BAB 3 PENANGANAN JARINGAN KOMUNIKASI MULTIHOP TERKONFIGURASI SENDIRI UNTUK PAIRFORM-COMMUNICATION

Bab ini akan menjelaskan tentang penanganan jaringan untuk komunikasi antara dua sumber yang berpasangan. Pada bagian awal bab dilakukan identifikasi permasalahan yang akan diselesaikan pada penelitian tugas akhir ini sesuai dengan rumusan masalah yang telah disusun sebelumnya. Selanjutnya, akan dijelaskan rancangan algoritma dan metode penyelesaian yang akan diterapkan untuk menemukan solusi permasalahan. Selain itu juga akan dipaparkan mengenai penelitian sejenis yang menjadi rujukan perancangan algoritma.

3.1 Definisi Permasalahan

Permasalahan gangguan dari luar sering menjadi kendala bagi pembangunan infrastruktur jaringan komunikasi yang handal. Berbagai kerusakan yang mungkin ditimbulkan oleh berbagai bentuk gangguan akan membutuhkan biaya yang besar dan waktu yang lama untuk dapat diperbaiki. Permasalahan serupa juga dialami ketika perlu dilakukan penambahan komponen baru ke dalam jaringan komunikasi yang sudah ada dalam rangka peningkatan kualitas. Perlu perubahan konfigurasi jaringan yang akan memakan biaya dan waktu pula, bahkan sampai dapat mengakibatkan penggantian jaringan lama menjadi jaringan yang baru sama sekali. Hal ini tentu saja akan mengganggu fungsi dari jaringan itu sendiri sebagai sarana pemenuhan kebutuhan komunikasi.

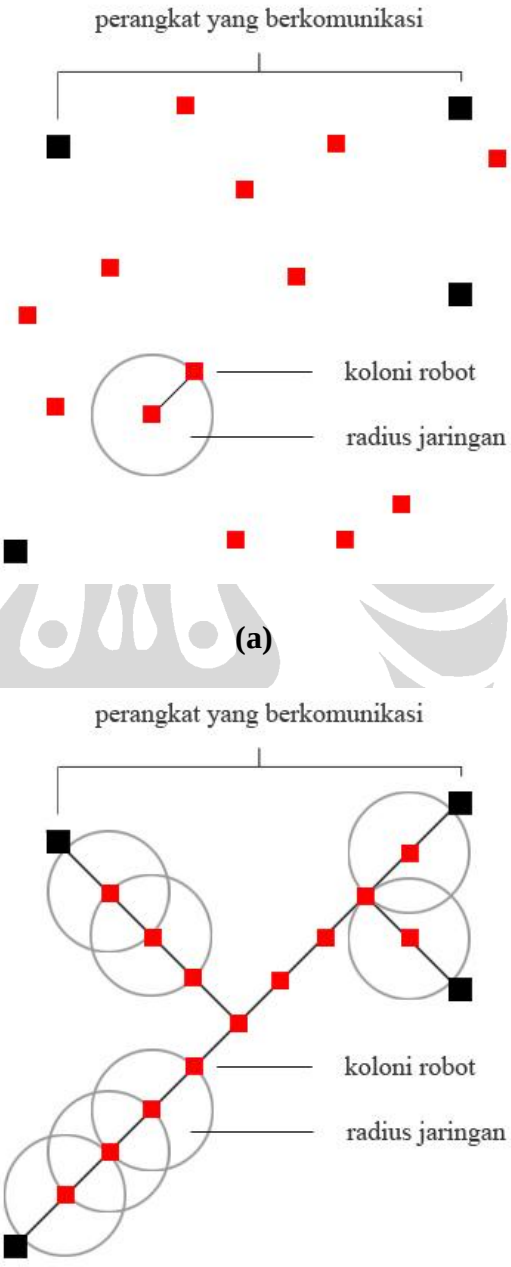
Penelitian tugas akhir ini dimaksudkan untuk memberikan prototipe solusi terhadap permasalahan yang disebutkan di atas. Melalui penelitian ini, penulis berupaya merancang suatu jaringan komunikasi yang bersifat adaptif, yaitu jaringan yang dapat menangani perubahan yang terjadi pada konfigurasinya, baik disebabkan oleh kerusakan komponen maupun komponen baru yang ditambahkan ke dalam jaringan. Ide utamanya adalah untuk mengembangkan sifat dapat melakukan konfigurasi terhadap diri sendiri oleh jaringan yang dirancang. Hal ini akan diimplementasikan dengan menggunakan suatu sistem cerdas yang akan melakukan eksplorasi dalam rangka menemukan konfigurasi jaringan yang sesuai

dengan yang diharapkan. Sistem cerdas yang dimaksud adalah suatu koloni robot bergerak yang dilengkapi oleh perangkat yang memadai untuk saling bekerja sama dalam membangun konfigurasi jaringan komunikasi yang dimaksud.

Dalam perkembangan teknologi robotika sendiri, penggunaan koloni robot dalam kegiatan penjelajahan sudah cukup sering digunakan. Dengan semakin banyaknya dikembangkan algoritma yang mendukung perilaku kecerdasan kolektif, pemanfaatan koloni robot pada kegiatan seperti ini semakin menunjukkan hasil yang menjanjikan. Hal inilah yang kemudian menjadi salah satu alasan lain penggunaan koloni robot bergerak pada proses perancangan jaringan komunikasi adaptif dalam penelitian tugas akhir ini.

Adapun bentuk teknologi yang cocok digunakan untuk menerapkan jaringan komunikasi adaptif ini adalah jenis teknologi komunikasi nirkabel dengan bentuk jaringan *multihop*. Artinya untuk dapat saling berkomunikasi di dalam jaringan, seluruh perangkat tidak berhubungan secara langsung satu sama lain, melainkan dihubungkan melalui beberapa *router* yang menentukan jalur komunikasi sesuai dengan konfigurasi yang ditentukan. Teknologi sejenis ini sudah pernah dikenal sebelumnya, yaitu disebut dengan *wireless sensor network*, di mana router-router pada jaringan komunikasi merupakan komponen yang dilengkapi perangkat teknologi komunikasi nirkabel untuk dapat meneruskan informasi dari satu router ke router yang lain. Kemudian, dengan menggunakan robot bergerak untuk menggantikan fungsi router tersebut, kemampuan jaringan untuk menemukan konfigurasinya sendiri dapat diperoleh melalui kegiatan eksplorasi robot dalam suatu ruang kerja yang diberikan batasan tertentu. Adapun jenis teknologi yang digunakan tidak menggunakan media kabel sebagai sarana pemindahan informasi, memungkinkan robot untuk dapat bergerak bebas dalam melakukan penjelajahan. Hal yang kemudian harus diselesaikan adalah bagaimana cara membangun metode bagi robot-robot ini untuk saling berkomunikasi agar dapat menemukan konfigurasi jaringan yang sesuai. Solusi yang ditemukan diharapkan merupakan yang paling optimum guna menekan biaya dan sumber daya yang dibutuhkan untuk membangun aplikasi ini pada kehidupan nyata.

Teknologi yang menggunakan koloni robot sebagai media penyusun jaringan ini kemudian disebut sebagai teknologi jaringan robot bergerak atau *mobile robotic network*. Untuk dapat lebih memahami bentuk teknologi yang sudah dijelaskan tersebut di atas, dapat dilihat melalui penjelasan secara visual pada gambar 3.1 berikut di bawah ini.



Gambar 3.1 - Jaringan Robot Bergerak

Berdasarkan gambar tersebut dapat dilihat bahwa kondisi awal permasalahan yang diberikan seperti pada gambar 3.1.a. Koloni robot diharapkan dapat menemukan solusi permasalahan seperti pada gambar 3.1.b. Untuk dapat menemukan konfigurasi jaringan komunikasi dari posisi awal yang saling tersebar dan acak, koloni robot harus dapat saling berkoordinasi hingga pada akhirnya masing-masing robot dapat menemukan posisi yang sesuai. Ketika semua robot sudah terhubung melalui suatu jaringan robot, maka perangkat atau sumber yang akan dihubungkan melalui jaringan dapat saling berkomunikasi. Pada gambar 3.1.b dapat dilihat bahwa robot berfungsi sebagai router pada saat proses komunikasi antar sumber berlangsung.

Dari gambar ini dapat dilihat, jika terjadi kerusakan pada salah satu robot, maka jaringan komunikasi akan terputus. Hal ini akan berarti kondisinya akan menjadi lebih kurang sama seperti gambar 3.1.a di mana belum ditemukan konfigurasi jaringan yang menghubungkan sumber-sumber komunikasi. Robot akan kembali melakukan ekplorasi untuk menemukan solusi yang baru.

Dari penjelasan ini, dapat dilihat bahwa sifat konfigurasi diri sendiri dapat memberikan sifat adaptif kepada jaringan sesuai dengan yang diharapkan. Oleh karena itu, pada penelitian ini akan difokuskan pada perancangan metode untuk memberikan kemampuan dapat melakukan proses konfigurasi sendiri terhadap jaringan komunikasi yang dibangun. Untuk alasan penyederhanaan, pada bagian ini akan dibahas dulu metode untuk membangun konfigurasi jaringan komunikasi *multihop* dengan kasus dua perangkat atau sumber komunikasi yang dihubungkan atau disebut sebagai *pairform-communication.*

3.2 Rancangan Penelitian dan Simulasi

Untuk menjelaskan dan menguji algoritma yang dikembangkan pada penelitian tugas akhir ini, maka dirancang suatu simulasi untuk menunjukkan hasil yang diperoleh dari penerapan algoritma tersebut pada permasalahan yang diberikan. Simulasi ini dirancang sedekat-dekatnya dengan kondisi sebenarnya di lapangan

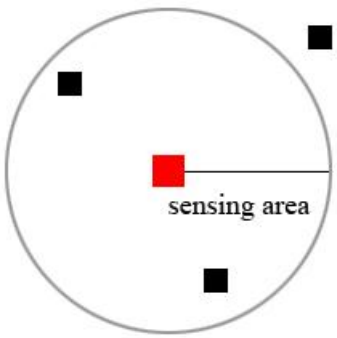
agar hasil yang ditemukan nantinya dapat diterapkan pada aplikasi dalam kehidupan nyata. Berikut adalah rancangan simulasi beserta beberapa asumsi yang digunakan pada penelitian ini.

