ANALISIS SIMULASI EKSPLORASI SLAM BERBASIS FRONTIER MENGGUNAKAN MULTI-ROBOT

FRONTIER BASED SLAM SIMULATION ANALYSIS EXPLORATION USING MULTI-ROBOT

Raka Prasetya Nugraha[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]rakaraih@student.telkomuniversity.ac.id, [2]agungnj@telkomuniversity.ac.id, [3]muhammadfaris@telkomuniversity.ac.id

Abstrak

Penggunaan eksplorasi berbasis *frontier* untuk melakukan eksplorasi secara menyeluruh pada wilayah tidak dikenali dengan menggunakan *SLAM* serta *Navigation Stack*. Hasil pemetaan yang dilakukan *SLAM* digunakan sebagai *frontier point* dengan menggunakan *marker msgs* sebagai tujuan navigasi robot dengan memperhitungkan *frontier cost* dari hasil selisih *costmap* menggunakan parameter yang ada pada *navigation stack* dan *explore_lite*. Eksplorasi berbasis *frontier* akan diberhentikan jika pemetaan telah menyeluruh atau terdapat kendala dimana eksplorasi terlalu lama sehingga robot memutuskan menghentikan *frontier point* tersebut. Map merging digunakan untuk menyatukan hasil pemetaan setiap robot dimana dibutuhkan penggabungan tf robot menggunakan *tf_static*, pada *tf_static* dibutuhkan salah satu peta untuk dijadikan *parent tree* yang digunakan *child tree* robot untuk menggabungkan peta setiap robot ke *parent tree*. Pergerakan robot dari kedua simulasi dapat dilihat menggunakan *ground route*, yakni membandingkan posisi robot dalam peta dan *environment simulation* yang dapat dilihat pada grafik plot odometri dan *processing time* yang dilakukan robot.
Kata Kunci: *Multi-Robot, Frontier-Based, SLAM, Motion Planning*.

Abstract

The use of frontier-based exploration to conduct thorough exploration of unknown areas using SLAM and the Navigation Stack. The results of the mapping carried out by SLAM are used as frontier points using the msgs marker as the destination for robot navigation by calculating the frontier cost from the difference in the costmap using the parameters on the navigation stack and explore_lite. Frontier-based exploration will be stopped if the mapping has been comprehensive or there are obstacles where the exploration takes too long so the robot decides to stop the frontier point. The merging map is used to unify the mapping results of each robot where it is necessary to merge the tf robot using tf_static, in tf_static one of the maps is needed to be the parent tree used by the child tree robot to combine the maps of each robot into the parent tree. The movement of the robot from the two simulations can be seen using a ground route, which is comparing the position of the robot on the map and the simulation environment which can be seen in the odometric plot graph and the robot's processing time.

Keywords: *Multi-Robot, Frontier-Based, SLAM, Motion Planning*.

1. Pendahuluan

Eksplorasi berbasis *frontier* digunakan robot untuk menelusuri wilayah secara menyeluruh dimana robot akan bergerak menuju perbatasan wilayah yang belum ditelusuri hingga tidak menyisakan wilayah untuk dapat ditelusuri, dengan menggabungkan multi-robot dengan eksplorasi berbasis *frontier* maka hal ini dapat menjamin bahwa penelusuran menggunakan multi-robot dapat dilakukan secara menyeluruh dan menghasilkan peta yang akurat dalam menelusuri wilayah tidak dikenali [1][2]. Penggunaan penelusuran berbasis *frontier* dibutuhkannya *SLAM (Simultaneous Localization and Mapping)* dimana cara kerja dari *SLAM* yakni melakukan lokalisasi dan mensimulasikan robot dalam waktu yang bersamaan sehingga hasil pemetaan wilayah yang sedang berlangsung dapat dipantau secara *real-time* [3][4][5]. Untuk robot dapat berjalan secara otonom menggunakan metode *frontier exploration* maka diperlukan beberapa aspek di dalamnya yakni *LSM (Laser Scan Matching), Gmapping, serta Navigation Stack* [6]. Metode *scan matching* yang terdapat pada *LSM* yakni terbagi menjadi *Monte-Carlo scan matcher, Olson scan matcher,* dan *Hough scan matcher* [7]. *Gmapping* merupakan salah satu metode yang pada *SLAM* dimana pada *gmapping* didasari oleh *RBPF (Rao-Blackwellized Particle Filter)* dimana penggunaan *RBPF* umunya terbagi menjadi empat langkah yakni: *Sampling, Importance Weighting, Resampling, dan Map Estimation* [8]. *Navigation stack* merupakan *framework* yang telah disediakan oleh *ROS (Robot Operating System)* dimana pada dasarnya merupakan sebuah *controller* yang mengatur perintah kecepatan yang memungkinkan robot untuk mencapai tujuan yang telah diberikan, menghindari rintangan, dan mengikuti jalur yang telah ditentukan dengan menghitung posisi robot dari pembacaan sensor [9][10][11]. Hasil pemetaan setiap robot kemudian disatukan membentuk peta keseluruhan pada wilayah tidak dikenali [12]. Berdasarkan keterangan yang sudah dijelaskan maka penelitian ini bertujuan untuk membuktikan penggunaan multi robot dalam melakukan penelusuran dan pemetaan berbasis *frontier* untuk menelusuri wilayah tidak dikenali secara menyeluruh. Aplikasi yang digunakan untuk simulasi penelusuran berbasis *frontier* menggunakan *ROS* dan Gazebo, dengan menggunakan metode dan peralatan yang ada diharapkan *frontier-based exploration* dapat melakukan pemetaan secara menyeluruh.

