektif dalam eksplorasi area yang dilakukan di dalam ruangan [4]

2.3 Gmapping

Gmapping merupakan salah satu metode yang digunakan dalam SLAM untuk memetakan suatu wilayah. Gmapping menggunakan sinar laser untuk memetakan area yang dijelajahi. Sinar laser yang dipancarkan akan memantul kembali ke arah asal sinar laser. Jarak pantulan laser dengan alat pemancar laser menjadi data yang dapat digunakan untuk membuat peta. Gmapping biasanya digunakan untuk eksperimen sederhana di SLAM, terutama dalam simulasi. Simulasi menggunakan Gmapping dapat digunakan dengan menggunakan framework aplikasi ROS (Robot Operating System) yang tersedia pada OS Ubuntu (Sistem Operasi).

.

2.4 Extended Kalman Filter

EKF (*Extended Kalman Filter)* memiliki formula di algoritmanya yang berfungsi untuk memperkirakan atau mengestimasi hasil pengukuran selanjutnya berdasarkan data-data yang sudah didapat [6]. EKF bertujuan untuk mengkalkulasi nilai yang bersifat non-linier. EKF adalah perkembangan dari Algoritma Kalman Filter yang berfungsi sama namun perbedaannya adalah Algoritma Kalman Filter berfungsi untuk mengkalkulasi nilai-nilai yang bersifat linier [7]. Algoritma EKF sangat populer dan sering digunakan dalam penyelesaian kalkulasi sistem navigasi seperti SLAM karena kesederhanaan dan faktor efektifitasnya [8].

Di dalam formula EKF (*Extended Kalman Filter)* kondisi model transisi dan observasi tidak harus berupa fungsi linier dan bisa saja berupa fungsi yang berbeda.

$$\mathcal{X}_{\mathcal{k}} = \mathcal{f}(\mathcal{X}_{\mathcal{k}-1}, \mathcal{U}_{\mathcal{k}}) + \boldsymbol{w}_{\mathcal{k}} \quad (1)$$

$$\boldsymbol{Z}_{\mathcal{k}} + \boldsymbol{h}(\boldsymbol{X}_{\mathcal{k}}) + \boldsymbol{v}_{\mathcal{k}} \quad (2)$$

Kedua persamaan di atas yaitu persamaan (1) dan persamaan (2) merupakan persamaan prediksi yang berkaitan. Persamaan (1) merupakan persamaan yang diturunkan dari basis filter Kalman dan persamaan (2) merupakan persamaan yang menjumlahkan nilai model observasi yang memetakan ruang keadaan aktual ke dalam ruang yang diamati (h (X_k)) dan Nilai noise yang diamati diasumsikan nol yaitu white noise Gaussian dengan kovariansi (v_k) [19].

$\mathbf{W_k}$ dan $\mathbf{V_k}$ adalah variabel proses dan pengamatan yang bersifat *noises* yang keduanya diasumsikan bernilai nol (0) untuk rata-rata *Multivariat Gaussian* dengan *Covariance* $\mathbf{Q_k}$ dan $\mathbf{R_k}$ adalah vektor kontrol [9].

Fungsi $\mathcal{f}$ bisa digunakan untuk menghitung data yang menunjukkan keadaan yang terprediksi dari hasil estimasi sebelumnya. Variabel $\boldsymbol{h}$ juga merupakan fungsi yang bisa digunakan untuk menghitung data-data ukuran yang sudah terprediksi dari hasil data-data kondisi terprediksi dari $\mathcal{f}$ yang baru saja disebutkan. Namun $\mathcal{f}$ dan $\boldsymbol{h}$ tidak bisa diterapkan sebagai kovarian secara langsung. Sebaliknya, matriks turunan parsial (*Jacobian*) telah terhitung [10].

Di setiap langkah waktunya, *Jacobian* akan dievaluasi dengan kondisi yang telah terhitung. Matriks ini dapat dihitung di dalam persamaan Kalman Filter. Proses ini pada dasarnya yang membuat fungsi non-linier di sekitar nilai estimasi yang disebutkan tadi.