1. Simulasi dilakukan pada suatu ruang kerja tertutup dan terbatas. Untuk alasan penyederhanaan, ruang kerja yang digunakan berupa segiempat dengan suatu ukuran panjang dan lebar tertentu. Selain itu diasumsikan juga bahwa ruang kerja ini juga bebas halangan dan rintangan.
2. Simulasi dirancang agar bersifat terdesentralisasi. Hal ini bertujuan untuk menghindari salah satu kerugian terbesar pada sistem yang bersifat terpusat, yaitu kegagalan sistem secara keseluruhan jika terjadi kerusakan pada komponen yang berperang sebagai pusat atau server.
3. Untuk mendukung simulasi yang bersifat terdesentralisasi, maka koloni robot yang digunakan pada simulasi ini adalah koloni robot otonom penuh, di mana setiap perintah dan informasi diproses langsung pada masing-masing robot yang bersangkutan.
4. Untuk alasan penyederhanaan, robot-robot otonom bekerja dalam sistem koordinat yang sama, yaitu suatu sistem koordinat absolut. Ditentukan suatu titik koordinat yang menjadi titik pusat (0, 0), yaitu batas kiri bawah dari ruang kerja simulasi serta titik maksimum (x_m, y_m), yaitu batas kanan atas dari ruang kerja simulasi. Dengan demikian posisi semua objek yang ada dalam simulasi dinyatakan dalam nilai axis-x dan ordinat-y positif dengan $P(x, y) = (0 <= x <= x_m, 0 <= y <= y_m)$.
5. Objek-objek yang terlibat dalam simulasi ini adalah sebagai berikut.
    a. Target, merupakan perangkat atau sumber yang akan dihubungkan ke dalam jaringan komunikasi *multihop* ini.
    b. Robot, merupakan perangkat yang akan melakukan penjelajahan dalam rangka menemukan konfigurasi jaringan yang diinginkan.

3.3 Spesifikasi Robot

Seperti yang sudah dijelaskan sebelumnya, komponen utama yang berperan penting dalam proses perancangan jaringan komunikasi *multihop* ini adalah koloni robot yang dilengkapi dengan perangkat yang memadai untuk membentuk suatu jaringan komunikasi jenis nirkabel. Berikut adalah spesifikasi robot yang digunakan dalam penelitian tugas akhir ini. Robot-robot otonom mempunyai beberapa fungsi atau kemampuan utama sebagai berikut.

1. *Sensing*, adalah kemampuan robot untuk mengenali objek-objek yang ada di sekitarnya dalam jarak tertentu. Luas daerah dimana robot dapat mengenali objek lain tersebut dinamakan sebagai *sensing area*. Daerah ini diasumsikan berbentuk lingkaran dengan jarak radius tertentu dari robot itu sendiri. Kemampuan robot untuk mengenali objek lain diasumsikan ideal karena penelitian ini lebih difokuskan pada proses konfigurasi jaringan. Artinya, robot dijamin dapat mengenali semua objek, baik target ataupun robot lain yang berada dalam *sensing area*. Gambar 3.2 berikut di bawah ini merupakan representasi grafik dari *sensing area*.



Gambar 3.2 - Sensing Area

2. *Communication*, adalah kemampuan robot untuk berkomunikasi dengan perangkat lain, baik target maupun robot lain, yang ada dalam simulasi. Untuk dapat membangun hubungan komunikasi dengan objek lain, robot harus mengenali objek yang akan dihubunginya tersebut terlebih dahulu. Ada batasan jarak tertentu yang dapat dijangkau oleh fungsi ini karena

fungsi *communication* ini didukung oleh teknologi komunikasi jenis nirkabel. Oleh karena itu, ditentukan bahwa robot hanya dapat berkomunikasi dengan objek-objek yang berada dalam *sensing area*-nya. Untuk alasan penyederhanaan, proses pertukaran dan pemindahan informasi selama berkomunikasi diasumsikan ideal. Artinya informasi yang dipindahkan dari suatu robot ke robot lain dijamin bebas error.

3. *Computation*, adalah kemampuan robot untuk mengolah data yang diberikan untuk kemudian menggunakan hasil yang diperoleh darinya sesuai dengan kebutuhan, misalnya robot dapat menentukan jarak antara dirinya dengan objek lain untuk mencegah terjadi tabrakan.
4. *Locomotion*, adalah kemampuan robot untuk bergerak atau berpindah dari suatu tempat ke tempat yang lain untuk mencapai tujuannya. Untuk alasan penyederhanaan, gerakan robot diasumsikan sempurna tanpa ada faktor fisik yang dapat mengganggu pergerakan. Robot diasumsikan dapat bergerak ke segala arah tanpa harus berotasi terlebih dahulu karena mekanisme pergerakan robot juga bukan merupakan fokus dari penelitian.

3.4 Pergerakan Robot

Salah satu fungsi utama yang dimiliki oleh robot adalah fungsi bergerak atau *locomotion*. Masalah pergerakan robot merupakan salah satu hal yang penting dibahas sebelum mendefinisikan cara kerja dari fungsi-fungsi yang lain karena pergerakan robot dapat menentukan algoritma penyelesaian masalah yang dapat digunakan pada penelitian. Berikut akan dijelaskan algoritma yang digunakan untuk menentukan pergerakan robot.

Robot bergerak berdasarkan informasi posisi robot saat ini serta informasi posisi tujuan yang akan dicapainya. Selanjutnya kedua informasi ini akan disebut sebagai “posisi awal” dan “posisi tujuan”.

Untuk dapat mencapai posisi tujuan tersebut, robot bergerak dalam langkah yang menunjukkan proses perpindahan robot dari posisi awal hingga posisi tujuan. Oleh

karena itu, jumlah langkah yang diperlukan oleh robot ini dapat juga dilihat sebagai jumlah iterasi yang dibutuhkan robot sebelum mencapai posisi tujuan dari posisi awal. Hubungan ini kemudian dapat dilihat sebagai berikut.

Total Jarak = Range Per Langkah * Langkah

Total Jarak = Range Per Langkah * Iterasi **(3.1)**

Untuk menentukan pergerakan robot, dapat dilakukan dengan memanfaatkan garis lurus yang menghubungkan kedua posisi awal dan posisi tujuan. Jika robot bergerak mengikuti koordinat yang melalui garis lurus ini, maka robot dapat mencapai posisi tujuan yang diinginkannya. Berikut adalah langkah komputasi yang diperlukan untuk menentukan pergerakan robot dengan memanfaatkan informasi posisi awal dan posisi akhir.

1. Tentukan sudut yang dibentuk garis dengan sumbu x positif, dengan menggunakan rumus sebagai berikut.

$$\tan \theta = (y_2 - y_1) / (x_2 - x_1)$$

$$\theta = \arctan ((y_2 - y_1) / (x_2 - x_1)) \quad \textbf{(3.2)}$$

$P_1 (x_1, y_1)$ = posisi awal.

$P_2 (x_2, y_2)$ = posisi tujuan.

Θ = sudut pergerakan terhadap sumbu x positif

2. Dalam setiap iterasi, robot dapat berpindah sejauh nilai range pada garis lurus. Berdasarkan nilai perpindahan tersebut dapat ditentukan pergerakan pada sumbu x dan sumbu y dengan menggunakan rumus.

$$\mathbf{P_1' = P_1 + \Delta P}$$

$$\mathbf{P_1' = (x_1 + \Delta x, y_1 + \Delta y)} \qquad \textbf{(3.3)}$$

$$\mathbf{P_1' = (x_1 + \Delta d * \cos \theta, y_1 + \Delta d * \sin \theta)}$$

P_1' (x_1', y_1') = posisi robot baru setelah pergerakan.

P_1 (x_1, y_1) = posisi lama robot sebelum pergerakan.

ΔP = jarak pergerakan dari posisi lama.

Δx = jarak pergerakan, pada sumbu x.

Δy = jarak pergerakan, pada sumbu y.

Δd = jarak pergerakan per langkah, yaitu range robot.

Θ = sudut pergerakan terhadap sumbu x positif

3. Robot berhenti ketika sudah mencapai posisi tujuannya. Hal ini dapat ditentukan dengan cara menghitung jarak antara posisi robot saat ini dengan posisi tujuan yang diharapkan. Untuk menentukan jarak, dapat dilakukan dengan menggunakan rumus jarak antara dua titik.

$$\mathbf{| P_2 - P_1 | = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \qquad \textbf{(3.4)}$$

P_2 (x_2, y_2) = posisi robot tujuan.

P_1 (x_1, y_1) = posisi lama robot sebelum pergerakan.

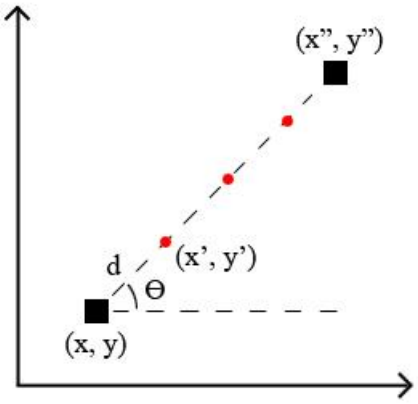
Untuk menentukan apakah robot sudah mencapai tujuan, dilakukan dengan cara membandingkan jarak robot – posisi tujuan sama dengan nol, yang berarti tidak terdapat jarak di antara keduanya.

$$\mathbf{| P_2 - P_1 | = 0} \qquad \textbf{(3.5)}$$

Akan tetapi, karena pergerakan robot pada setiap iterasi bersifat diskrit terhadap range-nya, sedangkan tidak ada jaminan bahwa jarak yang harus ditempuh merupakan kelipatan dari nilai range, maka digunakan nilai threshold.

$$| P_2 - P_1 | <= \textbf{Threshold} \qquad (3.6)$$

Gambar 3.3 berikut di bawah ini merupakan representasi grafik dari perumusan algoritma pergerakan seperti yang dijelaskan di atas. Robot bergerak dari posisi awal P(x, y) pada bagian kiri bawah menuju posisi tujuan P”(x”, y”) pada bagian kanan atas. Robot dapat mencapai tujuan dalam tiga kali iterasi seperti yang ditunjukkan titik-titik merah karena pergerakan robot terbatas oleh range d. Pada saat iterasi pertama, robot bergerak menuju posisi P’(x’, y’).



Gambar 3.3 - Pergerakan Robot

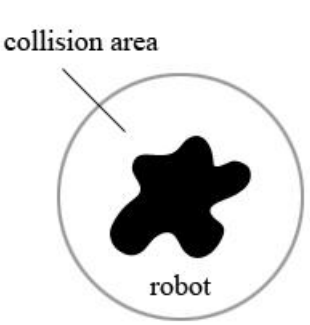
3.5 Pencegahan Benturan

Salah satu masalah yang harus diperhatikan dalam pergerakan robot adalah masalah benturan. Terdapat kemungkinan dimana robot akan bertabrakan dengan objek yang lain karena dalam simulasi robot merupakan objek yang bergerak dan bukan satu-satunya objek yang ada pada ruang kerja, dan karena robot. Jika hal ini

tidak ditangani dengan baik, maka hasil yang diperoleh melalui simulasi akan sulit untuk diterapkan dalam kehidupan yang sebenarnya. Pada penelitian ini tidak akan diperhitungkan apa yang akan terjadi setelah terjadinya benturan, tetapi akan dirancang suatu metode tertentu yang dapat mencegah terjadi benturan antar robot dengan objek lain.

