TELEKONTRAN, VOL. 8, NO. 1, APRIL 2020
DOI : 10.34010/telekontran.v8i1.3083
p-ISSN : 2303 – 2901
e-ISSN : 2654 – 7384

Algoritma Perencanaan Jalur Kendaraan Otonom Berbasis Hibridisasi Algoritma BFS dan *Path Smoothing*

Path Planning Algorithm for Autonomous Vehicles Based on Hybridization of BFS and Path Smoothing Algorithm

Muhammad Aria Rajasa Pohan
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10 Bandung
Email : muhammad.aria@students.itb.ac.id

Abstrak – Pada penelitian ini dirancang algoritma alternatif untuk perencanaan jalur kendaraan otonom. Algoritma yang diusulkan adalah hibridisasi dari algoritma *Breadth First Search* (BFS) dan algoritma *path smoothing* (BFS – *path smoothing*). Keuntungan dari algoritma BFS adalah dapat memberikan solusi yang menuju solusi optimal. Agar diperoleh solusi yang optimal, maka jalur yang dihasilkan oleh algoritma BFS akan diproses lebih lanjut oleh algoritma path smoothing. Walaupun algoritma BFS - *path smoothing* memiliki waktu komputasi yang tinggi, tetapi untuk tujuan mendapatkan solusi yang optimal, waktu komputasi BFS – *path smoothing* masih lebih rendah daripada algoritma RRT* untuk mendapatkan solusi yang optimal. Algoritma RRT* adalah salah satu algoritma yang umum digunakan untuk perencanaan jalur pada kendaraan otonom. Proses hibridisasi ini dilakukan dengan cara menjalankan algoritma BFS terlebih dahulu untuk memberikan solusi awal. Solusi awal tersebut kemudian ditingkatkan kualitasnya menggunakan algoritma *path smoothing* untuk memperoleh solusi yang optimal. Pengujian algoritma BFS-*path smoothing* ini dilakukan dalam lingkungan yang disimulasikan. Pengujian yang dilakukan meliputi pengujian keamanan, waktu komputasi dan pengujian optimalitas dan *completeness*. Digunakan beberapa kasus *benchmark* yang ada, yaitu lingkungan *narrow*, *maze*, *trap* dan *clutter*. Kriteria optimalitas yang dibandingkan adalah biaya jalur dan waktu komputasi. Pada pengujian, performansi dari algoritma BFS-*path smoothing* dibandingkan dengan performansi dari algoritma RRT*. Kami menunjukkan bahwa algoritma yang diusulkan dapat menghasilkan output jalur dengan kualitas sebaik RRT* tetapi dengan waktu konvergensi yang lebih baik. Maka algoritma BFS-*path smoothing* ini dapat menjadi alternatif algoritma yang digunakan pada perencanaan jalur kendaraan otonom.

Kata Kunci : *Beadth First Search*, *path smoothing*, perencanaan jalur, pengujian simulasi, RRT*

Abstract *- In this paper an alternative algorithm is designed for autonomous vehicle path planning. The proposed algorithm is a hybridization of the Breadth First Search algorithm (BFS) and the path smoothing algorithm (BFS - path smoothing). The advantage of the BFS algorithm is that it can provide solutions that lead to optimal solutions. In order to obtain an optimal solution, then the path produced by the BFS algorithm will be further processed by the path smoothing algorithm. Although the BFS - path smoothing algorithm has a high computational time, but for the purpose of getting an optimal solution, the BFS - path smoothing computation time is still lower than the RRT* algorithm to get the optimal solution. RRT* algorithm is one algorithm that is commonly used for autonomous vehicles path planning. This hybridization process is carried out by first running the BFS algorithm to provide an initial solution. The initial solution is then improved by using the path smoothing algorithm to obtain an optimal solution. The BFS-path smoothing algorithm is tested in a simulated environment. Tests carried out include safety testing, computational and optimality testing and completeness testing. Several benchmark cases are used, namely narrow, maze, trap and clutter environment. The optimality criteria that are compared are path costs and computational time. In testing, the performance of the BFS-path smoothing algorithm is compared with the performance of the RRT* algorithm. We show that the proposed algorithm can produce path output with quality as good as RRT * but with better convergence time. Then the BFS-path smoothing algorithm can be an alternative algorithm used in autonomous vehicle path planning.*

Keywords *: Breadth First Search, path smoothing, path planning, simulation testing, RRT**

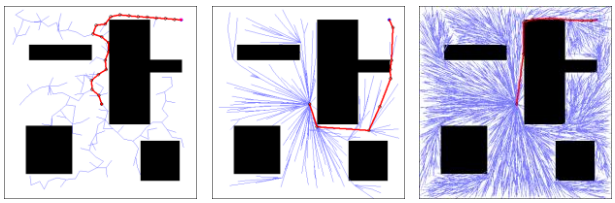
I. PENDAHULUAN

Kendaraan otonom adalah kendaraan tanpa pengemudi yang dapat melakukan perjalanan ke tujuan yang telah ditentukan tanpa memerlukan intervensi pengemudi manusia [1]. Agar kendaraan otonom dapat mencapai tujuan tanpa pengemudi manusia, maka sistem kendaraan otonom harus dapat mengenali kondisi lingkungan dan membangun jalur untuk mencapai tujuan berdasarkan informasi keadaan lingkungan [2,3]. Dalam perencanaan jalur tersebut tentu kendaraan otonom harus mempertimbangkan peta area jalan dan kendaraan lain yang ada pada suatu jalan. Penelitian ini akan berfokus pada pengembangan algoritma alternatif untuk perencanaan jalur pada kendaraan otonom.