Bentukan formula untuk fungsi prediksi *state* (3) dan fungsi prediksi estimasi kovariansi (2.4):

$$x_{\mathcal{k}|\mathcal{k}-1} = \mathcal{f}\left(x_{\mathcal{k}-1|k-1}, u_k\right) \quad (3)$$

$$P_{\mathcal{k}|k-1} = F_{\mathcal{k}} P_{\mathcal{k}-1|\mathcal{k}-1} F_{\mathcal{k}}^{\intercal} + Q_{\mathcal{k}} \quad (4)$$

Bentuk formula sisa pengukuran:

$$\mathcal{y}_{\mathcal{k}} = z_{\mathcal{k}} - h(x_{\mathcal{k}|\mathcal{k}-1}) \quad (5)$$

Bentuk formula kovariansi residual:

$$S\mathcal{k}=H\mathcal{k}P\mathcal{k}|\mathcal{k}-1H\mathcal{k}\intercal+R\mathcal{k}S_{\mathcal{k}} = H_{\mathcal{k}} P_{\mathcal{k}|\mathcal{k}-1} H_{\mathcal{k}}^{\intercal} + R_{\mathcal{k}} \quad (6)$$

Bentuk formula nilai Kalman Optimal:

$$K_k = P_{k|k-1}H_k^T + S_k^{-1} \quad (7)$$

Bentuk formula untuk keadaan yang diperbarui:

$$x_{k|k} = x_{k|k-1} + K_k y_k \quad (8)$$

Bentuk formula kovariansi yang diperbarui:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (9)$$

Bentuk formulasi *state* transisi dan observasi matriks pada Jacobian berikut:

$$F_k = \frac{\partial f}{\partial x}\Big|_{x_{k-1|k-1}, u_k} \quad (10)$$

$$H_k = \frac{\partial h}{\partial x}\Big|_{x_{k|k-1},} \quad (11)$$

Keterangan :
S_k, K_k, $X_{k|k}$, $P_{k|k}$, Y_k, H_k, R_k, Fk = Variabel proses
f = variabel prediksi non-linear
I = Identitas

Persamaan (3) dan persamaan (4) adalah persamaan yang berfungsi untuk menghitung perkiraan nilai keadaan posteriori pada suatu waktu, diberikan pengamatan sampai dengan dan termasuk pada saat itu, dan matriks kovariansi. Sedangkan persamaan (5), (6), (7), (8), (9), (10), dan (11) merupakan persamaan yang berfungsi untuk memperbarui nilai persamaan (3) dan (4) yang telah ada sebelumnya. telah diperoleh.

2.5 Algoritma D* (D-Star)

Algoritma D* (dilafalkan "D star") adalah salah satu dari tiga algoritma pencarian inkremental [14]. Tiga algoritma itu adalah: 1) D* asli, 2) Focussed D*, 3) D* Lite [15] [16] [17].

Ketiga algoritma tersebut memiliki formulasi untuk bisa menyelesaikan proses perencanaan jalur berbasis asumsi yang sama, termasuk perencanaan dengan asumsi di suatu area yang kosong di mana robot harus menavigasikan ke koordinat tujuan yang diberikan di salah satu area yang tidak diketahui [14]. Algoritma ini membuat asumsi tentang area yang tidak diketahui (misalnya jika area itu tidak memiliki hambatan) dan menemukan jalur terpendek dari koordinat tempat robot itu berada ke koordinat tempat tujuan di bawah asumsi-asumsi ini. Ketika mengamati informasi tempat baru (seperti hambatan yang sebelumnya tidak diketahui), ia menambahkan informasi ke dalam memori dan, jika perlu, mengganti rute yang sudah dibuat sebelumnya supaya bisa didapatkan rute terpendek ke koordinat tujuan tadi. Berkat ketiga algoritma D* tersebut, proses tadi bisa diselesaikan dengan cepat meski dieksekusi secara berulang-ulang. Ketiga algoritma D* tersebut bahkan lebih efisien dibanding algoritma A* berulang yang berfungsi sama [14]. Algoritma D* dan variannya tersebut juga banyak diimplementasikan pada robot *mobile* dan navigasi kendaraan otonom, karena itu algoritma D* bisa dijadikan kajian pustaka pada penelitian ini [14].