Sebelum menentukan algoritma yang akan digunakan untuk mengatasi masalah benturan ini, perlu didefinisikan pada keadaan yang bagaimana benturan tersebut mungkin terjadi. Untuk itu perlu ditentukan masalah dimensi robot. Pada penelitian tugas akhir ini, tidak ditentukan bentuk robot yang akan digunakan pada implementasi yang sebenarnya karena desain fisik robot yang paling baik diterapkan untuk menyelesaikan masalah tidak menjadi fokus utama. Oleh karena itu, dimensi robot hanya didefinisikan sebagai bentuk tidak tentu.

Untuk mengetahui terjadi benturan, ditentukan luas daerah yang berjarak tertentu dari robot yang jika dimasuki oleh objek lain dianggap sebagai gejala akan terjadinya benturan. Pada keadaan seperti inilah kemudian akan dilakukan proses pergerakan untuk mencegah benturan. Untuk lebih jelasnya, definisi wilayah benturan atau disebut sebagai *collision area* ini dapat dillihat pada gambar 3.4 berikut di bawah ini.



Gambar 3.4 - Collision Area

Dari gambar 3.4 tersebut, dapat dilihat bahwa *collision area* robot dapat dirumuskan sebagai berikut.

Collision Area = Ukuran Robot + Threshold **(3.7)**

Adapun penggunaan *collision area* sebagai metode untuk mendeteksi terjadinya benturan dimaksudkan untuk alasan penyederhanaan penelitian yang dilakukan. Hal ini disebabkan karena akan cukup sulit untuk mengetahui terjadinya benturan jika dilihat dari tabrakan yang benar-benar terjadi dalam keadaan yang sebenarnya, yang berarti akan bergantung kepada desain, ukuran, dan bentuk fisik robot yang tidak dibahas pada penelitian tugas akhir ini. Penggunaan nilai threshold juga dimaksudkan sebagai ambang batas waktu sebelum terjadinya benturan yang sesungguhnya sehingga benturan robot dapat dicegah sebelum benar-benar terjadi. Oleh karena itu, pada penelitian ini akan dibahas metode mencegah terjadinya benturan, bukan untuk menangani efek benturan itu sendiri.

Berikut akan dijelaskan algoritma yang digunakan robot untuk mencegah terjadinya benturan antara robot dengan robot lain ataupun target.

1. Setelah menentukan pergerakan pada setiap iterasi, lakukan pengecekan apakah jika robot bergerak akan mungkin terjadi benturan atau tidak. Benturan mungkin terjadi jika ada objek lain yang akan masuk ke dalam *collision area* robot tersebut. Hal ini disebut sebagai gejala benturan.
2. Jika terjadi gejala benturan, maka lakukan beberapa alternatif pergerakan yang dapat menghilangkan gejala benturan tersebut, yaitu sebagai berikut.
    a. Lakukan pengecekan apakah dengan bergerak ke kanan masih akan terjadi gejala benturan atau tidak. Jika tidak, maka robot bergerak ke kanan pada iterasi berikutnya dan lanjutkan kepada langkah 3. Jika masih terjadi gejala benturan, lanjutkan kepada langkah 2.b.
    b. Lakukan pengecekan apakah dengan bergerak ke kiri masih akan terjadi gejala benturan atau tidak. Jika tidak, maka robot bergerak ke kiri pada iterasi berikutnya dan lanjutkan kepada langkah 3. Jika masih terjadi gejala benturan, lanjutkan kepada langkah 2.c.

c. Lakukan pengecekan apakah dengan bergerak ke belakang masih akan terjadi gejala benturan atau tidak. Jika tidak, maka robot bergerak ke belakang pada iterasi berikutnya dan lanjutkan kepada langkah 3. Jika masih terjadi gejala benturan, lanjutkan kepada langkah 2.d.

d. Lakukan pengecekan apakah dengan bergerak ke depan masih akan terjadi gejala benturan atau tidak. Jika tidak, maka robot bergerak ke depan pada iterasi berikutnya dan lanjutkan kepada langkah 3. Jika masih terjadi gejala benturan, lanjutkan kepada langkah 2.e.

e. Robot berhenti bergerak selama batas waktu yang acak. Jika masih terjadi gejala benturan saat robot akan kembali bergerak, lakukan kembali langkah 2.a dan seterusnya.

3. Robot dapat melanjutkan kembali algoritma pergerakan biasa jika sudah tidak terdapat lagi gejala benturan. Algoritma pencegahan benturan ini akan kembali dijalankan pada saat iterasi berikutnya dieksekusi.

3.6 Perilaku Robot Otonom dalam Simulasi

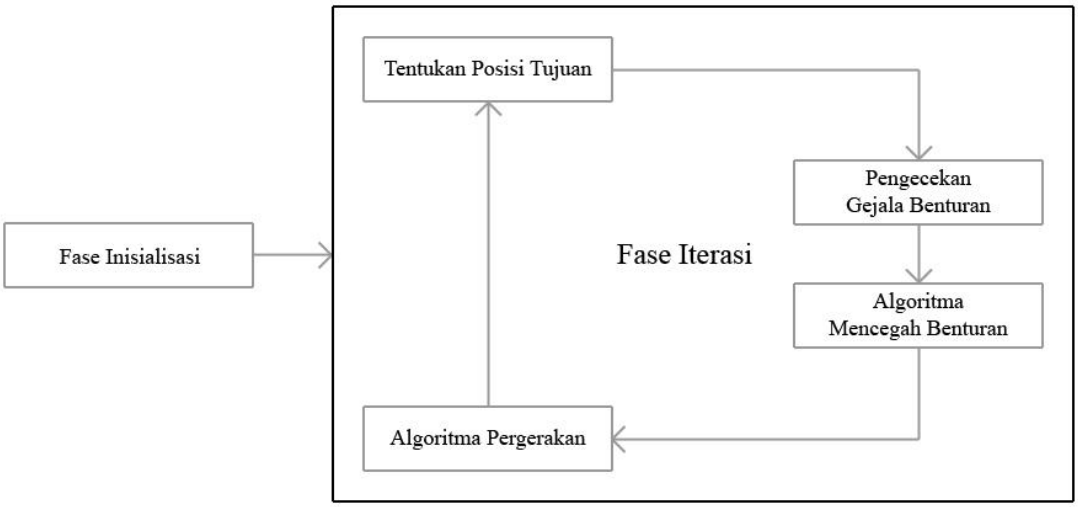
Pada bagian ini akan dijelaskan mengenai perilaku robot otonom dalam simulasi. Hal ini terkait dengan kemampuan robot untuk mengolah serta memanfaatkan data dan informasi dalam rangka proses pengambilan keputusan yang merupakan salah satu faktor paling penting dalam suatu sistem cerdas.

Dalam simulasi ini, terdapat dua tahapan utama yang dikerjakan oleh masing-masing robot otonom, yaitu sebagai berikut.

1. Fase Inisialisasi, yaitu tahapan di mana robot diberikan informasi-informasi penting yang akan dibutuhkannya selama menjalankan tugas selama proses simulasi. Informasi ini seperti karakteristik robot, karakteristik target, posisi awal robot tersebut, posisi target-target yang akan dihubungkan di dalam konfigurasi jaringan komunikasi, jumlah robot yang ada dalam simulasi dan sebagainya.

2. Fase Iterasi, yaitu tahapan di mana robot melakukan kegiatan eksplorasi untuk dapat menemukan konfigurasi jaringan komunikasi yang sesuai dengan yang diharapkan. Pada tahapan ini, robot melakukan komputasi untuk melakukan pergerakan dalam rangka mencapai tujuan akhirnya berdasarkan algoritma yang sudah dijelaskan sebelumnya.

Gambar 3.5 berikut di bawah merupakan diagram yang merepresentasikan kedua tahapan di atas yang menentukan perilaku robot dalam simulasi.



Gambar 3.5 - Perilaku Robot Otonom dalam Simulasi

Berdasarkan diagram tersebut, dapat dilihat bahwa sekalipun pergerakan robot sudah didefinisikan melalui algoritma pergerakan dan algoritma pencegahan benturan, tetapi tahapan yang paling berperan penting bagi robot dalam simulasi untuk dapat menemukan konfigurasi jaringan adalah bagaimana cara menentukan posisi yang akan dituju pada setiap iterasinya. Oleh karena itu, perlu dipikirkan algoritma untuk menentukan posisi tujuan pergerakan robot otonom.

3.7 Algoritma Chained Shortest Distance

Pada kegiatan eksplorasi dalam simulasi, robot otonom diharapkan dapat menemukan berbagai posisi yang secara bersama-sama akan membentuk suatu jaringan sensor yang merupakan solusi bagi konfigurasi jaringan komunikasi *multihop* yang diharapkan. Posisi-posisi inilah yang kemudian menjadi tujuan pergerakan masing-masing robot, di mana harus dipikirkan suatu metode untuk menentukan koordinasi koloni pergerakan robot agar tidak terjadi konflik dan dapat ditemukan solusi yang diinginkan.

Algoritma *Chained Shortest Distance* ini merupakan algoritma yang penulis kembangkan untuk menentukan koordinasi pergerakan robot dalam simulasi. Algoritma ini berdasarkan prinsip bahwa robot memiliki hubungan dengan target jika robot tersebut dapat berkomunikasi dengan target secara langsung, ataupun robot dapat berkomunikasi dengan robot lain yang dapat berkomunikasi dengan target. Dengan membentuk rantai hubungan komunikasi, maka konfigurasi jaringan komunikasi akan ditemukan pada saat seluruh target saling terhubung satu sama lain. Berikut akan dijelaskan bagaimana algoritma ini dapat menyelesaikan permasalahan.