Memang telah banyak algoritma yang dapat digunakan untuk perencanaan jalur. Algoritma perencanaan jalur diharapkan dapat merancang jalur yang optimal menuju tujuan dengan menghindari rintangan atau batasan jalan yang ada. Algoritma perencanaan jalur seperti AD* [4] dapat menemukan solusi optimal pada lingkungan yang terdiskritisasi. Namun, penggunaan metode diskritisasi lingkungan yang berdimensi besar dapat menurunkan kinerja algoritma. Penelitian yang dilakukan oleh Pivtoraiko [5] juga masih melibatkan diskritisasi lingkungan. Algoritma *Ant Colony Optimization* (ACS) [6] dapat juga diterapkan untuk perencanaan jalur robot. Tetapi algoritma ACS dapat terjebak pada solusi minimum lokal dan juga algoritma ACS memiliki performansi yang kurang baik pada lingkungan yang sempit. Penelitian [7,8] mengusulkan metode perencanaan gerak reaktif berdasarkan bacaan dari sensor-sensor yang digunakan. Tetapi metode perencanaan gerak reaktif ini tidak dapat dianggap sebagai perencanaan global. Adapun metode berbasis teori kontrol memerlukan formulasi model yang tepat dari lingkungan maupun dari kendaranaan [9]. Pemodelan lingkungan dan kendaraan ini dapat menjadi tugas yang cukup kompleks. Penelitian lain mengusulkan penggunaan perencanaan gerak berbasis sampel atau *Sampling Based Planning* (SBP). Algoritma SBP memiliki keunggulan berupa kemampuan untuk memberikan solusi yang cepat untuk masalah yang kompleks [10]. Algoritma yang paling umum digunakan dari SBP mungkin adalah algoritma *Probabilistic Road-Map* (PRM) [11] dan algoritma *Rapidly-exploring Random Tree* (RRT) [12]. Algoritma PRM memiliki keunggulan dapat memberikan solusi yang bersifat asimptotik optimal, yaitu algoritma PRM dapat memberikan solusi yang optimal jika waktu yang diberikan mencukupi. Tetapi algoritma PRM memiliki waktu komputasi yang tinggi. Adapun algoritma RRT memiliki keuntungan memiliki waktu komputasi yang ringan. Tetapi kelemahan algoritma RRT adalah solusi yang diberikan bersifat sub-optimal [13]. Terdapat beberapa penelitian yang berusaha mengembangkan algoritma RRT. Karaman [14] mengembangkan algoritma RRT menjadi RRT*. Algoritma RRT* mampu menghasilkan solusi yang bersifat asimptotik optimal. Tetapi waktu komputasi dari algoritma RRT* masih cukup tinggi. Contoh perbandingan waktu komputasi antara algoritma RRT dan RRT* dalam menghasilkan solusi optimal ditunjukkan pada Gambar 1. Algoritma RRT dapat menghasilkan solusi awal dalam waktu 0,19 detik sedangkan algoritma RRT* membutuhkan waktu 1,27 detik untuk menghasilkan solusi awal. Dan algoritma RRT* membutuhkan waktu 10,64 detik untuk menghasilkan solusi yang optimal.Akgun dan Stilman [15] mengusulkan algoritma B-RRT*, yaitu versi pencarian jalur seperti RRT* tetapi dilakukan secara dua arah, yaitu dari node awal dan dari node tujuan. Kumar dkk. [16] mengusulkan algoritma yang dinamakan *Goal-Bias-RRT*. Pada algoritma ini, ekspansi node menuju node random dilakukan beberapa kali. Adapun Gammell dkk. [17] mengusulkan algoritma *Informed*-RRT*. Algoritma *Informed-RRT** ini membatasi wilayah pengambilan sampel baru untuk algoritma RRT* agar lebih efisien. Proses pengambilan sampel pada algoritma *Informed-RRT** dilakukan dalam wilayah elips yang mengelilingi node akhir dan node awal. Sedangkan Nasir dkk [18] mengusulkan algoritma RRT*-Smart. RRT*-Smart menggunakan dua fitur utama, yaitu *route optimization* dan *intelligent sampling*. Algoritma *Breadth First Search* (BFS) memang telah diteliti oleh Subramanian dkk. [19] untuk diimplementasikan pada kasus perencanaan jalur robot. Tetapi untuk beberapa kasus, jalur yang dihasilkan tidak benar-benar mencapai kondisi yang optimal. Sepengetahuan penulis, belum ada penelitian sebelumnya yang menggabungkan algoritma Breadth First Search (BFS) dengan algoritma *Path Smoothing* untuk menghasilkan algoritma perencanaan jalur kendaraan otonom.

Kontribusi dari penelitian ini adalah untuk mengusulkan algoritma alternatif dapat digunakan pada perencanaan jalur kendaraan otonom. Algoritma yang diusulkan didasarkan pada integrase algoritma BFS dan algoritma *Path Smoothing* (BFS-*path smoothing*). Proses

hibridisasi dilakukan dengan cara menjalankan algoritma BFS untuk mendapatkan solusi jalur awal. Solusi awal tersebut kemudian ditingkatkan kualitasnya menggunakan algoritma *path smoothing* sehingga memperoleh solusi yang optimal. Untuk memperoleh jalur yang melengkung halus, digunakan penerapan teknik smoothing menggunakan algoritma Reed Sheep Planner. Pengujian algoritma BFS-*path smoothing* dilakukan secara simulasi menggunakan beberapa kasus *benchmark*, yaitu lingkungan *narrow*, *maze*, *trap* dan *clutter*. Performansi yang diukur adalah biaya jalur dan waktu komputasi. Performansi dari algoritma BFS-*path smoothing* kemudian dibandingkan dengan algoritma RRT*. Kami menunjukkan bahwa algoritma yang diusulkan dapat menghasilkan output jalur dengan kualitas yang lebih tinggi daripada jalur yang diproduksi oleh RRT*.



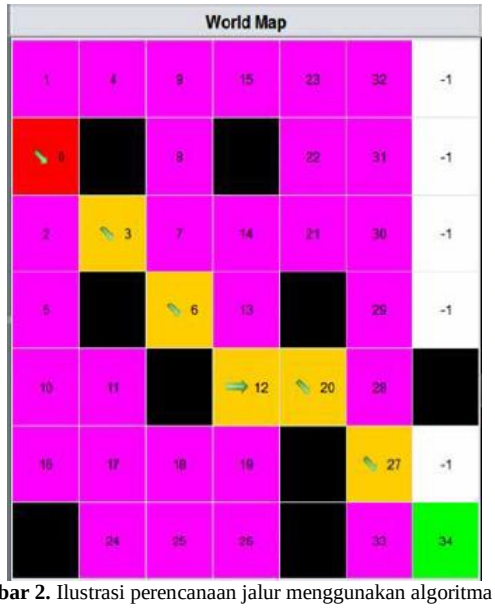
0,19 detik (a) | 1,28 detik (b) | 10,64 detik (c)

Gambar 1. Contoh perbandingan waktu komputasi dari (a) algoritma RRT dan (b) algoritma RRT*. (c) Algoritma RRT* membutuhkan waktu yang lebih lama untuk menghasilkan jalur yang optimal

II. METODE

Proses hibridisasi dilakukan dengan cara menjalankan algoritma BFS untuk mendapatkan solusi jalur awal. Solusi awal tersebut kemudian ditingkatkan kualitasnya menggunakan algoritma *path smoothing* sehingga memperoleh solusi yang optimal. BFS dimulai pada posisi awal (start) dan mulai mencari tujuan dengan memperluas pohon pencarian pada semua node yang belum dikunjungi dan tidak terisi halangan. Pohon pencarian diperluas pada semua arah yang diijinkan bagi agen untuk melakukan perjalanan selanjutnya. Maka dari posisi agen, pohon pencarian dapat diperluas ke node atas, bawah, kiri, kanan, atas kiri, atas kanan, bawah kiri dan bawah kanan. Sehingga ada 8 kemungkinan arah perluasan pohon pencarian. Jika pada suatu node terdapat rintangan, maka pohon pencarian tidak akan diperluas pada node tersebut. Pohon pencarian juga tidak akan diperluas ke node yang sudah dikunjungi.