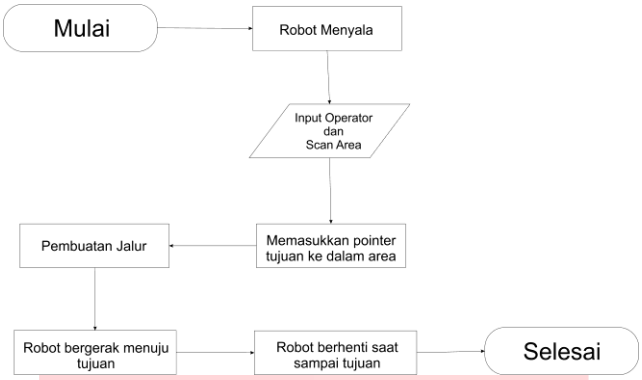
Berikut merupakan beberapa operasi dasar dari algoritma D*:

- BARU, artinya belum pernah ditempatkan di daftar BUKA.
- BUKA , artinya saat ini ada di daftar BUKA.
- DITUTUP, artinya tidak lagi ada daftar BUKA.
- RAISE, menunjukkan biayanya lebih tinggi daripada terakhir kali dalam daftar BUKA.
- RENDAH, menunjukkan biayanya lebih rendah daripada terakhir kali dalam daftar BUKA.

Algoritma D* bekerja secara iteratif memilih sebuah *node* dari daftar BUKA dan mengevaluasinya. Kemudian menyebarkan *node* ke semua *node* yang ada dan menempatkannya pada daftar BUKA. Proses propagasi ini disebut ekspansi. Berbeda dengan A* kanonik yang mengikuti jalur awal hingga akhir, D* dimulai dari penelusuran di *node* tujuan. Setiap *node* yang

diperluas memiliki *backpointer* yang merujuk *node* berikutnya lalu mengarah ke *node* awal, dan setiap *node* memiliki informasi yang tepat dalam menuju tujuan tersebut. Ketika *node* awal diperluas, proses algoritma selesai, dan jalur akan bisa ditemukan ketika menelusuri *backpointer* tersebut. Konsep ini biasa dikenal dengan *backward chaining*.

2.6 Perancangan Algoritma D*



Gambar 1. Diagram Perancangan Algoritma D*.

Pada perancangan Algoritma D* yang digunakan, penulis memfokuskan untuk bisa memperoleh data dari Algoritma D*. Berikut adalah langkah-langkah proses Algoritma D* bekerja:

1. Simulasi dimulai dan Robot di-*spawn* di dalam area yang digunakan untuk simulasi.
2. Robot memulai proses *scan* area simulasi dengan menggunakan *script code* yang menyimpan Algoritma D* sekaligus membuat jalur untuk sampai tujuan.
3. Robot bergerak menuju tujuan dan akan berhenti bergerak di tempat tujuan.

2.7 Gazebo

Gazebo merupakan aplikasi simulasi yang berfungsi untuk mensimulasikan animasi 3D. Gazebo digunakan untuk keperluan simulasi robot ROS yang dihubungkan ke Gazebo. Beberapa versi aplikasi Gazebo dapat mendukung fungsi kerangka kerja ROS. Namun seiring berjalannya waktu, Gazebo dibuat berdiri sendiri tanpa harus menggunakan ROS dalam fungsi simulasinya. Versi Gazebo yang masih dapat mendukung ROS adalah versi Gazebo di bawah 7.0. Versi gazebo di atas 7.0 tidak memerlukan kerangka kerja ROS untuk keperluan simulasi.

2.8 RViZ

RViZ merupakan aplikasi simulasi 3D yang berfungsi untuk menampilkan animasi dengan informasi lengkap yang lengkap mulai dari kondisi peta, Turtlebot, hingga sebaran fungsi node dan topik yang akan disimulasikan. Aplikasi ini sangat berguna untuk mengamati aktivitas fungsional semua perangkat yang berhubungan langsung dengan simulasi. Untuk SLAM, RViZ sangat berguna untuk mengamati aktivitas eksplorasi dalam simulasi, mulai dari kondisi di atas tanah di area simulasi hingga kondisi komponen robot.

2.9 Turtlebot

Turtlebot adalah jenis robot seluler yang digunakan pengembang untuk menerapkan algoritme dan kerangka kerja ROS yang mereka gunakan. Ada beberapa model Turtlebot yang dapat digunakan dalam simulasi yaitu; (1) Model Burger, (2) Model Wafel, (3) Model Pi Waffle.