1. Robot otonom memulai simulasi dengan posisi awal yang acak. Sedangkan posisi target-target harus ditentukan posisinya.
2. Robot menyimpan informasi yang dibutuhkannya untuk melakukan kegiatan eksplorasi, yaitu informasi mengenai posisi dirinya masing-masing beserta informasi mengenai posisi seluruh target. Akan tetapi, robot tidak mengetahui posisi robot-robot lainnya dalam simulasi. Informasi ini diberikan kepada robot otonom selama fase inisialisasi.
3. Target memiliki prioritas urutan, di mana saat simulasi dimulai robot otonom bergerak menuju target prioritas terendah sebagai posisi tujuan.
4. Robot otonom mengubah posisi tujuan kepada target berikutnya ketika ia sudah berhasil terhubung dengan target yang sedang ditujunya. Seperti yang dijelaskan sebelumnya, hubungan ini bisa berupa hubungan langsung antara

robot dengan target atau hubungan tidak langsung antara robot dengan target melalui robot-robot lain. Berikut adalah cara robot untuk melakukan pengecekan hubungan dengan target.

a. Selama bergerak robot melakukan pendeteksian apakah terdapat objek lain yang masuk ke dalam *sensing area*-nya. Jika iya, maka robot akan membangun hubungan dengan objek lain tersebut.

b. Jika objek yang berada dalam *sensing area* adalah target, maka robot akan langsung menentukan bahwa dirinya telah terhubung dengan target yang sedang ditujunya tersebut. Hal ini berarti robot terhubung secara langsung dengan target.

c. Jika objek yang berada dalam *sensing area* adalah robot lain, maka robot akan dapat mengakses informasi yang disimpan oleh robot lain tersebut melalui proses komunikasi. Robot kemudian dapat mengetahui apakah robot lain tersebut terhubung dengan target yang sedang ditujunya atau tidak. Jika iya, robot akan menarik kesimpulan bahwa sekarang ia juga sudah terhubung dengan target tersebut. Hal ini berarti robot terhubung secara tidak langsung dengan target.

d. Robot kemudian mengubah posisi tujuannya kepada target yang berikutnya. Jika semua target sudah pernah dituju, maka robot berhenti bergerak dan kegiatan penjelajahan selesai dilakukan.

5. Selain bergerak menuju posisi tujuan yang ditentukan, robot juga harus dapat mempertahankan hubungan yang sudah terbentuk agar tidak terputus. Artinya, ketika robot sudah terhubung dengan suatu target dan akan bergerak menuju target berikutnya, jika pergerakan tersebut dapat menyebabkan putusnya hubungan dengan target yang sudah terhubung dengannya, maka robot akan memprioritaskan untuk berhenti bergerak dalam rangka menjaga hubungan dengan target sebelumnya tersebut. Dengan menerapkan hal ini, pada saat robot-robot otonom berhasil mencapai seluruh target yang ada dalam simulasi, dapat dijamin bahwa

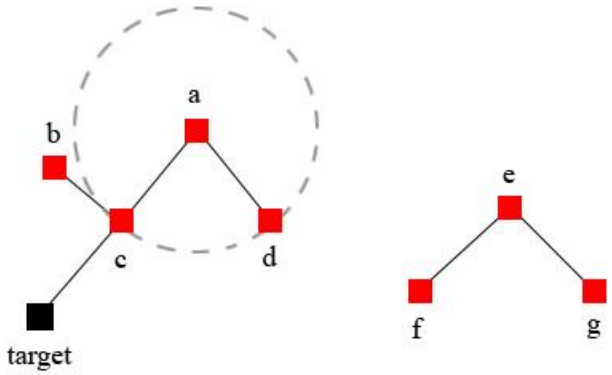
target-target telah terhubung dalam suatu konfigurasi jaringan komunikasi yang dicari.

6. Pada saat robot membangun hubungan dengan objek-objek di dalam *sensing area*-nya, terdapat kemungkinan bahwa robot akan terhubung dengan target melalui lebih dari satu kemungkinan hubungan. Hal ini akan menyebabkan terjadinya jalur hubungan yang redundan pada konfigurasi jaringan komunikasi yang sedang dibangun. Oleh karena itu, diperlukan suatu metode tertentu untuk memutuskan jalur hubungan yang redundan tersebut, yaitu dengan cara sebagai berikut.

   a. Pada saat robot membangun hubungan dengan robot lain dalam *sensing area*-nya, robot tidak hanya mendapatkan informasi hubungan robot lain tersebut dengan target, melainkan juga harus mendapatkan jarak hubungannya.

   b. Jika terdapat lebih dari satu kemungkinan hubungan antara robot dan target, maka robot memprioritaskan hubungan melalui robot yang jaraknya lebih pendek terhadap target. Robot memutuskan hubungan melalui robot yang jaraknya lebih jauh terhadap target.

Berdasarkan penjelasan di atas dapat dilihat bahwa robot mengetahui ada tidaknya hubungan dengan target melalui informasi yang diperolehnya dari objek lain yang terhubung secara langsung dengannya, sedangkan objek lain tersebut juga mendapatkan informasi serupa dengan cara yang sama. Hal ini berarti informasi mengenai hubungan tersebut diperoleh melalui proses penelusuran yang dimulai dari suatu robot hingga target. Jika pada proses penelusuran ini berakhir pada target berarti robot memiliki hubungan dengan target. Proses penelusuran informasi ini kemudian dimodelkan dalam suatu bentuk *searching tree*, di mana algoritma yang digunakan adalah algoritma DFS atau *Depth First Search*. Untuk lebih jelasnya dapat dilihat pada gambar 3.6 berikut di bawah ini.



Gambar 3.6 - Penelusuran Informasi pada Robot

Dari gambar 3.6 tersebut, robot a akan mendapatkan informasi bahwa dirinya terhubung dengan target melalui *searching tree* dengan menggunakan algoritma DFS. Pertama kali robot a akan menelusuri robot c sebagai salah satu robot yang terhubung secara langsung dengannya. Proses penelusuran dilanjutkan kepada robot b yang terhubung dengan robot c, karena robot c bukan merupakan target. Kemudian, karena robot b tidak memiliki hubungan dengan robot lainnya, akan dilakukan backtracking dan penelusuran dilanjutkan kepada objek lain yang terhubung dengan robot c. Objek yang ditemukan berikutnya adalah target. Robot c kemudian menyimpan informasi bahwa dirinya terhubung dengan target karena pada *searching tree* ini target berhasil ditemukan. Dari informasi yang disimpan ini, robot a kemudian mengetahui bahwa dirinya terhubung dengan target juga.

Dengan menerapkan proses penelusuran yang serupa, semua robot pun dapat menentukan apakah dirinya terhubung dengan target atau tidak. Pada contoh gambar 3.6 tersebut di atas, robot b, c, dan d akan mengetahui bahwa mereka terhubung dengan target. Sedangkan robot e, f, dan g akan mengetahui bahwa mereka tidak terhubung dengan target karena dalam proses penelusuran *searching tree* yang mereka lakukan tidak pernah ditemukan target.

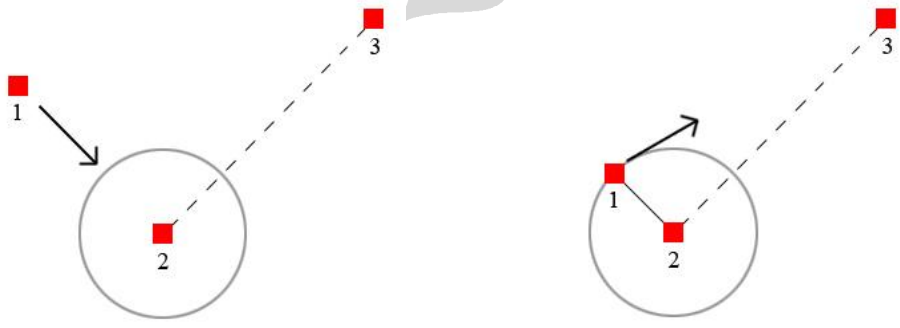
Algoritma ini diharapkan dapat memberikan solusi terhadap permasalahan yang akan diselesaikan. Akan tetapi, pada kenyataannya terdapat beberapa contoh kasus yang menyebabkan algoritma ini tidak efektif untuk digunakan. Hal tersebut yang kemudian menyebabkan perlunya dilakukan beberapa langkah perbaikan terhadap

algoritma *Chained Shortest Distance*. Berikut akan dijelaskan langkah perbaikan tersebut yang kemudian disebut sebagai algoritma *Optimized Chained Shortest.*

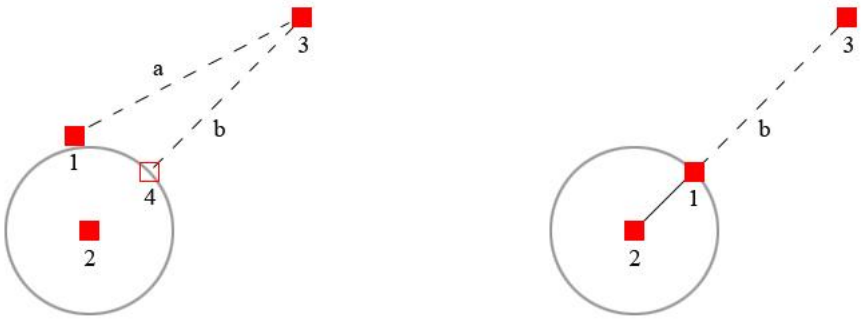
3.8 Algoritma Optimized Chained Shortest

Algoritma ini dikembangkan untuk mengatasi keterbatasan yang ditemukan pada algoritma *Chained Shortest Distance*. Sebelum dijelaskan mengenai algoritma ini, akan lebih dahulu dijabarkan tentang kasus yang menimbulkan keterbatasan ini.

Pada saat robot otonom pertama kali memasuki fase iterasi, robot akan bergerak mendekati salah satu target yang memiliki prioritas paling tinggi, hingga robot terhubung dengan target tersebut melalui hubungan secara langsung ataupun secara tidak langsung. Setelah itu, robot akan menentukan target berikutnya yang memiliki prioritas lebih rendah daripada target yang sudah terhubung sebelumnya dan robot kemudian bergerak mendekati target tersebut. Akan tetapi, kali ini dengan syarat bahwa robot tersebut harus tetap menjaga agar dirinya tetap terhubung dengan target yang sudah didekati sebelumnya. Robot akan berhenti jika ternyata pergerakannya menyebabkannya keluar dari rentang komunikasi tersebut, meskipun belum berhasil terhubung dengan target yang sedang berusaha didekatinya. Pada kenyataannya, robot bisa saja berhenti pada posisi yang sebenarnya belum mencapai posisi yang optimum. Hal ini dapat dilihat pada gambar 3.7 berikut di bawah ini.



(a) **(b)**



(c) **(d)**

Gambar 3.7 - Penerapan Algoritma Optimized Chained Shortest

Pada gambar 3.7.a robot 1 bergerak menuju posisi 2 yang diasumsikan merupakan target yang mempunyai prioritas paling tinggi, dari arah tegak lurus terhadap 3 yang diasumsikan merupakan target lain yang akan dituju setelahnya. Pada gambar 3.7.b, posisi 2 masuk ke dalam rentang *sensing area* robot 1 dan robot 1 mendapatkan informasi bahwa dirinya telah terhubung dengan target yang sedang dituju. Perlu diperhatikan bahwa posisi 2 juga dapat dipandang memiliki radius tertentu yang berlaku sama seperti *sensing*, yang dapat digunakan untuk menentukan hubungan suatu robot dengan posisi 2 tersebut. Pada gambar 3.7 ini, digunakan *sensing area* posisi 2 untuk melihat hubungan robot 1 dan posisi 2.