2. Dasar Teori

2.1 SLAM

SLAM digunakan dalam penelusuran wilayah tidak dikenali dikarenakan SLAM dapat melokalisasi dan melakukan pemetaan secara simultan dimana hal ini digunakan pada metode *frontier* untuk menelusuri batas wilayah tidak dikenali sehingga robot dapat bergerak ke suatu wilayah secara otonom, dikarenakan *SLAM gmapping* merupakan salah satu aspek yang dibutuhkan metode *frontier* untuk dapat melakukan penelusuran berbasis *frontier* [6][13].

2.1.1 Gmapping

Gmapping merupakan salah satu percabangan metode dari SLAM dikarenakan *gmapping* didasari oleh cara kerja *Rao-blackwellized Particle Filter* dimana pemetaan berbasis *grid* menggunakan partikel untuk pembuatan peta, setiap partikel mempunyai nilai sendiri terhadap posisi dan pembuatan map [6][14]. *Gmapping* yang didasari oleh *RBPF* terdapat *scan-matcher* dimana untuk langkah optimasi pada *gmapping* yakni sebagai berikut:

1. *Measurement*: pengukuran yang diambil terhadap pemindaian laser.
2. *Scan matching*: pemindaian sebelum dan pada saat ini dicocokkan.
3. *Sampling*: partikel baru akan memberitahukan pose robot menggunakan *motion model (odometry)* dan partikel baru akan dievaluasi seberapa baik partikel baru sesuai dengan peta yang diperoleh *(scan-matching)* dan jika *scan-matcher* dinyatakan telah sesuai maka pada partikel sekitar akan dikembalikan oleh *scan-matcher*. Setelah nilai yang ada pada *scan-matcher* dikembalikan maka *Gaussian approximation* $N(\mu_t^{(i)}, \sum_t^{(i)})$ didapatkan untuk *"improved proposal distribution"* dari mana pose baru (partikel) $x_t^{(i)}$ diperkirakan.
4. *Importance weighting*: bobot yang ada partikel $\omega_t^{(i)}$ kemudian diukur seberapa baik partikel dari distribusi *"improved proposal distribution"* merepresentasikan distribusi target untuk setiap partikel yang didasari *"importance sampling principal"* adalah sebagai berikut:

$$w_t^i = \frac{p(x_{1:t}^i | z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^i | z_{1:t}, u_{1:t-1})} \quad (2.1)$$

Persamaan 2.1 merupakan hasil perhitungan bobot yang ada pada *RBPF* (w_t^i) dengan membagi nilai estimasi pose robot $p(x_{1:t}^i | z_{1:t}, u_{1:t-1})$ dengan scan matching *Monte Carlo approximation* $\pi(x_{1:t}^i | z_{1:t}, u_{1:t-1})$ untuk diperhitungkan bobot partikel yang digunakan.

5. *Resampling*: Tergantung pada hasil N_{eff}, dimana N_{eff} mendeskripsikan seberapa baik distribusi target di representasikan oleh partikel yang diperoleh dari sampling step, dimana resampling step sebagai berikut:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (\tilde{\omega}^{(i)})^2} \quad (2.2)$$

Persamaan 2.2 merupakan perhitungan untuk partikel dengan membagi nilai dengan sampel yang sudah dilakukan menggunakan *Gaussian approximation* yang digunakan untuk memilih seberapa baik partikel yang diperoleh untuk digunakan *map estimation* untuk direpresentasikan ke dalam pemetaan.

6. *Map estimation*: peta yang dibawa oleh masing-masing partikel diperbaharui dengan memperhitungkan *pose* robot dan pengamatan yang didapat melalui fungsi:

$$p(m_t^{(i)} | x_{1:t}^{(i)}, z_{1:t}) \quad (2.3)$$

Persamaan 2.3 merupakan persamaan hasil pengamatan dengan memperhitungkan peta ($m_t^{(i)}$) berdasarkan sampel $x_t^{(i)}$ yang kemudian di tampilkan ke peta untuk digunakan *frontier* sebagai pembentukan *frontier point* menggunakan *marker msgs*.

Metode *gmapping* selain didasari oleh *RBPF* penggunaan *occupancy grid* memungkinkan untuk menggabungkan berbagai sensor dapat berjalan lancar pada *grid* resolusi tinggi. *Occupancy grid* dapat didefinisikan juga sebagai rasio waktu pada *grid* yang terlihat dan terbaca oleh sensor dimana wilayah tidak dikenali dapat diinisialisasi setengah dari nilai *occupancy* dari hasil pemindaian, berikut merupakan persamaan *grid* pada resolusi tinggi:

$$p(x, y) = \frac{\#occupied}{\#occupied + \#empty} \quad (2.4)$$

Persamaan 2.4 menjelaskan peta yang ditampilkan dari hasil pemindaian akan ditampilkan ke dalam map untuk mengetahui wilayah yang sudah diketahui dari hasil pemindaian laser (*occupied*) dan wilayah yang belum terpindai (*empty*), hasil *occupied* kemudian akan dibagi dengan penjumlahan *occupied* dan *empty* untuk direpresentasikan pada *grid* pada resolusi tinggi.