Contoh ilustrasi perencanaan jalur menggunakan algoritma BFS dapat dilihat pada **Gambar 2** [19]. Pada Gambar 2, node awal adalah kotak dengan warna merah, dan node tujuan adalah kotak dengan warna hijau. Kotak berwarna hitam adalah rintangan yang tidak boleh dilalui oleh jalur yang dibuat. Sedangkan kotak berwarna kuning adalah rancangan jalur yang menghubungkan node awal dengan node tujuan.

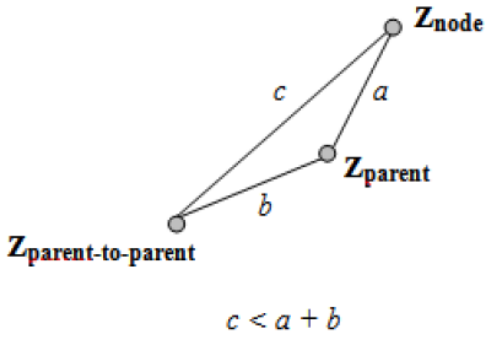


Gambar 2. Ilustrasi perencanaan jalur menggunakan algoritma BFS [19]

Berdasarkan **Gambar 2**, terlihat salah satu kekurangan dari algoritma BFS. Algoritma BFS menyusun jalur melalui kotak 3 – 6 – 12 – 20 – 27. Dalam lingkungan sebenarnya yang tidak terdiskritisasi, seharusnya jalur optimal adalah 3 – 6 – 20 -27. Maka pada algoritma yang diusulkan, agar kotak 6 dapat terhubung langsung ke kotak 20 tanpa melalui kotak 12, akan digunakan algoritma *path-smoothing*.

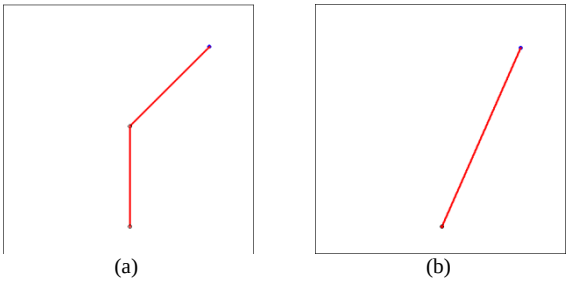
Ilustrasi dari proses *path smoothing* ditunjukkan pada **Gambar 3**. Jika terdapat tiga node (z_{node} - z_{parent} – $z_{parent\text{-}to\text{-}parent}$) dengan cost adalah a dan b, maka fungsi *path optimization* akan memeriksa apakah z_{node} dan $z_{parent\text{-}to\text{-}parent}$ dapat dihubungkan langsung tanpa melalui z_{parent}. Kedua node tersebut dapat dihubungkan jika tidak terdapat *obstacle* diantara keduanya. Maka jika kedua node tersebut dihubungkan akan diperoleh jalur c yang lebih pendek daripada a+b.

Perbandingan jalur yang dihasilkan oleh algoritma BFS dengan BFS-*path smoothing* dapat ditunjukkan pada **Gambar 4**. Pada **Gambar 4(a)** adalah jalur yang dihasilkan algoritma BFS untuk menghubungkan titik start (lingkaran merah) ke titik tujuan (lingkaran biru). **Gambar 4(b)** adalah jaur yang dihasilkan algoritma BFS-*path smoothing*.



$$c < a + b$$

Gambar 3. Ilustrasi proses *path smoothing* [18]



(a) (b)

Gambar 4. Perbandingan jalur yang dihasilkan oleh: (a) algoritma BFS dan (b) BFS-*path smoothing*

Seperti ditunjukkan pada **Gambar 4(a)**, algoritma BFS hanya dapat membangun jalur pada sudut 0^o, 45^0, 90^o, 135^o, 180^o, 225^o, 270^o, 315^0, dan 360^o. Jika jalur terpendek dapat dibangun pada sudut 60^o antara titik awal dan titik akhir, algoritma BFS tidak dapat melakukannya, karena tidak ada pilihan gerak 60^o, pada algoritma BFS. Algoritma BFS harus membangun jalur pada sudut 0^o terlebih dahulu hingga ditemukan jalur bersudut 45^o yang menghubungkan node saat ini dengan node akhir, seperti ditunjukkan pada **Gambar 4(a)**. Jalur yang dibangun oleh algoritma BFS pada **Gambar 4(a)** terdiri dari tiga node yaitu node start, node tengah dan node goal. Dengan menggunakan algoritma *path smoothing*, maka dapat diperiksa apakah node start dapat dihubungkan langsung dengan node goal tanpa melalui node tengah. Maka menggunakan algoritma *path smoothing* dapat diperoleh jalur seperti **Gambar 4(b)**.

III. HASIL DAN DISKUSI

Untuk memverifikasi algoritma yang kami usulkan, tes dilakukan dalam lingkungan yang disimulasikan. Pengujian yang dilakukan meliputi pengujian keamanan, waktu komputasi dan pengujian optimalitas dan *completeness*. Usulan pengujian keamanan adalah untuk memastikan tidak ada tabrakan dengan kendaraan terdekat. Usulan waktu komputasi dan pengujian optimalitas adalah untuk memastikan tidak ada kemacetan karena menunggu pergerakan kendaraan otonom, dan untuk memastikan algoritma dapat memberikan jalur akhir itu. Sedangkan usulan pengujian *completeness* adalah untuk memastikan bahwa algoritma selalu dapat menemukan solusi jalur jika solusi ada