Robot 1 kemudian menentukan posisi 3 sebagai tujuan berikutnya setelah terhubung dengan posisi 2. Akan tetapi, robot 1 langsung berhenti ketika mencapai posisi pada gambar 3.7.c karena robot 1 menjadi tidak terhubung lagi dengan posisi 2 yang berarti putusnya hubungan dengan target sebelumnya.

Permasalahannya adalah posisi tersebut sebenarnya bukanlah posisi yang paling baik bagi robot 1 untuk berhenti. Pada kenyataannya, robot 1 akan lebih baik jika berhenti pada posisi 4 yang memiliki jarak yang lebih dekat terhadap posisi 3 serta masih terhubung dengan posisi 2. Oleh karena itu, diperlukan suatu metode untuk menemukan posisi 4 dari posisi robot 1 berhenti karena posisi 2 akan keluar dari rentang *sensing area*-nya. Berikut ini akan dijelaskan mengenai metode yang dinamakan sebagai algoritma *Optimized Chained Shortest* ini.

Asumsikan posisi 4 adalah posisi yang akan ditemukan ketika robot mencapai posisi 1 dan berhenti. Posisi 1 memiliki jarak *b* terhadap posisi 3 sehingga garis yang menghubungkan posisi 1 dan 3 kemudian disebut sebagai garis *b*. Posisi 4 memiliki jarak *a* terhadap posisi 3 sehingga garis yang menghubungkan posisi 3 dan 4 kemudian disebut sebagai garis *a*. Posisi 4 kemudian dapat dicari dengan cara menemukan titik potong antara garis *b* dengan lingkaran yang merepresentasikan *sensing area* robot 2.

Garis *a* merupakan garis singgung bagi lingkaran *sensing area* posisi 2, sedangkan menurut teori geometri mengatakan bahwa hanya terdapat 1 titik potong antara lingkaran dan suatu garis singgung. Hal ini akan menyebabkan robot 1 keluar dari rentang *sensing area* posisi 2 begitu meninggalkan posisinya tersebut. Oleh karena itu, robot 1 segera berhenti di titik tersebut. Padahal pada kenyataannya, terdapat jarak yang lebih baik seperti yang ditunjukkan oleh garis *b* jika robot 1 berada pada posisi 4.

Untuk mengatasi hal ini, ide utamanya adalah mengganti pergerakan robot pada saat robot akan keluar dari rentang *sensing area* posisi 2 agar alih-alih robot berhenti ia akan bergerak menuju posisi 4 seperti pada gambar 3.7.c. Jika diperhatikan dari gambar tersebut dapat dilihat bahwa terdapat dua garis yang dilalui oleh posisi 4 tersebut, yaitu garis lurus *b* dan garis lengkung lingkaran batas *sensing area* posisi 2. Persamaan garis lurus *b* dapat diturunkan dari posisi 2 dan posisi 3 karena robot 1 tentu sudah memiliki informasi ini pada saat terhubung posisi 2 dan posisi 3 sebagai posisi tujuan. Persamaan lingkaran kemudian dapat diturunkan dari posisi robot 1 sendiri dengan jarak antara robot 1 dan posisi 2 yang merupakan radius dari *sensing area* kedua robot tersebut. Dengan melakukan substitusi persamaan garis dapat ditentukan posisi titik tempat kedua garis berpotongan, dalam hal ini adalah posisi 4. Dengan diketahuinya posisi 4, sekarang robot 1 dapat bergerak menuju posisi 4 dengan mengikuti algoritma pergerakan robot seperti semula. Robot 1 kemudian berhenti di posisi seperti yang ditunjukkan pada gambar 3.7.d.

Berikut adalah perumusan dari pergerakan robot ini. Berdasarkan algoritma pergerakan, robot bergerak setelah menentukan posisi tujuannya. Pada gambar 3.7.a di atas, pergerakan robot akan mengikuti tahap seperti berikut.

1. Bergerak menuju target dengan prioritas tertinggi, yaitu posisi 2.

$$\theta = \arctan((y_2 - y_1) / (x_2 - x_1))$$
$$\textbf{Loop until } |P_2 - P_1| <= C_s \qquad (3.8)$$
$$P_1' = (x_1 + \Delta d * \cos\theta, y_1 + \Delta d * \sin\theta)$$
$$\textbf{End Loop}$$

P_1' (x_1', y_1') = posisi robot baru setelah pergerakan.

P_1 (x_1, y_1) = posisi lama robot sebelum pergerakan.

P_2 (x_2, y_2) = posisi tujuan.

C_s = rentang *sensing area*.

Δx = jarak pergerakan, pada sumbu x.

Δy = jarak pergerakan, pada sumbu y.

Δd = jarak pergerakan per langkah, yaitu range robot.

Θ = sudut pergerakan terhadap sumbu x positif

2. Bergerak menuju target berikutnya, yaitu posisi 3, dengan tetap menjaga agar tetap berada dalam rentang *sensing area* posisi 2.

$$\theta = \arctan((y_3 - y_1) / (x_3 - x_1))$$

Loop until $|P_3 - P_1| <= C_s$

  If $|P_2 - P_1| >= C_s$ **Then**

    Break

  $P_1' = (x_1 + \Delta d * \cos\theta, y_1 + \Delta d * \sin\theta)$

End Loop **(3.9)**

P_1' (x_1', y_1') = posisi robot baru setelah pergerakan.

P_1 (x_1, y_1) = posisi lama robot sebelum pergerakan.

P_2 (x_2, y_2) = posisi tujuan.

C_s = rentang *sensing area*.

Δx = jarak pergerakan, pada sumbu x.

Δy = jarak pergerakan, pada sumbu y.

Δd = jarak pergerakan per langkah, yaitu range robot.

Θ = sudut pergerakan terhadap sumbu x positif

3. Tentukan posisi 4 dengan menemukan perpotongan antara persamaan lingkaran *sensing area* dan persamaan garis yang menghubungkan kedua target. Untuk menemukan posisi tersebut dilakukan dengan mensubstitusi persamaan garis ke dalam persamaan lingkaran, seperti berikut.

$$(x - x_2)^2 + (y - y_2)^2 = r^2$$

$$(y - y_2) / (y_3 - y_2) = (x - x_2) / (x_3 - x_2) \quad \textbf{(3.10)}$$

4. Misalkan posisi baru yang didapatkan adalah P(x, y), maka robot selanjutnya bergerak menuju posisi tujuan tersebut.

$$\theta = \arctan((y - y_1) / (x - x_1))$$

Loop until $|P - P_1| <= 0$

$$P_1' = (x_1 + \Delta d * \cos\theta, y_1 + \Delta d * \sin\theta) \quad (3.11)$$

End loop

P_1' (x_1', y_1') = posisi robot baru setelah pergerakan.

P_1 (x_1, y_1) = posisi lama robot sebelum pergerakan.

P (x, y) = posisi tujuan.

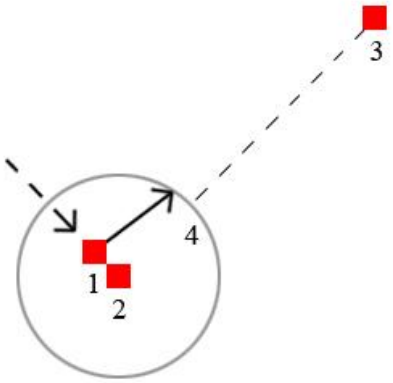
Δx = jarak pergerakan, pada sumbu x.

Δy = jarak pergerakan, pada sumbu y.

Δd = jarak pergerakan per langkah, yaitu range robot.

Θ = sudut pergerakan terhadap sumbu x positif

Selain metode yang dijabarkan di atas, proses optimisasi algoritma *Chained Shortest Distance* ini dapat juga dilakukan dengan cara mengganti nilai threshold pada langkah 1, dari radius *sensing area* menjadi *collision area*. Artinya, robot baru bergerak menuju target berikutnya ketika sudah akan terjadi benturan dengan target yang sedang ditujunya. Hal ini berguna untuk memberikan ruang gerak yang lebih luas kepada robot sebelum langsung keluar lagi dari rentang *sensing area* target yang baru saja dicapainya. Untuk lebih jelasnya, keterangan ini dapat dilihat pada gambar 3.8 berikut di bawah ini.



Gambar 3.8 - Alternatif Optimisasi Posisi

Meskipun dengan langkah ini, posisi terakhir robot 1 tidak sebaik posisi 4, tetapi langkah ini juga sudah dapat memberikan perbaikan kepada posisi terakhir robot 1 sebelum berhenti.

3.9 Algoritma Prior to the Nearest

Algoritma ini merupakan algoritma alternatif yang dikembangkan dalam rangka mendapatkan solusi yang lebih baik daripada algoritma yang dikembangkan sebelumnya. Algoritma ini diadaptasi dari penelitian lain yang memiliki beberapa kemiripan dengan permasalahan yang diteliti pada penelitian tugas akhir ini. Penelitian yang dimaksud adalah penelitian yang dilakukan oleh J. Takahashi, K. Sekiyama dan T. Fukuda dari Departemen Micro-nano Systems Engineering, Nagoya University, Jepang. Penelitian ini berjudul "*Cooperative Object Tracking with Mobile Robotic Sensor Network*". Sebelum membahas tentang algoritma yang digunakan untuk menyelesaikan permasalahan, terlebih dulu dijelaskan mengenai penelitian yang menjadi rujukan pengembangan algoritma.

3.9.1 Pengawasan Objek dengan Jaringan Robot Bergerak

Salah satu tren penggunaan teknologi robot pada saat sekarang adalah untuk keperluan eksplorasi atau penjelajahan karena robot dapat digunakan untuk mencapai daerah-daerah yang tidak dapat dicapai oleh manusia dalam kondisi

biasa. Agar dapat diperoleh manfaat melalui kegiatan penjelajahan yang dilakukan robot tersebut, maka diperlukan proses pemindahan informasi yang ditemukan oleh robot ke tempat yang di mana operator manusia melakukan pengawasan. Hal ini berkaitan dengan pemantauan kegiatan eksplorasi robot dari tempat yang berjauhan. Belakangan teknologi seperti ini juga sering digunakan untuk keperluan pengawasan objek yang berada pada suatu tempat yang jauh.