2.2 Dynamic Window Approach

Dynamic Window Approach (DWA) merupakan *velocity search method* untuk melakukan mengatur *translation* dan *rotational velocity* robot yang dilakukan secara langsung dalam ruang *velocity* untuk menghindari *reactive collision* pada robot, selain itu *DWA* juga menggabungkan dinamika robot untuk dapat melakukan penelusuran [15].

2.2.1 Search Space

1. Memilih *velocity* dua dimensi pada search space dimana hanya lintasan melingkar ditentukan secara unik oleh *translational* dan *rotational velocity* (v, w).
2. Menemukan pasangan yang dapat menjamin lintasan aman robot berhenti sebelum mencapai *obstacle* terdekat pada *corresponding curvature* dengan himpunan v_a, berikut merupakan persamaan dari v_a:

$$v_a = \{(v, w) \mid v \leq \sqrt{2\, dist(v, w)\, \dot{v}_b} \wedge w \leq \sqrt{2\, dist(v, w)\, \dot{w}_b}\} \quad (2.5)$$

Persamaan 2.5 digunakan untuk membatasi kecepatan robot dengan memilih lintasan yang telah ditentukan (v, w) untuk kecepatan translasi dan rotasi serta menemukan pasangan yang dapat diterima untuk memastikan lintasan aman robot sebelum mencapai *obstacle* yang ada pada *corresponding curvature*.

3. Menentukan *dynamic window* dengan membatasi *velocity* yang diterima (v_a) untuk yang dapat dicapai dalam interval waktu singkat mengingat akselerasi robot yang terbatas. Kemudian v_d dengan interval waktu *(t)* didefinisikan. Berikut merupakan persamaan dari v_d:

$$v_d = \{(v, w) \mid v \in [v_a - \dot{v}\, t, v_a + \dot{v}\, t] \text{ ^ } w \in [w_a - w\, t, w_a + w\, t]\} \quad (2.6)$$

Persamaan 2.6 menjelaskan pembatasan ruang kecepatan pada persamaan 2.5 yakni dengan memberikan waktu interval dengan akselerasi waktu yang dibatasi pada parameter yang telah ditentukan.

2.2.2 Objective Function

Hasil yang didapat dari *search space* digunakan pada *objective function* sebagai pemabatasan kecepatan dan interval waktu penelusuran yang kemudian digunakan pada persamaan berikut:

$$G(v, w) = \sigma(\alpha\, heading(v, w) + \beta\, dist(v, w) + \gamma\, velocity(v, w)) \quad (2.7)$$

Persamaan 2.7 dari persamaan tersebut maka penjelesan terbagi menjadi empat poin yakni *target heading*, *clearance*, *velocity* dan *smoothing* sebagai berikut:

1. Target heading, Fungsi $heading(v, w)$ mengevaluasi perbedaan antara sudut titik target dan sudut posisi yang diprediksi robot untuk bergerak dari posisi sekarang menggunakan kecepatan saat ini setelah beberapa interval waktu t.
2. Clearance, Fungsi $dist(v, w)$ menunjukkan jarak *obstacle* terdekat yang berpotongan dengan lintasan robot saat ini. Jika tidak terdapat perpotongan maka nilai fungi diatur ke konstanta besar.
3. Velocity, Fungsi $velocity(v, w)$ menghitung nilai kecepatan *linear* dalam *velocity set*.
4. Smoothing, Penggunaan normalisasi kepada tiga komponen fungsi objektif menjadi [0, 1] dimana menghitung koefisien $\alpha\ \beta\ \gamma$. Hal ini dapat membantu untuk menangani situasi dimana beberapa komponen terjadi bagian yang tidak tersambung yang membuat beberapa item mengambil peran utama.

2.3 Frontier-Based

Frontier-based exploration digunakan untuk dapat menelusuri wilayah secara menyeluruh dimana dalam melakukan penelusuran pada wilayah tidak dikenali membutuhkan *SLAM* dan *Navigation Stack* agar robot dapat berjalan secara otonom, hasil peta yang telah dibuat menggunakan *SLAM gmapping* digunakan *frontier-based exploration* untuk membuat *frontier point* dimana digunakan robot untuk bergerak menuju titik yang ada pada perbatasan wilayah [16]. Titik yang ada pada perbatasan akan difilter berdasarkan *cluster* dan menghilangkan *frontier point* yang tidak valid untuk mengurangi pemakaian komputasi. *Task allocator* memproses *point* yang telah difilter dan menugaskan robot untuk melakukan penelusuran berdasarkan titik yang telah difilter berdasarkan posisi robot (p_r) dan titik batas (p_f) [17]. Berikut persamaan untuk menentukan nilai *frontier point* sebagai berikut:

$$p_f = \{(x, y), x \in \mathbb{R}, y \in \mathbb{R}, \} \quad (2.8)$$

$$S_{info} = \pi r^2 \cap Area_{unknown} \quad (2.9)$$

$$d_E = \| p_r - p_f \|_2 \quad (2.10)$$

$$score(p_r, p_f) = c_1\, S_{info} - c_2\, d_e \text{ where } c_1, c_2 \in \mathbb{R} \quad (2.11)$$