D. Pengujian Keamanan

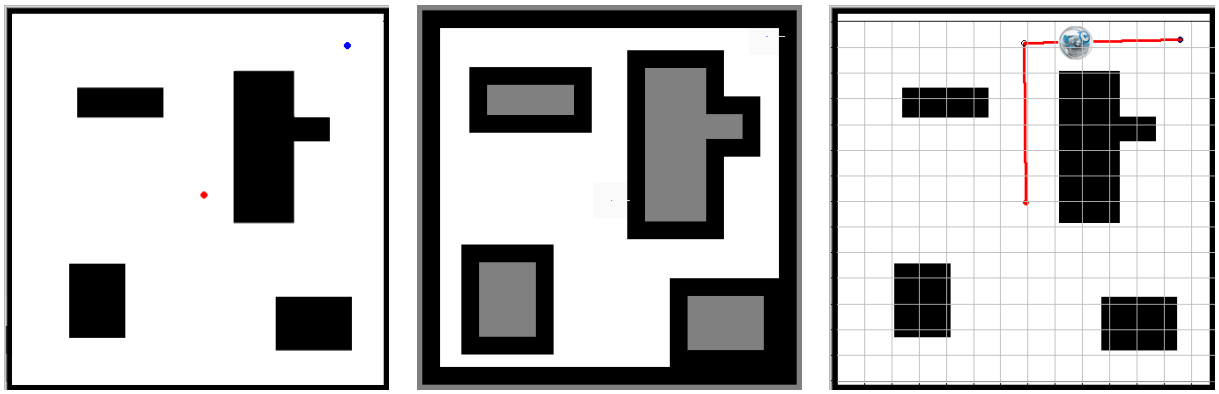
Untuk memastikan bahwa jalur yang dihasilkan memenuhi persyaratan keselamatan, maka digunakan peta *drivability*. Pada peta *drivability*, batas-batas jalur didefinisikan sebagai area yang tidak dapat dilintasi, dan area yang dapat dilintasi dengan aman. Menggunakan peta *drivability* akan membatasi ruang pencarian algoritma sehingga tidak membangun jalur yang akan menyebabkan tabrakan dengan objek lain.

Pengujian algoritma menggunakan peta *drivability* ini ditunjukkan pada **Gambar 5**. Pengujian dilakukan secara simultan menggunakan kendaraan dalam model unicycle yang memiliki dimensi lingkaran dengan jari-jari r. Lingkungan yang akan dilewati ditunjukkan pada **Gambar 5 (a)**. Peta *drivability* akan menambah area hambatan sejauh r sehingga jalur dalam bentuk garis dapat dilalui oleh kendaraan tak berawak dengan radius r tanpa menabrak objek di sekitarnya. Ilustrasi peta drivability ditunjukkan pada **Gambar 5 (b)**. Pergerakan kendaraan unicycle mengikuti jalur yang dihasilkan oleh algoritma ditunjukkan pada **Gambar 5 (c)**. Penggunaan peta drivability ini sama dengan Sivaraman [20] dan Tawari [21].

E. Pengujian Waktu Komputasi dan Pengujian Optimalitas

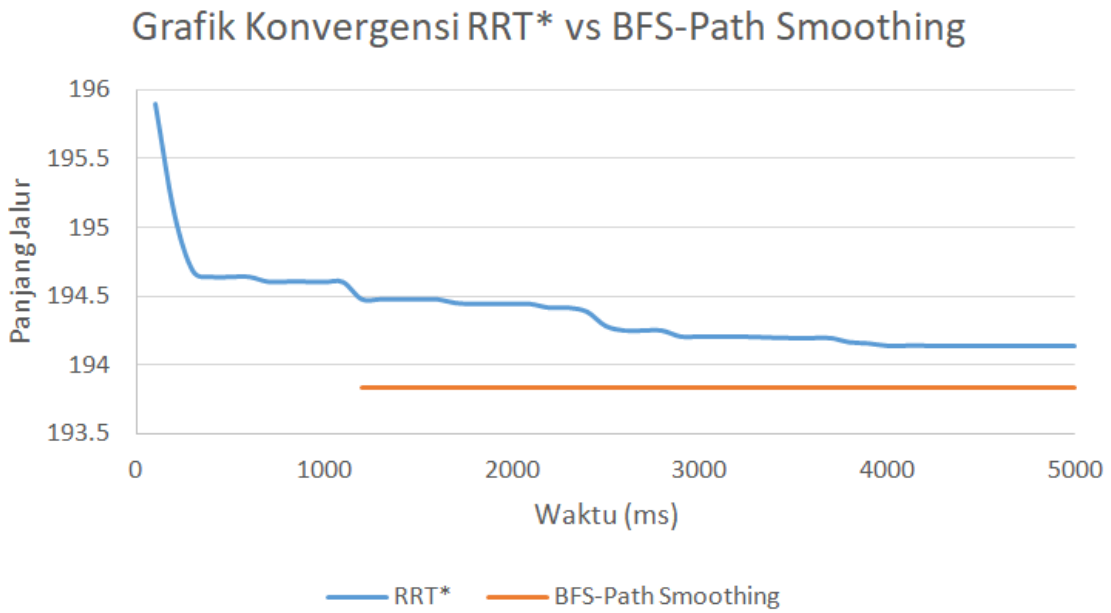
Pengujian waktu komputasi dilakukan dengan membandingkan kinerja algoritma yang diusulkan dengan algoritma perencanaan jalur standar (sering digunakan), yaitu algoritma RRT* [22]. Kinerja yang dibandingkan adalah waktu yang diperlukan untuk menghasilkan jalur yang mendekati jalur optimal untuk kasus tertentu. Terdapat enam skenario pengujian yang dilakukan, yaitu pengujian dalam kasus lingkungan tanpa hambatan, dengan hambatan, *narrow*, *maze*, *trap* dan *clutter*.

Gambar 6 – **Gambar 11** menunjukkan hasil pengujian algoritma BFS-*path smoothing* dan RRT* pada lingkungan tanpa hambatan. Masing-masing pengujian dilakukan sebanyak 20 kali. Hasil pengujian setiap algoritma lalu dirata-ratakan dan diplot pada grafik.

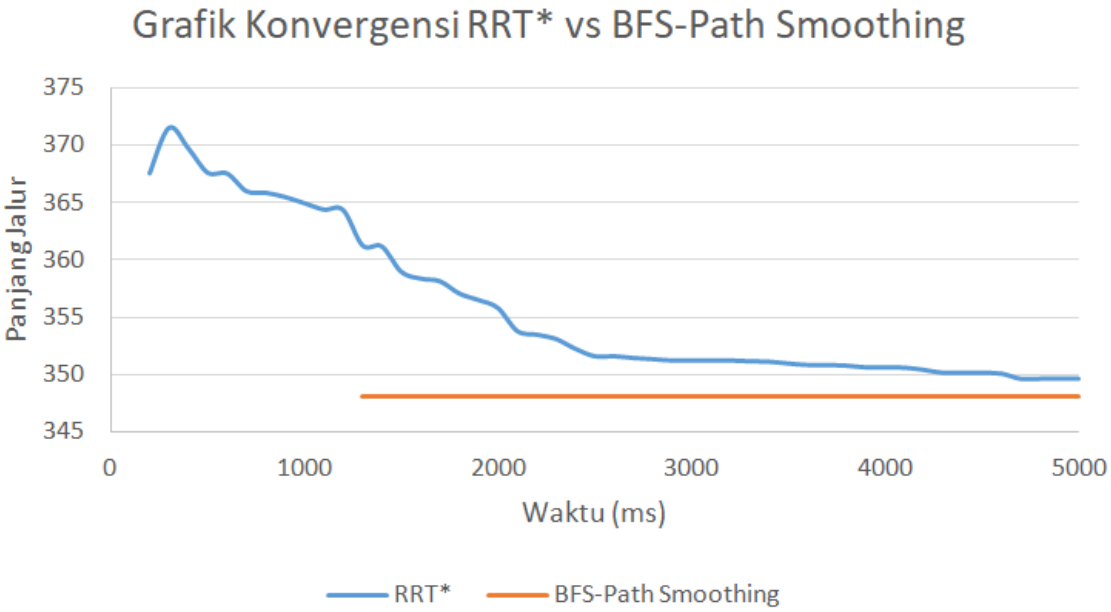


(a) (b) (c)