2.10 ROS (Robot Operating System)

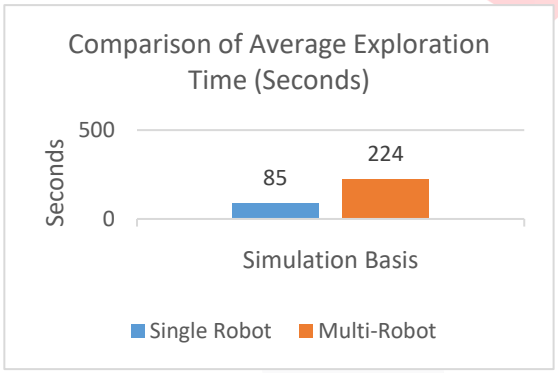
ROS adalah kerangka aplikasi pada Ubuntu 16.04 LTS OS (Sistem Operasi). ROS berfungsi untuk memudahkan developer dalam membuat, mendesain, dan mendesain bentuk dan fungsi robot. ROS memiliki berbagai fungsi dasar robot dalam paketnya. Semua berbagai fungsi disimpan dalam satu node dan topik. ROS adalah open source, yang berarti pengembang dapat memodifikasi fungsionalitas dasar framework. ROS biasanya digunakan oleh developer untuk menguji konsep SLAM sensor LiDAR pada robot menggunakan aplikasi RViZ. ROS juga dapat dihubungkan dengan aplikasi simulasi Gazebo 3D untuk mensimulasikan robot yang telah dibuat.

3. Pembahasan

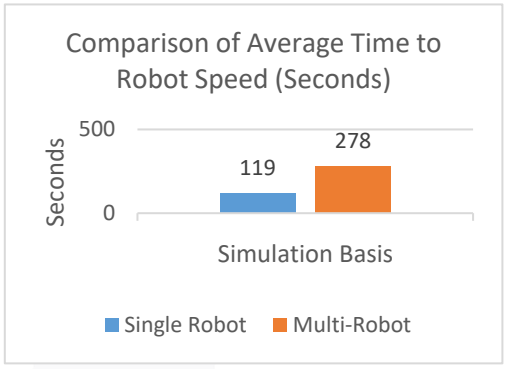
3.1 Pengujian

Pengujian dilakukan dengan asumsi Turtlebot tidak mengetahui daerah simulasi, oleh karena itu Turtlebot akan ditempatkan pada lokasi yang dekat dengan titik (0,0). Pengujian simulasi dilakukan dengan cara membandingkan jumlah informasi yang dapat diperoleh pada suatu waktu tertentu melalui seberapa luas area simulasi yang telah dieksplorasi, kondisi area simulasi, atau informasi lainnya. Pada pengujian ini juga akan ditampilkan berapa lama waktu yang dibutuhkan simulasi untuk memastikan perangkat yang digunakan cukup efektif atau tidak dalam proses komputasi.

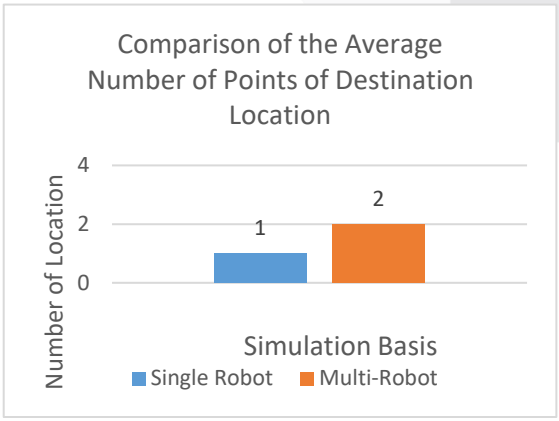
3.2 Hasil Data Pengujian



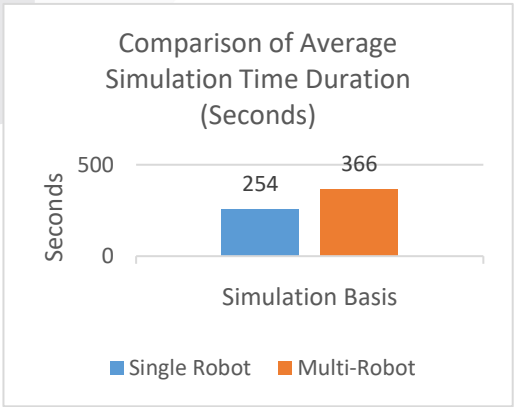
Gambar 2. Hasil Perbandingan Waktu Eksplorasi Simulasi Single-Robot dan Multi-Robot