Persamaan 2.8 hingga 2.10 merupakan persamaan yang akan digunakan pada persamaan 2.11. Persamaan 2.8 merupakan titik batas yang ditemukan pada peta. Persamaan 2.9 merupakan *information gain* (S_{info}) adalah jumlah informasi yang diperoleh dengan menghitung wilayah tidak dikenali dalam radius r dari *frontier point*. Persamaan 2.10 merupakan *euclidean distance* (d_E) untuk memperhitungkan nilai selisih antara posisi robot saat ini (p_r) dan titik batas (p_f) pada persamaan 2.8. Hasil persamaan yang telah dilakukan pada persamaan 2.8 hingga 2.10 digunakan kembali untuk menentukan nilai *frontier point* pada persamaan 2.11 namun robot perlu mempertimbangkan faktor lain yang berada di sekitar wilayah robot dan hal tersebut merupakan hal yang sulit untuk robot dapat bergerak lurus menuju *frontier point*. Hal itu dapat diatasi dengan menggunakan konstanta c_2 untuk merepresentasikan *cost* yang ada pada *mobile robot* untuk menavigasikan diri menuju *frontier point*. Untuk mengurangi eksplorasi secara berulang di dua wilayah dengan mengunakan konstanta c_1 dimana informasi jangkauan pada *frontier point* akan diperbesar dan memungkinkan robot untuk menjelajahi wilayah terdekat. Berikut merupakan persmaan dari c_1:

$$c_1 = \begin{cases} 1 & if \quad d_E > d_r \\ 2 & if \quad d_E \leq d_r \\ 3 & if \quad \| p_m - p_f \|_2 \leq d_r \end{cases} \quad (2.12)$$

Persamaan 2.12 menjelaskan pemilihan jangkauan frontier point berdasarkan nilai *euclidean distance* (d_E) dengan radius jangkauan *footprint* robot serta p_m merupakan *frontier point* untuk robot dapat bernavigasi sehingga perolehan *information gain* pada *frontier point* terdekat pada *task point* meningkat. Wilayah yang telah sepenuhnya dieksplorasi maka robot akan bergerak ke wilayah yang belum terjelajahi sehingga memungkinan robot untuk mengurangi penelusuran secara berulang.

2.4 Motion Planning

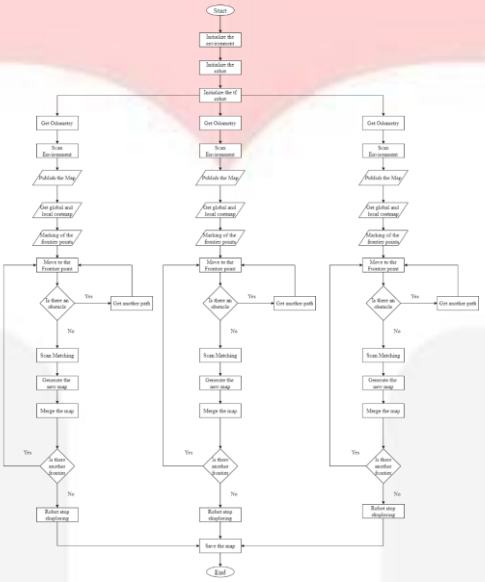
Penggunaan *motion planning* memungkinkan robot untuk menjalankan tugas tanpa saling bertabrakan satu sama lain, dalam *motion planning* terdapat dua kategori yakni *Path Planning dan Trajectory planning*. *Path planning* digunakan untuk merencanakan jalur robot dimana robot dapat melakukan perpindahan posisi sepanjang jalur kontinu yang telah dibuat. Jalur yang telah dibuat oleh path planning kemudian akan diikuti *trajectory planning* yang dimana dalam

trajectory planning terdapat *trajectory tracking* untuk melacak lintasan dengan terus mengikuti lintasan yang telah ditentukan oleh *path planning* [18][19].

1. Global planner, *Global planner* menghitung perencanaan dari pengaturan awal (x_{init}) ke pengaturan akhir robot (x_{end}) yang diinginkan. Perencanaan tersebut direpresentasikan sebagai urutan tujuan *local* $\{g_i \mid 1 \leq i \leq n\}$, x_{init} (g_1), dan x_{end} (g_n). Setiap tujuan dapat dicapai dari tujuan sebelumnya dengan mempertimbangkan masalah seperti menghindari *obstacle*.
2. Local planner, Tujuan yang sudah di buat *global planner* dan informasi pembacaan sensor, lokalisasi, dan odometri digunakan *local planner* untuk mengetahui posisi robot dimana kecepatan robot diatur oleh *controller* menggunakan algoritma pembatasan kecepatan robot seperti *DWA* [15].
3. Controller, Kecepatan yang digunakan pada *local planner* diatur oleh *controller* menggunakan *DWA* yang sudah diterapkan pada *local planner* dengan mengatur daya motor yang ada pada robot dan menjaga kecepatan tetap sama dengan *DWA*.

3. Perancangan

3.1 Gambaran Umum Frontier Exploration Pada Multi-Robot



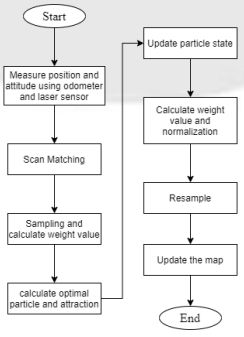
Gambar 3.1 Penelusuran Berbasis Frontier Menggunakan Multi-Robot

Gambar 3.1 menjelaskan cara kerja eksplorasi berbasis *frontier* pada multi robot. Sistem dimulai dengan menginisialisasi lingkungan simulasi, robot, serta tf yang berada dalam simulasi gazebo. Saat simulasi gazebo dijalankan, gazebo akan membuat robot dan simulasi virtual dimana pada setiap robot sudah ditentukan tf (*transform*). Penggunaan odometri pada robot dilakukan untuk mengetahui letak posisi pada peta yang akan dibangun serta mengetahui posisi pada simulasi gazebo, pada saat yang bersamaan robot melakukan pemindaian wilayah pada sekitar robot menggunakan *base laser* yang terpasang sehingga *gmapping* dapat diterapkan untuk dapat melakukan eksplorasi, *marker* yang diterapkan bertugas untuk menavigasikan robot untuk bergerak menuju *frontier point* diiringi dengan melakukan pengecekan *obstacle* yang ada pada sekitar robot.