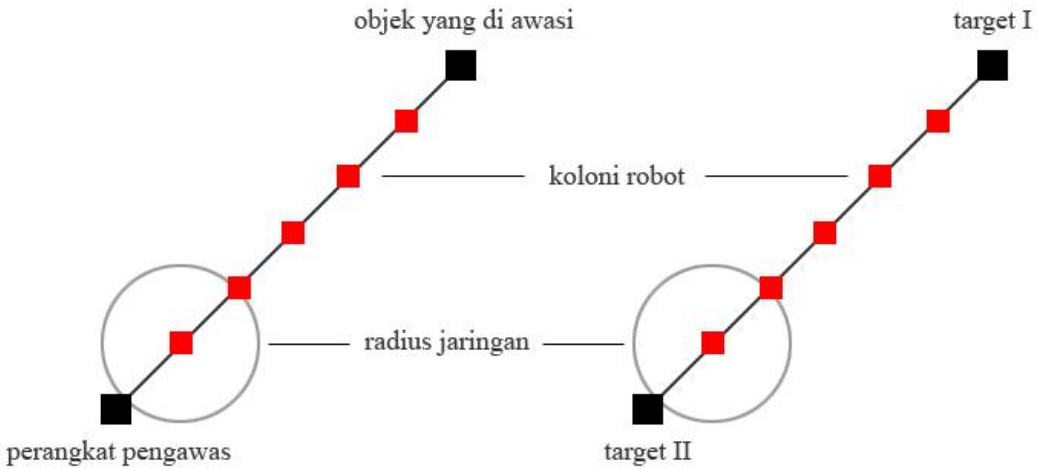
Salah satu permasalahan utama yang harus dipikirkan ketika membangun teknologi pemantauan jarak jauh adalah permasalahan komunikasi antara robot dengan perangkat pengawas. Teknologi komunikasi yang cocok untuk digunakan adalah jenis nirkabel yang tidak menggunakan media fisik untuk sarana pemindahan informasi karena robot diharapkan dapat bergerak bebas untuk keperluan penjelajahan. Salah satu permasalahan yang kemudian muncul adalah mahalnya biaya yang diperlukan untuk membangun jaringan komunikasi tanpa kabel yang dapat dihandalkan untuk jarak yang lebih jauh dan lebih luas.

Penelitian yang berjudul *Cooperative Object Tracking with Mobile Robotic Sensor Network* ini merupakan salah satu penelitian yang berupaya menyelesaikan permasalahan yang disebutkan di atas. Pada penelitian tersebut, yang menjadi fokus utama adalah pemantauan objek lain pada suatu jarak tertentu dengan menggunakan robot-robot bergerak yang dilengkapi perangkat komunikasi yang memadai. Penelitian ini sebenarnya dilakukan untuk mendukung kemajuan teknologi di bidang robotik, khususnya bidang jaringan robot. Akan tetapi, karena permasalahan komunikasi antar robot-robot ini menjadi salah satu bagian penting yang harus diperhitungkan, hasil yang ditemukan dari penelitian ini diharapkan juga dapat mendukung kemajuan teknologi di bidang komunikasi sendiri.

Tujuan utama dari penelitian yang dilakukan oleh Takahashi dkk ini sebenarnya adalah untuk membangun suatu jaringan robot yang memungkinkan proses pemantauan objek untuk yang jarak yang lebih jauh dan lebih luas. Hal ini diterapkan dengan memanfaatkan penggunaan koloni robot untuk mendukung proses ekplorasi yang lebih baik. Masing-masing robot dilengkapi perangkat komunikasi nirkabel yang memadai untuk saling berkomunikasi satu sama lain

dan nantinya akan melakukan estafet informasi untuk membentuk suatu jaringan robot. Hal ini serupa dengan permasalahan yang diteliti pada penelitian tugas akhir penulis, yaitu untuk menemukan konfigurasi jaringan komunikasi yang sesuai dalam rangka menghubungkan beberapa daerah yang berjauhan.

Gambar 3.9 berikut di bawah merupakan representasi visual dari permasalahan yang diteliti pada kedua penelitian ini. Gambar bagian kiri merupakan penelitian yang dilakukan oleh Takahashi dkk, sedangkan gambar pada bagian kanan adalah permasalahan yang akan diselesaikan pada tugas akhir ini.



Gambar 3.9 - Permasalahan pada Kedua Penelitian

Berdasarkan gambar tersebut dapat dilihat bahwa meskipun tujuan utama yang akan diselesaikan pada kedua penelitian ini sedikit berbeda, tetapi secara umum kedua permasalahan ini dapat dimodelkan ke dalam bentuk permasalahan yang serupa. Untuk menghubungkan suatu daerah dalam suatu jaringan komunikasi nirkabel tertentu dapat dilakukan dengan menyusun beberapa perangkat-perangkat teknologi nirkabel yang lebih kecil hingga membentuk suatu rute komunikasi. Dengan demikian, hal ini dapat dilakukan tanpa harus memiliki suatu perangkat jaringan nirkabel berkapasitas besar. Hal ini selain bertujuan menekan biaya yang dibutuhkan untuk membangun jaringannya akan dapat pula meningkatkan kehandalan dari jaringan tersebut.

3.9.2 Penyelesaian Masalah

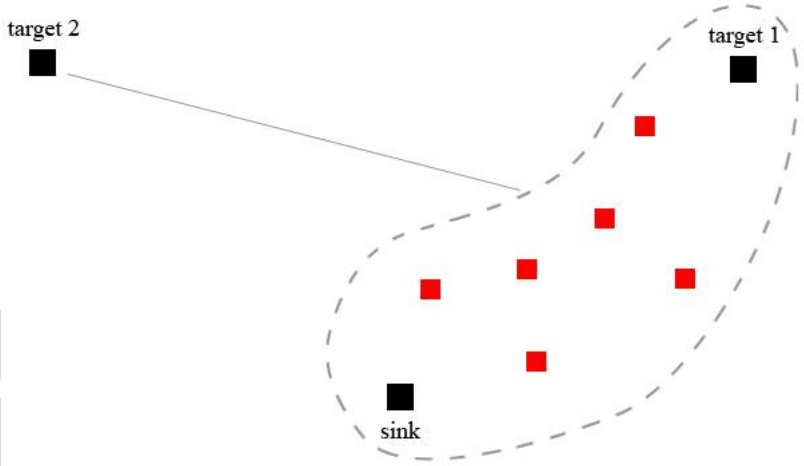
Untuk penjelasan lebih detail mengenai metode penyelesaian masalah dengan menggunakan algoritma ini dapat merujuk pada bagian 2.5 sebelumnya.

3.9.3 Keterbatasan Penelitian

Algoritma yang diajukan oleh Fukuda dkk pada penelitian yang mereka lakukan memang sudah dapat menemukan solusi permasalahan yang didefinisikan sebelumnya. Akan tetapi, pada kenyataannya terdapat beberapa permasalahan tertentu yang belum dapat diselesaikan oleh algoritma ini. Berikut adalah beberapa keterbatasan yang masih dijumpai dari penelitian ini.

1. Algoritma ini belum menjamin bahwa solusi yang ditemukan adalah solusi yang paling optimum untuk permasalahan ini. Jika simulasi diterapkan dalam kehidupan nyata, maka proses penekanan biaya yang diperlukan untuk membangun sistem yang sesuai pun akan sulit untuk dilakukan.
2. Tidak adanya metode untuk mengetahui apakah terdapat solusi terhadap permasalahan yang sedang dicari atau tidak. Pada penelitian ini jumlah robot otonom yang digunakan pada saat simulasi diasumsikan selalu melebihi jumlah minimal untuk dapat memenuhi kebutuhan pembangunan jaringan. Padahal pada kenyataannya dalam kehidupan sehari-hari jumlah sumber daya yang dapat digunakan tidak selalu tidak terbatas.
3. Belum dilakukan pengujian terhadap kasus pengawasan terhadap banyak objek di dalam simulasi. Pada kenyataannya terdapat beberapa kondisi yang tidak dapat diselesaikan oleh algoritma ini. Gambar 3.10 berikut di bawah merupakan salah satu contoh kasus yang tidak dapat diselesaikan oleh algoritma seperti yang telah dijabarkan sebelumnya. Posisi awal robot ditentukan secara acak sehingga terdapat kemungkinan robot-robot otonom mengumpul di suatu tempat tertentu saja pada saat simulasi dimulai. Hal ini akan menjadi masalah ketika objek yang akan diawasi lebih dari satu, di mana salah satu objek tersebut terpisah jauh dari daerah tempat robot

berkumpul karena robot menentukan prioritas pergerakannya berdasarkan jarak antara robot tersebut dari objek. Pada gambar 3.10 tersebut, target 2 tidak akan pernah dijadikan prioritas karena jaraknya yang jauh dari posisi robot di dalam simulasi.



Gambar 3.10 - Koloni Robot Terkumpul pada Lokasi Tertentu

3.10 Algoritma Position By Line

Algoritma berikutnya yang menjadi alternatif bagi metode penyelesaian pada penelitian tugas akhir ini adalah algoritma *Position By Line*. Algoritma ini dimaksudkan untuk mengatasi keterbatasan dari kedua algoritma yang sudah dijabarkan pada bagian sebelumnya, terutama pada bagian untuk menjamin bahwa solusi yang ditemukan merupakan solusi yang paling optimum. Adapun beberapa kriteria yang diperhitungkan dalam rangka mencapai solusi yang dianggap paling optimum ini antara lain sebagai berikut.

1. Dapat menemukan solusi untuk jumlah robot yang paling minimum. Jika solusi tidak ditemukan, berarti jumlah robot memang tidak mencukupi sehingga tidak aka nada solusi. Dengan demikian, dapat diketahui jumlah robot minimal yang dapat digunakan untuk membangun konfigurasi jaringan komunikasi *multihop* pada penelitian ini.

2. Dapat menemukan jalur yang terpendek antara target-target. Artinya, dari seluruh kemungkinan solusi konfigurasi yang dapat dibuat, solusi yang ditemukan adalah jalur yang menghubungkan target-target dalam jarak yang paling pendek. Dengan demikian, waktu yang diperlukan ketika dilakukan proses komunikasi akan jadi lebih singkat, dengan asumsi lama waktu yang dibutuhkan untuk berkomunikasi berbanding lurus dengan jarak jalur yang tersedia antara perangkat yang saling berhubungan.

Sebelum dijelaskan mengenai langkah penyelesaian pada algoritma ini, akan dijelaskan beberapa asumsi yang digunakan, yaitu sebagai berikut.

1. Spesifikasi robot yang digunakan seperti pada algoritma *Chained Shortest Distance*, yaitu terdapat empat fungsi utama: *sensing, communication, computation* dan *locomotion*. Robot hanya dapat berkomunikasi dengan objek lain yang berada di dalam rentang *sensing area*-nya.
2. Robot diberikan tambahan informasi mengenai jumlah robot lain di dalam simulasi pada saat fase inisialisasi.

Berikut adalah langkah penyelesaian masalah pada algoritma *Position By-Line* ini.