Gambar 3. Hasil Perhitungan Waktu Eksplorasi Simulasi Single-Robot dan Multi-Robot

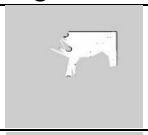
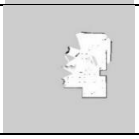
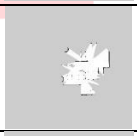
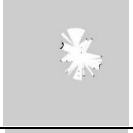


Gambar 3. Hasil Perbandingan Jumlah Lokasi Simulasi Single-Robot dan Multi-Robot



Gambar 4. Hasil Perbandingan Durasi Waktu Simulasi Single-Robot dan Multi-Robot

Tabel 4.10 Perbandingan Luas Area Simulasi yang Berhasil dieksplorasi

Tests	Single-Robot	Multi-Robot
1		
2		
3		
4		
5		
6		
7		
8		

4. Kesimpulan

Dari hasil simulasi yang telah disebutkan sebelumnya bisa diambil beberapa kesimpulan dengan sebagai berikut :

1. Simulasi berbasis multi robot dengan algoritma D * mampu diterapkan dan dapat mengunjungi 2 titik tujuan dibandingkan dengan simulasi robot tunggal yang hanya memiliki 1 titik tujuan. Hal ini sayangnya terbebani oleh waktu simulasi multi robot yang cenderung lebih lama dibandingkan simulasi robot tunggal dengan durasi waktu 254 detik, sedangkan untuk simulasi multi robot berdurasi 366 detik. Simulasi berbasis multi robot menggunakan D * Algoritma dapat diterapkan. Luas area eksplorasi akan bertambah mengingat simulasi berbasis multi robot mampu menjelajahi area simulasi meskipun letak masing-masing Turtlebot berjauhan.
2. Hasil pengujian juga membuktikan bahwa dalam simulasi multi robot, setiap Turtlebot dapat saling mendukung dalam kegiatan eksplorasi. Jika salah satu Turtlebots tidak merespon dan tidak stabil, maka salah satu Turtlebot dapat melanjutkan kegiatan penjelajahannya. Hal ini didukung oleh pengujian 6 dan 7. Hasil dari kedua pengujian tersebut adalah simulasi multi robot mampu mengunjungi minimal 1 titik tujuan dibandingkan dengan simulasi robot tunggal yang tidak dapat mengunjungi satu titik tujuan sama sekali pada pengujian 6.

3. Hasil pengujian membuktikan bahwa area eksplorasi yang berhasil dieksplorasi juga lebih luas pada simulasi multi robot. 7 dari 8 hasil pengujian yaitu selain hasil pengujian ke-7, ternyata wilayah wilayah eksplorasi lebih luas. Bukti ini juga diperkuat pada tes ke-8, hasil eksplorasi ekstensif menunjukkan terbaca lebih jelas meski kedua Turtlebot tidak berhasil mengunjungi satu titik pun tujuan.
4. Berdasarkan poin-poin sebelumnya, algoritma D * dapat diterapkan pada sistem pencarian jalur eksplorasi berbasis multi robot di SLAM. Kesimpulan ini membuktikan bahwa Algoritma D * dapat mendukung eksplorasi berbasis multi robot.

Daftar Pustaka:

[1] Avinash Gautam, & Sudeept Mohan. (2012) "A Review of Research in Multi-Robot Systems". Department of Computer Science and Information Systems Birla Institute of Technology and Science Pilani, India.

[2] Albina, K., & Lee, S. G. "*Hybrid Stochastic Exploration Using Grey Wolf Optimizer and Coordinated Multi-Robot Exploration Algorithms". IEEE Access, 7, 14246–14255, Februari, 2019.*

[3] Jason T.P. Tse, Stephen C.F.Chan & Grace Ngai. (2010) "An Introduction to the Multi-Modal Robot (MuMoMuRo) Control System". Department of Computing. The Hong Kong Polytechnic University Hong Kong SAR, China.