3.2 Perancangan Frontier Exploration Pada Multi-Robot

Penelusuran berbasis *frontier* menggunakan multi robot dibutuhkan *SLAM gmapping*, *navigation stack*, serta *map merging* [6][12]. *SLAM gmapping* berfungsi untuk melakukan lokalisasi dan pemetaan secara simultan yang dibutuhkan dalam penelusuran berbasis frontier pada wilayah tidak dikenali. *Navigation stack* dibutuhkan pada eksplorasi berbasis *frontier* sebagai acuan gerak setiap robot pada wilayah simulasi yang sedang berlangsung. *Map merging* dibutuhkan untuk menyatukan peta yang telah dihasilkan dari setiap robot.

3.2.1 Gmapping



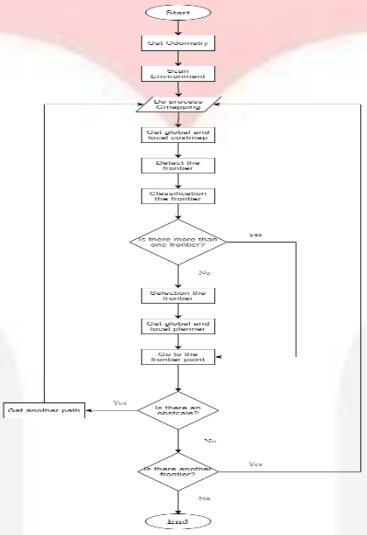
Gambar 3.2 Prosedur Gmapping Dalam Melakukan Pemetaan

Gambar 3.2 merupakan prosedur dari *gmapping* yang sedang berjalan berdasarkan cara kerja *RBPF* dimana langkah awal ialah mengukur wilayah sekitar serta mengetahui posisi menggunakan base scan dan odometri yang ada pada robot, selanjutnya robot akan melakukan scan matching untuk mengetahui hasil pemindaian laser serta posisi robot dengan memperhitungkan partikel sebelumnya. Keadaan partikel pada map sebelumnya kemudian akan dikalkulasikan ulang dengan partikel pada saat ini dan sebelumnya untuk mempersatukan partikel yang kemudian di tampilkan ke dalam map.

3.2.2 Navigation Stack

Sensor, odometri, dan *transform* dibutuhkan dalam *navigation stack* untuk mengetahui wilayah yang dipindai serta mengetahui posisi robot yang ada pada peta dimana *sensor source* merupakan *base_laser* yang terpasang pada setiap robot, odometri dan *transform* dibutuhkan untuk mengetahui posisi pergerakan robot dalam melakukan navigasi yang dituju akan tetapi untuk robot dapat bergerak menuju lokasi yang dituju dibutuhkan *move_base* sebagai acuan robot untuk bergerak yang terdapat *global* dan *local planner* serta *global* dan *local costmap* di dalam *move_base*. *Global planner* menggunakan peta serta radius sensor laser untuk menentukan perencanaan gerak dari titik A ke titik B, *local planner* menggunakan *motion model* untuk menetukan perencaan jalur terbaik yang memenuhi *global planer*. *Costmap* digunakan untuk menghindari obstacle untuk memastikan perencanaan menghindari sudut-sudut di sekitar *obstacle* pada *global costmap* yang digunakan untuk *global planner* dan *local costmap* digunakan untuk *local planner*.

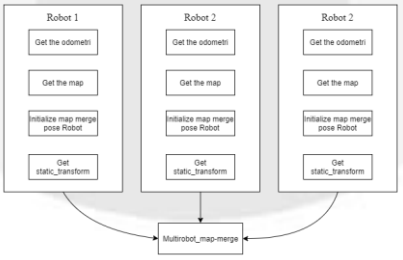
3.2.3 Frontier Exploration



Gambar 3.4 Cara Kerja Frontier-Based Exploration

Gambar 3.4 menjelaskan cara kerja *frontier exploration* pada *environment simulations* yang merupakan tahap awal mengetahui posisi robot menggunakan odometri. Penggunaan odometri dan *laser base_scan* digunakan untuk *scan matching* untuk mengkalkulasi partikel pada peta sebelumnya dan saat ini dicocokkan dan menghasilkan peta baru yang terdapat pada *SLAM gmapping*, partikel yang didapat kemudian ditentukan untuk menentukan *costmap* untuk memperkirakan *obstacle* pada area di sekitar robot sehingga dapat dilakukan navigasi [20]. *SLAM gmapping* dan *costmap* yang telah dilakukan kemudian menandai *frontier* dari hasil pemindaian menggunakan *marker* untuk dapat membedakan wilayah yang sudah dan belum belum dijelajahi. Jika robot mendeteksi terdapat lebih dari satu *frontier*, maka robot akan memilah jarak selisih yang lebih kecil antara *frontier* dengan robot [21]. Eksplorasi dilakukan apabila terdapat *frontier point* untuk dapat dijelajahi dimana membutuhkan *costmap* dan *planner* untuk dapat bergerak serta menghindari *obstacle* disekitar robot, jika pada peta tidak terdapat *frontier*, maka penelusuran berbasisis *frontier* telah selesai dilakukan dimana langkah selanjutnya menggabungkan peta yang telah dibuat oleh setiap robot menggunakan *map merge* [16].