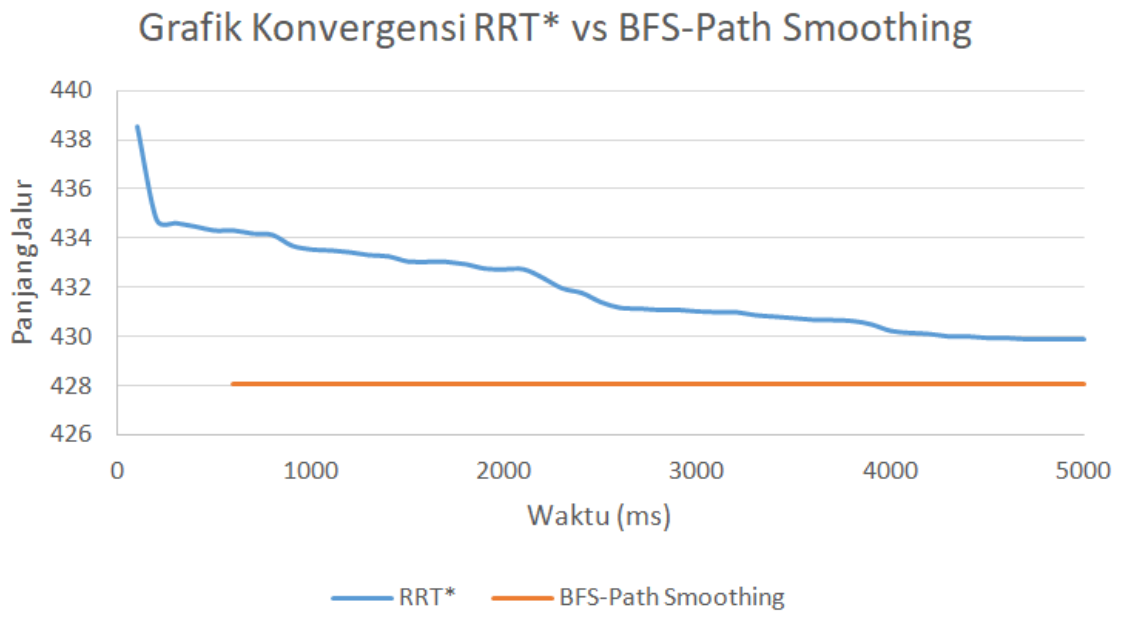
Gambar 5. (a) Lingkungan pengujian, (b) drivability map, (c) kendaraan unicycle dapat mengikuti jalur yang dibangun tanpa menabrak rintangan yang ada disekitarnya



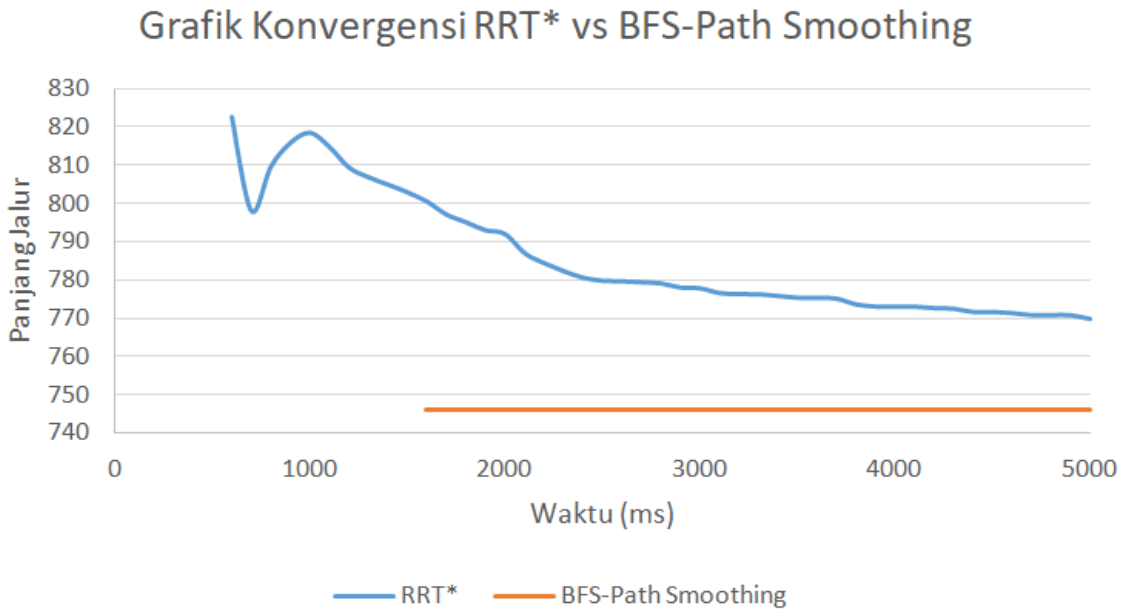
Gambar 6. Perbandingan panjang jalur yang dihasilkan oleh algoritma RRT* dan BFS-*path smoothing* terhadap waktu komputasi untuk lingkungan tanpa hambatan



Gambar 7. Perbandingan panjang jalur yang dihasilkan oleh algoritma RRT* dan BFS-*path smoothing* terhadap waktu komputasi untuk lingkungan dengan hambatan