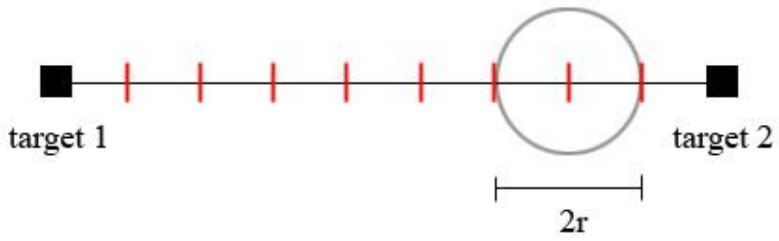
1. Robot otonom memulai simulasi dengan posisi awal yang acak, sedangkan perangkat pengawas dan objek yang diawasi harus diketahui posisinya.
2. Robot menyimpan informasi yang dibutuhkannya untuk melakukan kegiatan eksplorasi, yaitu informasi mengenai posisi dirinya masing-masing beserta informasi mengenai posisi target, serta jumlah robot di dalam simulasi. Akan tetapi, robot tidak mengetahui posisi robot-robot lainnya. Informasi ini diberikan kepada robot otonom pada fase inisialisasi.
3. Robot memanfaatkan informasi posisi target yang akan dihubungkan oleh jalur komunikasi, serta informasi radius *sensing area*-nya untuk mengetahui posisi-posisi mana saja yang akan ditempati oleh semua robot agar dapat ditemukan jalur berupa garis lurus antara target di dalam simulasi. Posisi-posisi yang mungkin ditempati ini disebut *feasible position.*

4. Setelah melakukan perhitungan untuk mendapatkan semua *feasible position*, robot kemudian menghitung jarak dari posisinya kepada semua *feasible position*. Posisi dengan jarak terpendek kemudian dijadikan prioritas tertinggi, hingga seterusnya sampai posisi terjauh dengan prioritas terendah.
5. Robot kemudian bergerak mendekati *feasible position* dengan prioritas tertinggi. Jika robot tidak dapat mencapai posisi tersebut dikarenakan telah adanya robot lain yang menempatinya, maka robot menentukan *feasible position* dengan prioritas tertinggi berikutnya sebagai posisi tujuan. Robot bergerak menuju posisi tersebut.
6. Ulangi langkah 5 secara berulang jika robot masih belum berhasil mencapai posisi yang ditujunya. Jika semua *feasible position* tidak dapat dikunjungi, maka robot akan berhenti.

Algoritma ini bekerja dengan memanfaatkan persamaan garis yang menghubungkan target dalam suatu garis lurus. Berdasarkan teori geometri, jarak terpendek antara dua titik adalah garis lurus yang menghubungkan keduanya sehingga jika algoritma ini dapat menemukan konfigurasi berupa garis lurus antara kedua target, maka solusi yang ditemukan adalah solusi yang paling optimum. Persamaan garis lurus yang menghubungkan kedua titik tersebut dapat diturunkan jika diketahui posisi kedua titik.

Syarat yang dibutuhkan untuk menentukan persamaan garis ini sudah dipenuhi karena robot otonom di dalam simulasi menyimpan informasi posisi target-target yang akan dihubungkan oleh jaringan komunikasi yang sedang dibangun. Dengan menggunakan informasi ini, dapat ditentukan jarak antara kedua target dalam garis lurus tersebut. Jarak maksimal antarrobot hanya dibolehkan sejauh radius *sensing area*-nya agar dapat saling terhubung satu sama lain. Jika jarak total antara target-target ini dibagi dengan nilai radius *sensing area* robot otonom, akan diperoleh jumlah robot yang diperlukan untuk dapat menyusun garis lurus tersebut. Dengan membagi garis lurus tadi sebanyak robot yang ada, dapat ditentukan posisi-posisi mana yang harus ditempati oleh masing-masing robot agar target-target di dalam simulasi dapat saling terhubung. Gambar 3.11 di

bawah ini akan menjelaskan mengenai prinsip yang digunakan pada algoritma *Position By-Line* ini.



Gambar 3.11 - Pembagian Garis dengan Radius Sensing Area

Dari gambar 3.11 tersebut dapat dilihat bahwa setelah menentukan persamaan garis yang menghubungkan kedua target dibagi dengan nilai *r* yang merupakan jarak radius *sensing area* robot otonom sehingga didapatkan posisi-posisi yang disebut sebagai *feasible position* yang ditunjukkan oleh garis-garis vertikal merah. Posisi inilah yang kemudian ditempati oleh semua robot otonom agar kedua target saling terhubung dalam jaringan komunikasi.

Salah satu keunggulan lain dari algoritma ini adalah dapat menentukan jumlah robot minimum untuk menemukan solusi. Jika jumlah robot yang tersedia tidak mencukupi jumlah minimum robot yang diperlukan tersebut, maka robot-robot otonom tidak akan memulai simulasi hingga jumlah robot yang ada mencukupi.

Berikut adalah cara untuk menentukan apakah terdapat solusi jika diberikan suatu permasalahan yang harus diselesaikan dengan *m* unit robot otonom.

1. Tentukan jarak antara target dengan menggunakan rumus dua titik.

$$\mathbf{d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \qquad \mathbf{(3.12)}$$

D = jarak antara kedua target.

$P_1 (x_1, y_1)$ = posisi target 1.

$P_2 (x_2, y_2)$ = posisi target 2.

2. Untuk menentukan apakah terdapat solusi atau tidak, digunakan rumus sebagai berikut.

If (d / r <= m) then TRUE, Else then FALSE **(3.13)**

D = jarak antara kedua target.

R = radius *sensing area*.

M = Jumlah robot dalam simulasi.

3. Lakukan simulasi jika nilai yang diperoleh pada langkah bernilai TRUE.

Selanjutnya, untuk menentukan *feasible solution* robot dapat dimanfaatkan informasi jarak antara target, radius *sensing area* yang menentukan jarak maksimal antara robot, jumlah robot yang ada dalam simulasi, serta kemiringan garis terhadap sumbu *x*. Hal ini dilakukan dengan cara menggunakan rumus sebagai berikut.

For i=0 until (d / r)

$$P_i = (i * r * \cos\theta + x_1, i * r * \sin\theta + y_1) \quad (3.14)$$

End loop

Dari penjabaran di atas dapat dilihat bahwa algoritma *Position By-Line* dapat menemukan solusi yang paling optimum dan dapat digunakan untuk menentukan jumlah robot paling minimum yang diperlukan agar terdapat solusi. Kedua hal ini belum dapat dilakukan pada metode yang dikembangkan sebelumnya, baik pada algoritma *Chained Shortest Distance* maupun algoritma *Prior to the Nearest*.

BAB 4 PENANGANAN JARINGAN KOMUNIKASI MULTIHOP TERKONFIGURASI SENDIRI UNTUK MULTISOURCE-COMMUNICATION

Bab ini akan menjelaskan tentang penanganan jaringan untuk komunikasi antara banyak sumber pada suatu kelompok. Pada bagian awal bab dijelaskan perbandingan antara penanganan jaringan untuk *multisource-communication* dengan penanganan jaringan untuk *pairform-communication,* seperti yang dijabarkan pada bagian sebelumnya. Kemudian akan dijelaskan rancangan algoritma dan metode penyelesaian yang diterapkan untuk menemukan solusi permasalahan ini.

4.1 Permasalahan Banyak Sumber

Pada bagian sebelumnya telah dijelaskan bagaimana cara membangun konfigurasi jaringan komunikasi *multihop* terhadap permasalahan dengan dua sumber yang berkomunikasi. Terdapat tiga algoritma yang dikembangkan sebagai metode penyelesaian, yaitu: *Chained Shortest Distance*, *Prior to the Nearest*, serta *Position By Line*.

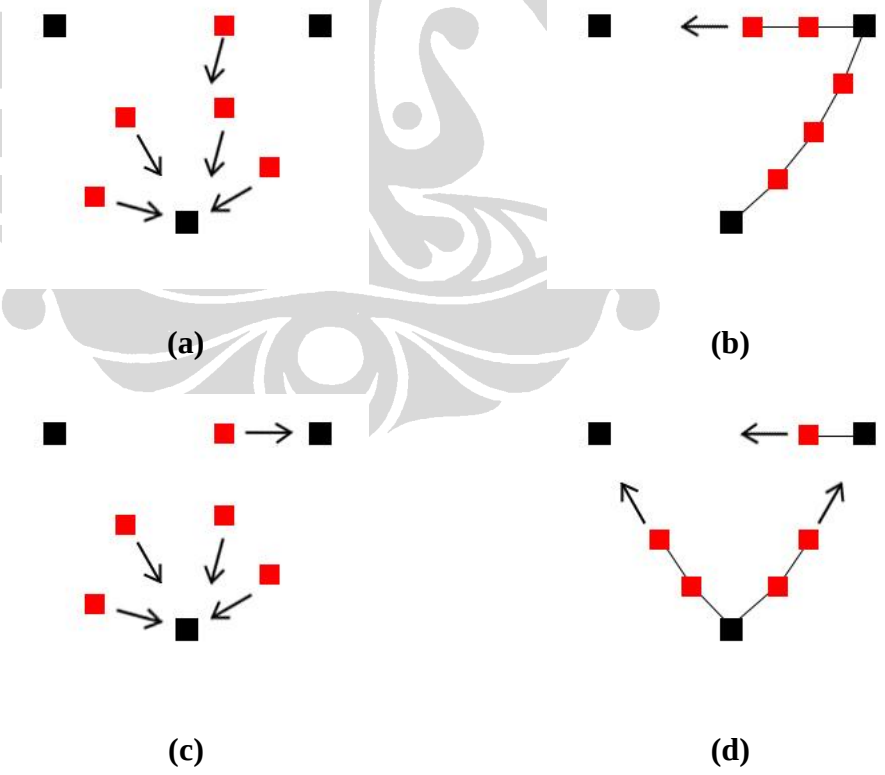
Penelitian tugas akhir ini kemudian akan dilanjutkan kepada tahap implementasi algoritma ke dalam simulasi untuk melihat kinerja dari masing-masing algoritma yang dikembangkan tersebut. Akan tetapi, sebelumnya pada bagian ini akan dibahas dulu mengenai penanganan jaringan komunikasi *multihop* terhadap sumber yang berkomunikasi berjumlah banyak atau lebih dari dua. Pada bab ini akan dilihat beberapa keadaan yang ditemui jika masing-masing algoritma diterapkan pada permasalahan banyak sumber ini, dibandingkan dengan permasalahan dengan dua sumber.

4.2 Algoritma CSD untuk Banyak Sumber

Algoritma *Chained Shortest Distance* atau CSD, pada prinsipnya membangun jaringan komunikasi melalui hubungan antara robot dan target, baik secara

langsung melalui hubungan robot dengan target, maupun secara tidak langsung nelalui hubungan robot dengan robot hingga mencapai target. Pengecekan hubungan ini dilakukan dengan menggunakan algoritma *Depth First Search*.