3.2.2 Map Merge



Gambar 3.5 Cara Kerja Map Merging

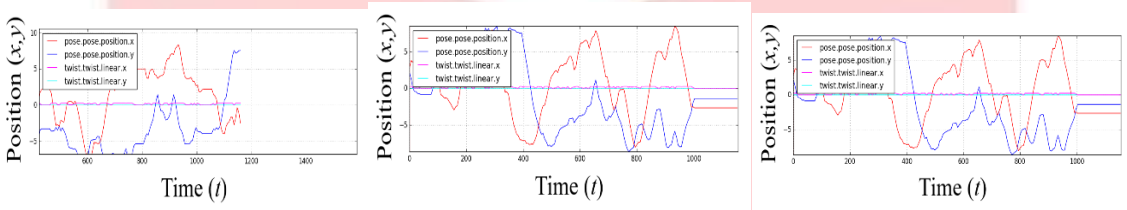
Gambar 3.5 menjelaskan penggabungan peta yang telah dilakukan oleh setiap robot menjadi satu keseluruhan peta, odometri dan hasil pemetaan setiap robot digunakan untuk mengetahui posisi robot pada peta yang yang dihasilkan yang kemudian digunakan untuk inisialisasi posisi pada peta keseluruhan. Penggunaan *static_transform* pada *map merging* dibutuhkan untuk menyatukan tf robot pada peta keseluruhan, jika hal ini tidak dilakukan maka pembuatan prosess peta keseluruhan akan terjadi *error* [12].

4. Pengujian

Tahap implementasi simulasi pengujian dilakukan dengan mengikuti alur proses perancangan pada bab sebelumnya, dengan menggunakan parameter yang dibutuhkan pada *frontier-based exploration* menggunakan multi robot maka implementasi dapat dilakukan. Berikut hasil simulasi *frontier-based exploration* pada multi robot dan single robot:

4.1 Hasil Pengujian

4.1.1 Pengujian Pada Multi Robot



(a) *First Robot* (b) *Second* Robot (c) *Third* Robot

Gambar 4.2 Odometri robot pertama (a), dua (b), dan tiga (c).

Gambar 4.11 merupakan pergerakan multi robot pada saat simulasi berlangsung. Sumbu vertikal menunjukkan posisi pergerakan robot menuju lokasi yang dituju, sedangkan horizontal menunjukkan lama waktu robot dalam melakukan pergerakan selama simulasi berlangsung. *Pose position* x dan y menunjukkan posisi gerak robot, *twist linear* x dan y menunjukkan rotasi gerak robot. Berikut merupakan *processing time* dan hasil pemetaan multi robot:

Tabel 4.1 *Processing Time* Multi Robot Pada *First* Robot

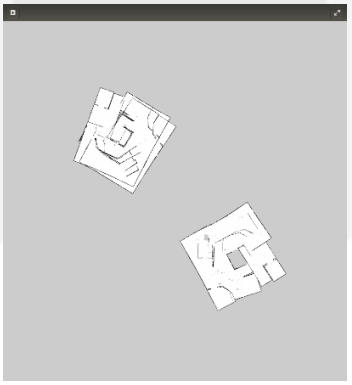
Processing Time	Position.x (*m*)	Position.y (*m*)	Twist.x (θ)	Twist.y (θ)
0 m 0 sec	5,54	8,16	0,25	0
5 m 0 sec	-7,96	7,53	0,26	0
10 m 0 sec	-7,61	-7,44	0,23	0
15 m 0 sec	6,40	-1,67	0	0
19 m 20 sec	-2,26	7,53	0,23	0

Tabel 4.2 *Processing Time* Multi Robot Pada *Second* Robot

Processing Time	Position.x (*m*)	Position.y (*m*)	Twist.x (θ)	Twist.y (θ)
0 m 0 sec	6,30	-7,27	0,26	0
5 m 0 sec	6,31	-1,10	0,26	0
10 m 0 sec	-5,27	-5,01	0,22	0
15 m 0 sec	-2,75	6,91	0,28	0
19 m 20 sec	-2,75	6,92	0	0

Tabel 4.3 *Processing Time* Multi Robot Pada *Second* Robot

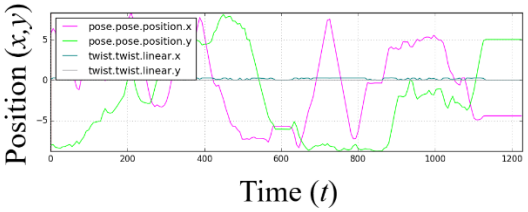
Processing Time	Position.x (*m*)	Position.y (*m*)	Twist.x (θ)	Twist.y (θ)
0 m 0 sec	6,30	-7,27	0,26	0
5 m 0 sec	6,31	-1,10	0,26	0
10 m 0 sec	-5,27	-5,01	0,22	0
15 m 0 sec	-2,75	6,91	0,28	0
19 m 20 sec	-2,75	6,92	0	0



Gambar 4.12 Hasil Pemetaan Multi Robot.