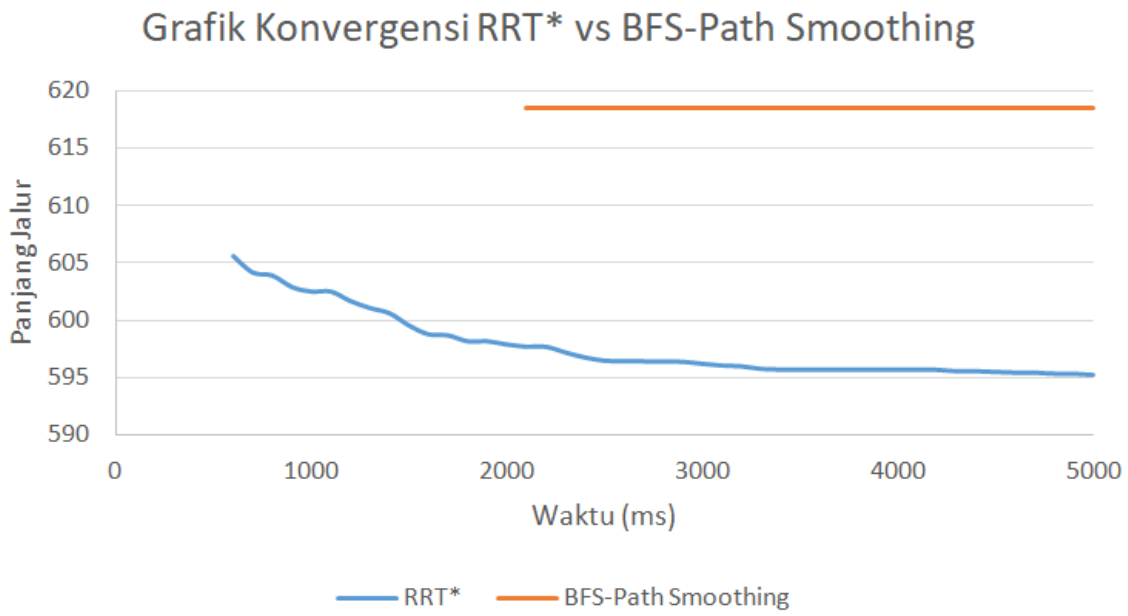
Gambar 8. Perbandingan panjang jalur yang dihasilkan oleh algoritma RRT* dan BFS-*path smoothing* terhadap waktu komputasi untuk lingkungan *narrow*



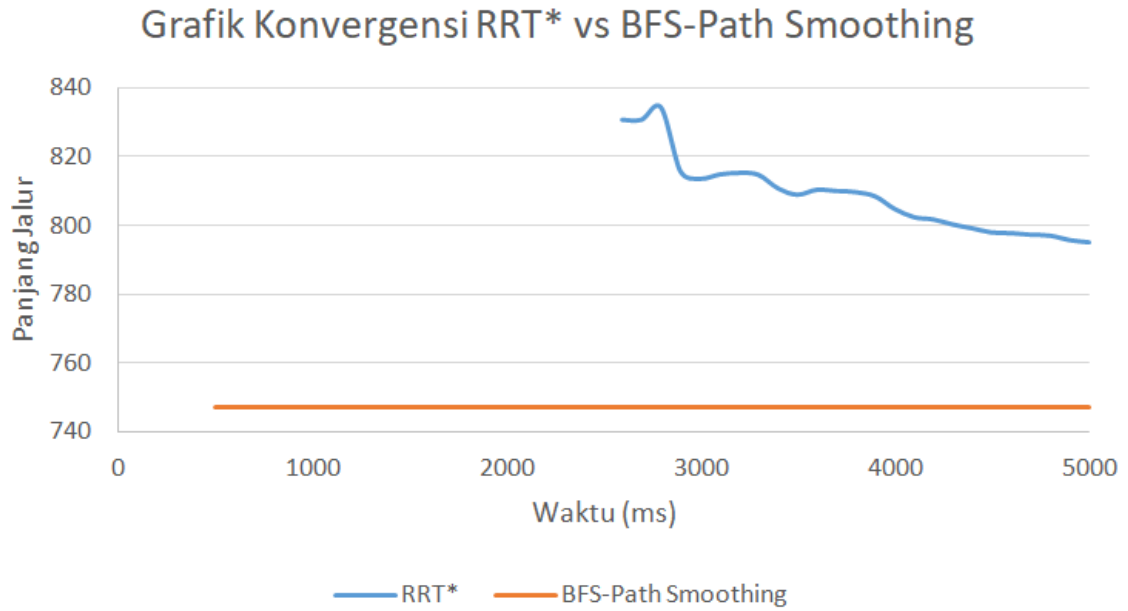
Gambar 9. Perbandingan panjang jalur yang dihasilkan oleh algoritma RRT* dan BFS-*path smoothing* terhadap waktu komputasi untuk lingkungan *trap*

Pada **Gambar 6** – **Gambar 11**, garis biru menunjukkan panjang jalur yang dihasilkan oleh algoritma RRT*. Sesuai sifatnya yaitu asymptotic optimal [13], algoritma RRT* dapat mengalami peningkatan kualitas jalur seiring waktu iterasi. Semakin lama, panjang jalur yang dihasilkan oleh algoritma RRT* menjadi semakin pendek. Belum adanya garis biru selama beberapa waktu semenjak waktu ke-0 disebabkan algoritma RRT* membutuhkan waktu beberapa lama sebelum pohon pencarian dapat menghubungkan node awal dengan node tujuan.

Dari **Gambar 6 – Gambar 9** dan **Gambar 11** menunjukkan algoritma RRT* dapat membangun jalur yang menghubungkan node awal dengan node tujuan lebih cepat daripada algoritma BFS-*path smoothing*. Tetapi walaupun algoritma BFS-*path smoothing* membutuhkan waktu yang lebih lama untuk menghubungkan node awal dengan node tujuan, tetapi jalur yang dihasilkan adalah jalur yang minimal. Terlihat seiring berjalannya waktu, kualitas jalur yang dihasilkan oleh algoritma RRT* belum dapat sebaik kualitas jalur yang dihasilkan oleh algoritma BFS-*path smoothing*.



Gambar 10. Perbandingan panjang jalur yang dihasilkan oleh algoritma RRT* dan BFS-*path smoothing* terhadap waktu komputasi untuk lingkungan *maze*

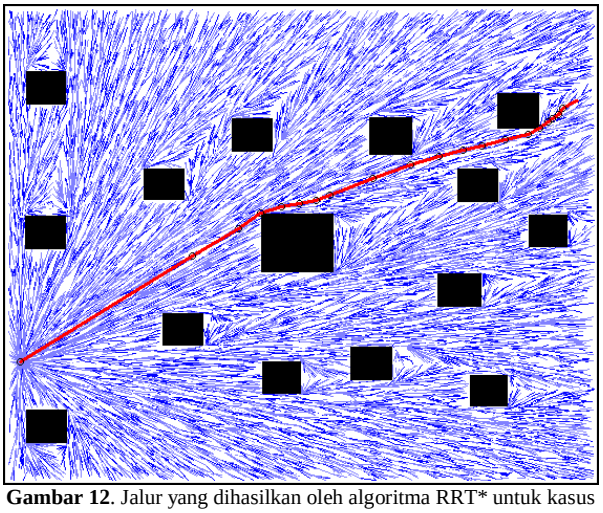


Gambar 11. Perbandingan panjang jalur yang dihasilkan oleh algoritma RRT* dan BFS-*path smoothing* terhadap waktu komputasi untuk lingkungan *clutter*