[4] H. Jacky Chang, "P-SLAM: Simultaneous Localization and Mapping With Environmental-Structure Prediction", *IEEE Transaction On Robotics*, vol. 23, No. 2, April 2007.

[5] Ivan Maurovic, Marija Seder, Kruno Lenac, Ivan Petrovic, "*Path Planning for Active* SLAM *Based on the D* Algorithm With Negative Edge Weights*," *IEEE Transaction On Systems*, 2017.

[6] Pip Tools, "Algoritma Extended Kalman Filter". *Pip Tools*, 11 Desember 2015. [Online]. Tersedia : https://piptools.net/algoritma-extended-kalman-filter/ [Diakses: 23 September 2019].

[7] Pip Tools. "Algoritma Kalman Filter". *Pip Tools*, 9 Desember 2015. [Online]. Tersedia: https://piptools.net/algoritma-kalman-filter/ [Diakses: 23 September 2019].

[8] Hongjian Wang, Jing Wang, Liping Qu, Zhenye Liu, "*Simultaneous Localization and Mapping Based on Multilevel-EKF*," Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation, 2255-2258, August. 7 -10, 2011.

[9] Wikipedia. "Extended Kalman Filter". *Wikipedia*, 19 September 2019. [Online]. Tersedia: https://en.wikipedia.org/wiki/Extended_Kalman_filter [Diakses: 23 September 2019].

[10] Wikipedia. "Jacobian Matrix and Determinant". *Wikipedia*, 21 September 2019.[Online].Tersedia https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant [Diakses: 23 September 2019].

[11] Wikipedia. "Model Pedictive Control". *Wikipedia*, 17 September 2019. [Online]. Tersedia: https://en.wikipedia.org/wiki/Model_predictive_control [Diakses: 28 September 2019].

[12] Cindy Leung, Shoudong Huang, Gamini Dissanayake, "Active SLAM using Model Predictive Control and Attractor based Exploration*" Proceedings of the 2006 IEEE/RSJ*, 5027-5031, 9-15 October , 2006.

[13] *J.D. Hedengren; R. Asgharzadeh Shishavan; K.M. Powell; T.F. Edgar (2014). "Nonlinear modeling, estimation and predictive control in APMonitor". Computers & Chemical Engineering.* ***70*** *(5): 133–148.*

[14] *Wikipedia. "D*". Wikipedia, 10 September 2019. [Online]. Tersedia:* https://en.wikipedia.org/wiki/D* *[Diakses: 2 Oktobber 2019].*

[15] *Stentz, Anthony (1994), "Optimal and Efficient Path Planning for Partially-Known Environments", Proceedings of the International Conference on Robotics and Automation: 3310–3317, CiteSeerX 10.1.1.15.3683*

[16] Stentz, Anthony (1995), "*The Focussed D* Algorithm for Real-Time Replanning*", Proceedings of the International Joint Conference on Artificial Intelligence: 1652–1659, CiteSeerX 10.1.1.41.8257

[17] Hart, P.; Nilsson, N.; Raphael, B. (1968), "*A Formal Basis for the Heuristic Determination of Minimum Cost Paths*", IEEE Trans. Syst. Science and Cybernetics, SSC-4 (2): 100–10

[16] *Koenig, S.; Likhachev, M. (2005), "Fast Replanning for Navigation in Unknown Terrain", Transactions on Robotics, **21** (3): 354–363,* CiteSeerX 10.1.1.65.5979

[17] *Koenig, S.; Likhachev, M. (2002) "D*Lite"*

[18] Han, S. D., & Yu, J. (2020). *"DDM: Fast Near-Optimal Multi-Robot Path Planning Using Diversified-Path and Optimal Sub-Problem Solution Database Heuristics", IEEE Robotics and Automation Letters, 5(2), 1350–1357.* doi:10.1109/lra.2020.2967326

[19] *Carter,Mancini, Bruce,Ron (2009). Op Amps for Everyone. Texas Instruments. pp. 10–11.* ISBN 978-0080949482.