Gambar 4.12 merupakan hasil pemetaan yang telah dilakukan menggunakan multi robot dimana hasil pemetaan membutuhkan waktu selama 19 menit 30 detik dalam menghasilkan peta, hasil peta yang telah dibuat oleh setiap robot kemudian disatukan dan membentuk satu peta keseluruhan, meskipun hasil dari penggabungan peta tidak seperti yang diharapkan yakni terdapat peta yang terpisah dan terdapat penumpukkan peta.

4.1.1 Pengujian Pada Single Robot



Gambar 4.14 Odometri Pada *Single* Robot

Gambar 4.14 merupakan pergerakan *single* robot pada saat simulasi berlangsung. Sumbu vertikal menunjukkan posisi pergerakan robot menuju lokasi yang dituju, sedangkan horizontal menunjukkan lama waktu robot dalam melakukan pergerakan selama simulasi berlangsung. *Pose position* x dan y menunjukkan posisi gerak robot, *twist linear* x dan y menunjukkan rotasi gerak robot. Berikut merupakan *processing time* dan hasil pemetaan *single* robot:

Tabel 4.4 *Processing Time* Multi Robot Pada *Second* Robot

Processing Time	Position.x (m)	Position.y (m)	Twist.x (θ)	Twist.y (θ)
0 m 0 sec	5,74	-8,06	0,11	0
5 m 0 sec	-2,72	4,34	0,10	0
10 m 0 sec	-5,71	6,06	-0,01	0
15 m 0 sec	3,98	-2,91	0,26	0
19 m 20 sec	-4,43	5,02	0	0

Gambar 4.15 Hasil Pemetaan *Single* Robot

Gambar 4.15 merupakan hasil pemetaan yang dilakukan menggunakan *single* robot dalam *environment simulation* yang sama dengan multi robot dimana hasil pemetaan membutuhkan waktu selama 20 menit 20 detik dalam menghasilkan peta, dari peta yang telah dihasilkan terdapat lokasi yang belum tertelusuri hal ini disebabkan pertama karena terlalu banyaknya *frontier* yang harus dijelajahi, kedua pemilihan frontier dipilih berdasarkan nilai *frontier cost* tertinggi sehingga robot bergerak menuju nilai *cost* tertinggi, dan ketiga apabila jarak robot menuju *frontier* terlalu jauh maka robot akan mengabaikan frontier tersebut.

5. Kesimpulan dan Saran

5.1 Kesimpulan

1. Penggunaan *package explore_lite* dalam melakukan penelusuran berbasis *frontier* membutuhkan hasil pemindaian yang telah dilakukan *SLAM gmapping*. Hasil pemetaan digunakan untuk membuat *frontier point* menggunakan *marker msgs*, pemilihan *frontier point* ditentukan oleh nilai *frontier cost* tertinggi dari hasil selisih *costmap* yang ada pada *occupancy grid*.
2. Hasil nilai *frontier cost* dibatasi menggunakan parameter *min_frontier_size* untuk menentukan nilai minimum yang digunakan robot menanggapi pemilihan *frontier point* yang dituju, jika penelusuran membutuhkan waktu yang lama parameter *progress_timeout* digunakan untuk memberhentikan robot dalam melakukan penelusuran ke *frontier point* yang dituju dan menugaskan robot untuk menelusuri *frontier point* lainnya hingga tidak terdapat frontier yang belum terjelajahi.
3. Penggunaa *package map merge* digunakan untuk menyatukan hasil pemetaan dan tf setiap robot, inisialisasi posisi robot dalam *map merge* digunakan untuk menentukan posisi robot pada peta keseluruhan. *Static_transform* digunakan untuk menyatukan tf setiap robot dengan menjadikan satu peta menjadi *parent tree* dan *child tree* merupakan peta yang ingin disatukan peta ke *parent tree*, meskipun pada hasil analisis penggabungan peta terdapat penumpukan dan peta yang terpisah hal ini dikarenakan *parent tree* yang digunakan merupakan */map* dimana semestinya peta salah satu robot dijadikan *parent tree* yang digunakan robot lain untuk melakukan penelusuran.

5.2 Saran

1. Pembuatan *environment simulation* dibuat dua wilayah berbeda untuk membandingkan wilayah yang luas dan terdapat banyak *obstacle* serta wilayah sempit dan tidak terdapat *obstacle* untuk dibandingkan *ground route* robot.
2. Mengingat masih terdapat kelemahan dalam komunikasi antar robot pada saat melakukan penelusuran, penulis menyarankan bahwa membagi area penelusuran robot dengan menggunakan *publish point* dengan membatasi area penelusuran secara merata pada setiap robot.

DAFTAR PUSTAKA

[1] B. Yamauchi, "Yamauchi-frontierExploration98," in *AGENTS '98 Proceedings of the second international conference on Autonomous agents Pages 47-53*, 1998, no. May.

[2] A. Topiwala, P. Inani, and A. Kathpal, "Frontier Based Exploration for Autonomous Robot," 2018, [Online]. Available: http://arxiv.org/abs/1806.03581.

[3] W. A. Syaqur *et al.*, "Mobile Robot Based Simultaneous Localization and Mapping in UniMAP's Unknown Environment," in *2018 International Conference on Computational Approach in Smart Systems Design and Applications, ICASSDA 2018*, 2018, pp. 1–5, doi: 10.1109/ICASSDA.2018.8477629.