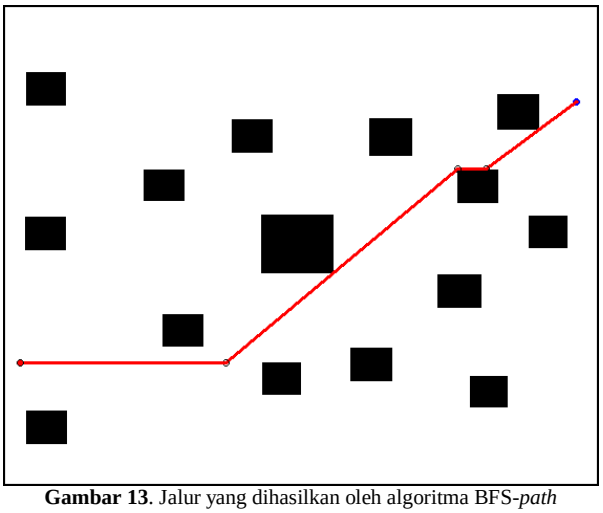
Perbedaan hasil dari pembahasan sebelumnya ditunjukkan pada **Gambar 10**. Pada **Gambar 10** ini, yaitu pengujian pada kasus *clutter*, algoritma RRT* justru menunjukkan performansi yang lebih baik daripada jalur yang dihasilkan oleh algoritma BFS-*path smoothing*. Sebagai ilustrasi, **Gambar 12** menunjukkan jalur yang dihasilkan oleh algoritma RRT* pada kasus *clutter*, sedangkan **Gambar 13** menunjukkan jalur yang dihasilkan oleh algoritma BFS-*path smoothing*.

F. Pengujian *completeness*

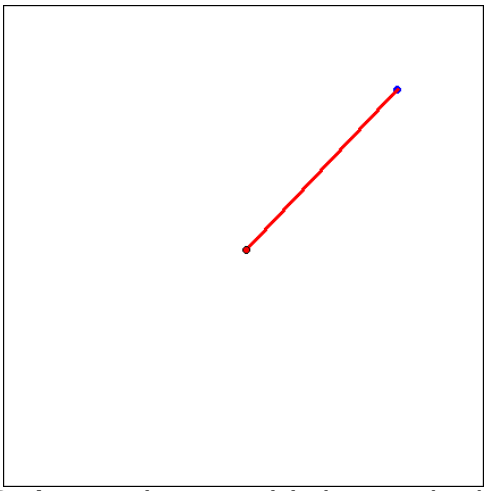
Pengujian *completeness* ini dilakukan dengan menjalankan algoritma dalam beberapa kasus benchmark. Pengujian dilakukan dalam kasus lingkungan tanpa hambatan, dengan hambatan, *narrow*, *maze*, *trap* dan *clutter*. Hasil tes algoritma pada lingkingan tanpa hambatan ditunjukkan pada **Gambar 14**. Hasil tes algoritma pada lingkungan dengan hambatan ditunjukkan pada **Gambar 15**. Hasil tes algoritma dalam lingkungan *narrow* ditunjukkan pada **Gambar 16**. Hasil tes algoritma dalam lingkungan maze ditunjukkan pada **Gambar 17**. Hasil tes algoritma dalam lingkungan perangkap ditunjukkan pada **Gambar 18**. Dan hasil tes algoritma dalam kekacauan lingkungan ditunjukkan pada **Gambar 19**. Dari enam tes ini menunjukkan bahwa algoritma berhasil untuk senantiasa membangun jalur dari posisi awal ke posisi tujuan.



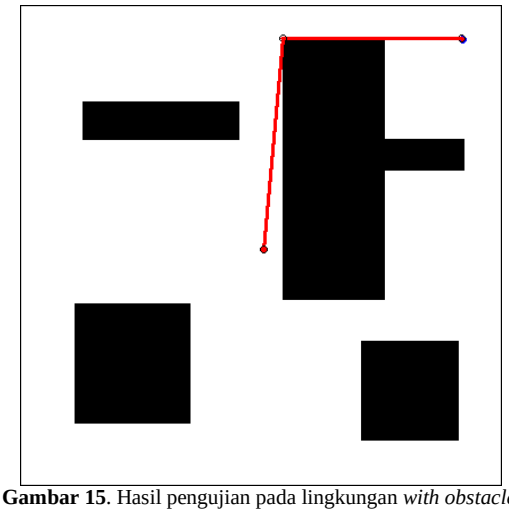
Gambar 12. Jalur yang dihasilkan oleh algoritma RRT* untuk kasus *clutter*. Panjang jalur yang dihasilkan adalah 592.61



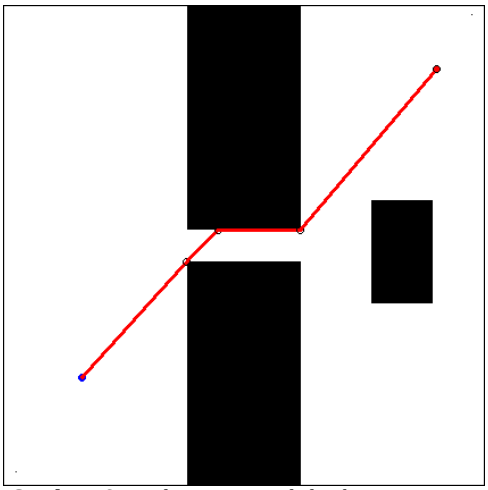
Gambar 13. Jalur yang dihasilkan oleh algoritma BFS-*path smoothing* untuk kasus *clutter*. Panjang jalur yang dihasilkan adalah 618.412



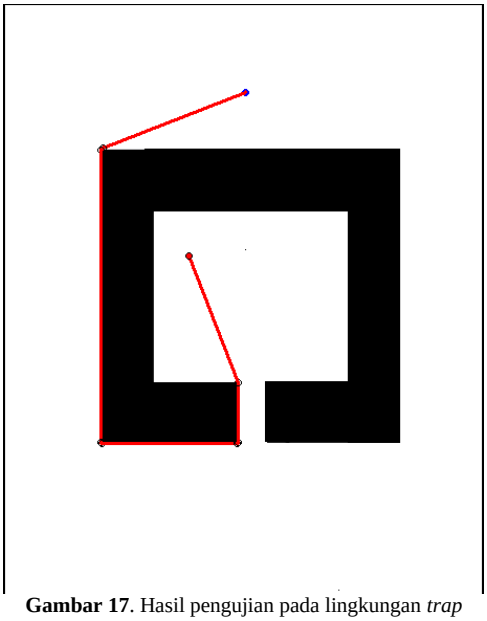
Gambar 14. Hasil pengujian pada lingkungan *no obstacle*