[4] B. L. E. A. Balasuriya *et al.*, "Outdoor robot navigation using Gmapping based SLAM algorithm," in *2nd International Moratuwa Engineering Research Conference, MERCon 2016*, 2016, pp. 403–408, doi: 10.1109/MERCon.2016.7480175.

[5] S. H. Chan, P. T. Wu, and L. C. Fu, "Robust 2D Indoor Localization Through Laser SLAM and Visual SLAM Fusion," in *Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018*, 2019, pp. 1263–1268, doi: 10.1109/SMC.2018.00221.

[6] E. Uslu, F. Cakmak, M. Balcilar, A. Akinci, M. F. Amasyali, and S. Yavuz, "Implementation of frontier-based exploration algorithm for an autonomous robot," in *INISTA 2015 - 2015 International Symposium on Innovations in Intelligent SysTems and Applications, Proceedings*, 2015, no. 2, doi: 10.1109/INISTA.2015.7276723.

[7] K. Krinkin, A. Filatov, A. Filatov, A. Huletski, and D. Kartashov, "The scan matchers research and comparison: Monte-Carlo, olson and hough," in *Conference of Open Innovation Association, FRUCT*, 2017, pp. 99–105, doi: 10.23919/FRUCT.2016.7892188.

[8] A. Li, X. Dai, D. Gong, and H. Ali, "Research on Rao-Blackwellized Particle Filter SLAM Based on Grey Wolf Optimizer," in *9th IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, CYBER 2019*, 2019, pp. 934–939, doi: 10.1109/CYBER46603.2019.9066735.

[9] Q. Xu, J. Zhao, C. Zhang, and F. He, "Design and implementation of an ROS based autonomous navigation system," in *2015 IEEE International Conference on Mechatronics and Automation, ICMA 2015*, 2015, pp. 2220–2225, doi: 10.1109/ICMA.2015.7237831.

[10] Z. Li and X. Mei, "Navigation and Control System of Mobile Robot Based on ROS," in *Proceedings of 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2018*, 2018, no. Iaeac, pp. 368–372, doi: 10.1109/IAEAC.2018.8577901.

[11] Y. Cheng and G. Y. Wang, "Mobile robot navigation based on lidar," in *Proceedings of the 30th Chinese Control and Decision Conference, CCDC 2018*, 2018, pp. 1243–1246, doi: 10.1109/CCDC.2018.8407319.

[12] S. Saeedi, L. Paull, M. Trentini, and H. Li, "Occupancy grid map merging for multiple robot simultaneous localization and mapping," *Int. J. Robot. Autom.*, vol. 30, no. 2, pp. 149–157, 2015, doi: 10.2316/Journal.206.2015.2.206-4028.

[13] M. S. Kashi, T. B. Sriram, and R. Mohan, "An Approach to Labelled Indoor Mapping using SLAM and Object Detection," in *Proceedings of 2019 IEEE Region 10 Symposium, TENSYMP 2019*, 2019, vol. 7, pp. 321–325, doi: 10.1109/TENSYMP46218.2019.8971298.

[14] Y. Abdelrasoul, A. B. S. H. Saman, and P. Sebastian, "A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM," in *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation, ROMA 2016*, 2017, doi: 10.1109/ROMA.2016.7847825.

[15] J. Wang, S. Wu, H. Li, and J. Zou, "Path planning combining improved rapidly-exploring random trees with dynamic window approach in ROS," in *Proceedings of the 13th IEEE Conference on Industrial Electronics and Applications, ICIEA 2018*, 2018, pp. 1296–1301, doi: 10.1109/ICIEA.2018.8397909.

[16] T. Tao, Y. Huang, F. Sun, and T. Wang, "Motion planning for SLAM based on frontier exploration," in *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation, ICMA 2007*, 2007, no. 60605021, pp. 2120–2125, doi: 10.1109/ICMA.2007.4303879.

[17] C. Y. Wu and H. Y. Lin, "Autonomous mobile robot exploration in unknown indoor environments based on rapidly-exploring random tree," in *Proceedings of the IEEE International Conference on Industrial Technology*, 2019, vol. 2019-Febru, pp. 1345–1350, doi: 10.1109/ICIT.2019.8754938.

[18] S. János and M. József, "Time-optimal motion planning for robots," in *International Journal of Mechanics and Control*, 2011, vol. 12, no. 1, pp. 61–74.

[19] I. Pérez-Hurtado, M. J. Pérez-Jiménez, G. Zhang, and D. Orellana-Martín, "Robot path planning using rapidly-exploring random trees: A membrane computing approach," in *2018 7th International Conference on Computers Communications and Control, ICCCC 2018 - Proceedings*, 2018, no. Icccc, pp. 37–46, doi: 10.1109/ICCCC.2018.8390434.

[20] A. K. Poernomo and H. S. Ying, "New cost function for multi-robot exploration," in *9th International Conference on Control, Automation, Robotics and Vision, 2006, ICARCV '06*, 2006, vol. 00, pp. 0–5, doi: 10.1109/ICARCV.2006.345220.

[21] J. Faigl and M. Kulich, "On determination of goal candidates in frontier-based multi-robot exploration," *2013 Eur. Conf. Mob. Robot. ECMR 2013 - Conf. Proc.*, pp. 210–215, 2013, doi: 10.1109/ECMR.2013.6698844.