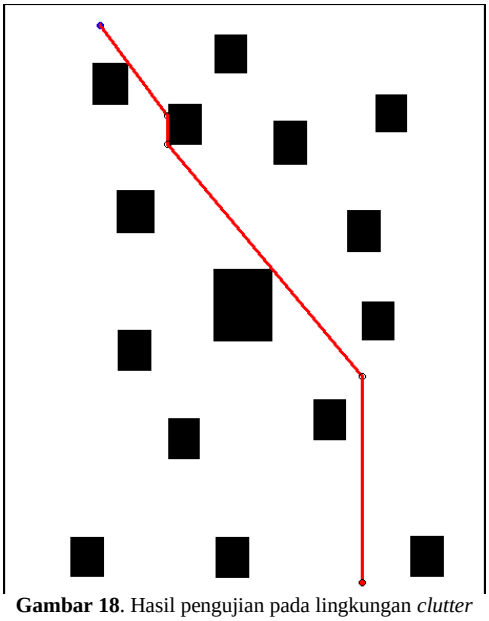
Gambar 15. Hasil pengujian pada lingkungan *with obstacle*



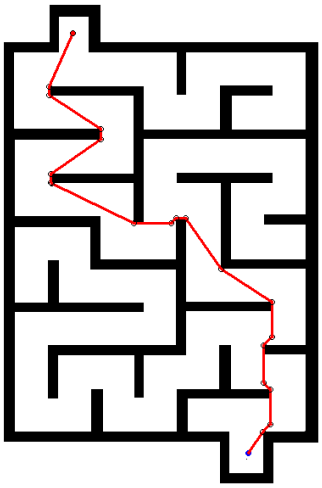
Gambar 16. Hasil pengujian pada lingkungan *narrow*



Gambar 17. Hasil pengujian pada lingkungan *trap*



Gambar 18. Hasil pengujian pada lingkungan *clutter*



Gambar 19. Hasil pengujian pada lingkungan *maze*

IV. KESIMPULAN

Makalah ini telah membahas desain algoritma perencanaan jalur yang dapat diterapkan pada kondisi jalan di Indonesia di mana terdapat kepadatan di jalan dan pergerakan acak kendaraan lain. Algoritma yang diusulkan telah melakukan pengujian keamanan, pengujian waktu komputasi, pengujian optimalitas dan pengujian kelengkapan. Pengujian keamanan dilakukan untuk memastikan keamanan pergerakan sehingga tidak menabrak kendaraan lain. Pengujian waktu komputasi dilakukan untuk memastikan tidak menyebabkan kemacetan atau antrian di belakang kendaraan otonom. Pengujian optimal dilakukan untuk memastikan bahwa algoritma ini menghasilkan jalur pergerakan kendaraan otonom yang optimal. Penelitian ini dapat membantu pengembangan kendaraan otonom untuk implementasi dalam kondisi jalan di Indonesia.

REFERENCES

[1] J. Zhao, B. Liang, dan Q. Chen, "The key technology toward the self-driving car," *International Journal of Intelligent Unmanned Systems*, vol. 6, no. 1, pp. 2-20, (2018)

[2] O.Garcia, R. Lemos, and J. V. Ferreira, "Local and Global Path Generation for Autonomous Vehicles Using Splines," *Ingeniería*, vol. 21 no. 2, pp. 188-200, (2016)

[3] S. Liu, L. Li, J. Tang, S. Wu, and J. L. and Gaudiot, "Creating Autonomous Vehicle System," *Synthesis Lectures on Computer Science*, vol. 6 no. 1, pp. i-186, (2017)

[4] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, dan S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613-1643, (2008)

[5] M. Pivtoraiko, R. A. Knepper, dan A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308-333, (2009)

[6] M. P. Garcia, O. Montiel, O. Castillo, R. Sepúlveda, dan P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Applied Soft Computing*, vol. 9, no. 3, pp. 1102-1110, (2009)

[7] F. Belkhouche, dan B. Bendjilali, "Reactive path planning for 3-D autonomous vehicles," *IEEE transactions on control systems technology*, vol. 20, no. 1, pp. 249-256, (2011)

[8] B. Manup, dan P. Raja, "Collision-avoidance for mobile robots using region of certainty: A predictive approach," *Journal of Engineering Science and Technology*, vol. 11, no. 1, pp. 18-28, (2016)

[9] M. Werling, S. Kammel, J. Ziegler, dan L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 346-359, (2012)

[10] M. Elbanhawi, dan M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56-77, (2014)

[11] L. E. Kavraki, P. Svestka, J. C. Latombe, dan M. H. Overmars, " Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566-580, (1996)

[12] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Tech. Rep., Computer Science Dept., Iowa State University*, pp. 1-4, (1998)

[13] S. Karaman, dan E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846-894, (2011)

[14] S. Karaman, dan E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," *In 49th IEEE conference on decision and control (CDC)*, pp. 7681-7687, (2010)

[15] B. Akgun, dan M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," *In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2640-2645, (2011)

[16] B. S. Kumar, P. S. Abhimanyu, B. V. V. P Bharagav, P. Agarwal, dan K. M. Krishna, "RoboCup SSL Team Description," *IRL RC*, pp. 1-4, (2010)

[17] J. D. Gammell, S. S. Srinivasa, S. S. dan T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," *In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997-3004, (2014)

[18] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, dan M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation of RRT," *International Journal of Advanced Robotic Systems*, vol. 10, no. 7, pp. 1-12, (2013)

[19] M. B. Subramanian, D. K. Sudhagar, dan G. RajaRajeswari, "Intelligent Path Planning Of Mobile Robot Agent By Using Breadth First Search Algorithm," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 3, pp. 1951-1955, (2014)

[20] S. Sivaraman, dan M.M. Trivedi, "Dynamic probabilistic drivability maps for lane change and merge driver assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2063-2073, (2014)

[21] A. Tawari, S. Sivaraman, M. M. Trivedi, T. Shannon, dan M. Tippelhofer, "Looking-in and looking-out vision for urban

intelligent assistance: Estimation of driver attentive state and dynamic surround for safe merging and braking," *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 115-120, (2014)

[22] M. Elbanhawi, dan M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56-77, (2014)

[23] D. J. Webb, dan J. Van Den Berg, " Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," *2013 IEEE International Conference on Robotics and Automation*, pp. 5054-5061, (2013)