Adapun supaya robot dapat terhubung dengan target, pertama kali robot menentukan urutan prioritas target-target yang ada, untuk kemudian akan mengunjungi target-target tersebut sesuai dengan urutan prioritas ini. Ketika robot telah terhubung dengan suatu target yang sedang ditujunya, robot selanjutnya akan mengubah tujuannya kepada target dengan prioritas tertinggi berikutnya. Jika selama pergerakannya robot akan terputus dengan target yang sebelumnya sudah dikunjungi atau telah terhubung, maka robot akan berhenti bergerak. Dengan melakukan hal ini secara berulang-ulang, maka pada saat target terakhir terhubung dengan sekurang-kurangnya satu robot, maka konfigurasi jaringan komunikasi akan ditemukan. Gambar 4.1 berikut di bawah akan menjelaskan urutan algoritma CSD ini pada kasus jaringan komunikasi dengan banyak sumber.



Gambar 4.1 - Pembagian Garis dengan Radius Sensing Area

Pada gambar tersebut, terdapat dua kasus yang berbeda, yaitu gambar 4.1.a dan 4.1.b sebagai kasus pertama, serta gambar 4.1.c dan 4.1.d sebagai kasus kedua.

Berdasarkan gambar tersebut dapat dilihat bahwa pergerakan robot sangat ditentukan oleh bagaimana dia menentukan urutan prioritas pada target. Pada kasus pertama, pemberian prioritas kepada target dilakukan secara seragam. Semua robot akan mengunjungi target dalam urutan yang sama, sehingga proses konfigurasi jaringan dilakukan layaknya pembangunan jalur komunikasi yang dimulai dari suatu titik, dilanjutkan kepada titik-titik yang lain hingga semua target terhubung dalam konfigurasi jaringan yang diharapkan.

Sedangkan kasus kedua, pemberian prioritas kepada target dilakukan secara berbeda-beda, yaitu tergantung kepada kedekatan robot dari masing-masing target. Hal ini akan menyebabkan robot mengunjungi target dimulai dari yang paling dekat dengannya, sehingga proses konfigurasi jaringan dilakukan layaknya pembangunan jalur komunikasi dari semua titik hingga semua jalur ini akan bersatu dalam suatu konfigurasi yang sama. Selanjutnya algoritma pergerakan robot pada kedua kasus mengikuti algoritma yang sudah dijelaskan pada permasalahan dengan dua sumber pada bab sebelumnya.

4.3 Algoritma PTN untuk Banyak Sumber

Algoritma *Prior to the Nearest* atau PTN, merupakan algoritma yang diadopsi dari penelitian lain yang dilakukan oleh Fukuda, dkk. Pada penelitian tersebut, algoritma ini digunakan untuk keperluan pengawasan objek dari suatu perangkat pengawas tertentu. Oleh karena itu, algoritma ini baru dapat menyelesaikan permasalahan jaringan komunikasi dengan dua sumber.

Prinsip utama yang digunakan pada algoritma ini adalah robot menentukan salah satu robot lainnya yang berjarak lebih dekat terhadap target, atau target itu sendiri jika robot merupakan objek terdekat dari target, sebagai prioritas tujuan pergerakan. Karena algoritma ini bertujuan untuk menghubungkan dua target dalam suatu konfigurasi jaringan komunikasi, maka terdapat dua prioritas

pergerakan, yaitu dua robot terdekat yang berjarak lebih dekat terhadap kedua target atau dirinya sendiri. Robot kemudian bergerak mendekati prioritas pertama, sebelum bergerak menuju prioritas kedua. Pada saat mendekati prioritas kedua, robot harus tetap terhubung dengan prioritas pertama.

Permasalahan baru muncul ketika target yang akan dihubungkan berjumlah lebih dari dua. Seperti yang sudah dijelaskan pada bagian keterbatasan penelitian pada bab sebelumnya, jika ada salah satu target yang berjarak sangat jauh dari kelompok robot, maka target tersebut tidak akan pernah dijadikan sebagai prioritas. Hal ini akan menyebabkan konfigurasi jaringan yang ditemukan tidak dapat menemukan semua target.

Permasalahan ini sebenarnya dapat diatasi dengan memecah persoalan jaringan komunikasi dengan banyak sumber menjadi beberapa sub-jaringan yang masing-masingnya terdiri atas dua sumber, kemudian menyelesaikannya dengan algoritma sebelumnya. Akan tetapi, cara seperti ini tentu memerlukan metode tertentu untuk menemukan sub-jaringan yang sesuai, serta untuk menentukan pembagian robot dalam menyelesaikan sub-jaringan yang diharapkan.

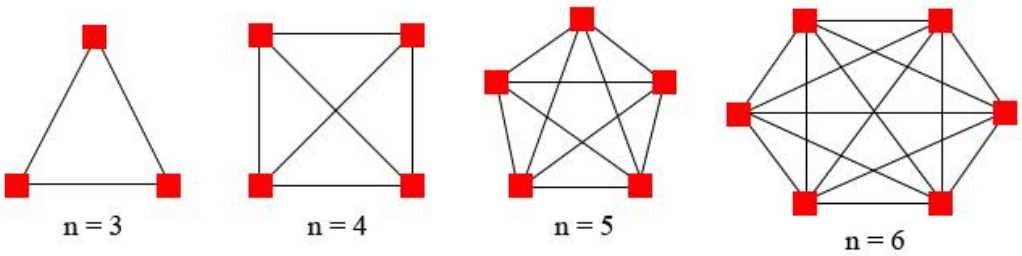
4.4 Algoritma PBL untuk Banyak Sumber

Algoritma *Position By Line* atau PBL adalah algoritma lain yang dikembangkan untuk menyelesaikan permasalahan pada penelitian tugas akhir ini. Tujuan algoritma ini adalah menemukan solusi yang dapat dijamin sebagai solusi yang paling optimum dari jumlah robot yang tersedia untuk menemukan konfigurasi jaringan komunikasi nirkabel, karena dua algoritma sebelumnya tidak memiliki sifat seperti ini. Selain itu, algoritma ini juga diharapkan dapat digunakan untuk membangun konfigurasi jaringan komunikasi nirkabel adapatif untuk permasalahan banyak sumber yang berkomunikasi.

Pada permasalahan dengan dua sumber, algoritma ini menemukan solusi permasalahan dengan menentukan posisi yang layak untuk ditempati oleh semua robot dalam rangka menemukan konfigurasi jaringan komunikasi yang sesuai.

Untuk mendapatkan posisi-posisi ini, dilakukan dengan cara memanfaatkan persamaan garis lurus yang menghubungkan kedua target, karena berdasarkan teori geometri, jarak terdekat antara dua titik ditentukan oleh garis lurus yang menghubungkannya. Selain itu, dengan memanfaatkan informasi ini juga dapat ditentukan jumlah robot minimal yang diperlukan untuk menghubungkan kedua target dalam garis lurus tersebut. Dengan demikian, solusi yang ditemukan pada algoritma ini akan bersifat optimum.

Kemudian pada permasalahan dengan banyak sumber, sebenarnya juga akan digunakan prinsip yang serupa. Konfigurasi jaringan paling optimum yang dicari merupakan garis lurus yang menghubungkan semua titik, sesuai dengan teori geometri yang disebutkan di atas. Oleh karena itu, solusi yang paling optimum untuk permasalahan dengan banyak sumber yang lebih dari dua, dapat dilihat pada gambar 4.2 berikut di bawah ini.



Gambar 4.2 - Konfigurasi Optimum untuk Permasalahan Banyak Sumber

Berdasarkan gambar di atas, dapat dilihat bahwa jika permasalahan banyak sumber ini direpresentasikan dalam bentuk graph, dimana target-target yang akan dihubungkan oleh konfigurasi jaringan komunikasi menjadi semua vertex dan jalur yang dibentuk oleh robot menjadi semua edge, maka solusi paling optimum yang dapat ditemukan pada permasalahan ini adalah *complete graph* dari jumlah target yang akan dihubungkan.

4.4.1 Cost dan Distance

Salah satu karakteristik yang dimiliki oleh algoritma *Position By Line* adalah dengan menyimpan informasi jumlah robot yang tersedia, dapat ditentukan ada

atau tidaknya solusi pada permasalahan yang diberikan. Simulasi hanya dilanjutkan jika jumlah robot memenuhi jumlah minimum yang dibutuhkan untuk menemukan solusi, sehingga dengan jumlah robot paling sedikit pun masih akan ditemukan konfigurasi jaringan komunikasi yang diharapkan.

Pada permasalahan dengan dua sumber, ada tidaknya solusi dapat ditentukan menentukan jumlah robot yang dibutuhkan untuk membentuk konfigurasi berupa garis lurus antara kedua target. Jika jumlah robot yang tersedia tidak memenuhi jumlah ini, maka dapat disimpulkan tidak ada solusi pada permasalahan tersebut.

Hal seperti ini dapat dilakukan, karena selain merupakan jarak yang paling optimum antara kedua target, garis lurus yang menghubungkan kedua target juga merupakan jalur dengan biaya yang paling minimum untuk menghubungkan semua target dalam jaringan komunikasi yang diharapkan. Hal ini kemudian disebut sebagai *cost* dan *distance* pada solusi konfigurasi jaringan.

Pada permasalahan dengan banyak sumber, perlu diperhatikan bahwa terdapat perbedaan antara nilai *cost* dan *distance* ini. Solusi yang paling optimum pada permasalahan ini adalah seperti yang ditunjukkan pada gambar 4.2 sebelumnya, memiliku nilai *distance* paling minimum, karena masing-masing target terhubung secara langsung dengan target yang lain. Akan tetapi, *cost* yang dibutuhkan akan besar karena banyaknya jalur yang harus dibuat untuk dapat menghubungkan semua target dalam konfigurasi jaringan. Hal ini menunjukkan bahwa nilai *cost* dan *distance* berbanding terbalik pada permasalahan dengan banyak sumber.

Pada kenyataannya, solusi permasalahan dengan banyak sumber bukanlah solusi yang unik. Selama semua target dapat terhubung dalam suatu graph yang sama, maka konfigurasi tersebut dapat diterima sebagai suatu solusi. Dengan kata lain, konfigurasi graph yang dapat menjadi solusi bagi permasalahan dengan banyak sumber adalah *connected graph* dari sejumlah target yang diberikan. Kemudian, dengan menentukan konfigurasi graph dengan *cost* terkecil, dapat ditentukan ada atau tidaknya solusi terhadap permasalahan yang diberikan. Karena *connected graph* dengan *cost* terkecil dari suatu graph merupakan *minimum spanning tree*

dari graph tersebut, maka rumus untuk menentukan jumlah robot minimum untuk menemukan solusi dapat diturunkan sebagai berikut.

If (MST's cost <= robot(s)) then TRUE, Else then FALSE (4.1)

Dari penjabaran ini, kemudian secara umum permasalahan dengan banyak sumber yang akan diselesaikan dengan menggunakan algoritma *Position By Line*, dapat dirumuskan sebagai berikut.

Min (Distance), where MST <= Robots <= Complete Graph (4.2)

4.4.2 Algoritma Extended Position By Line

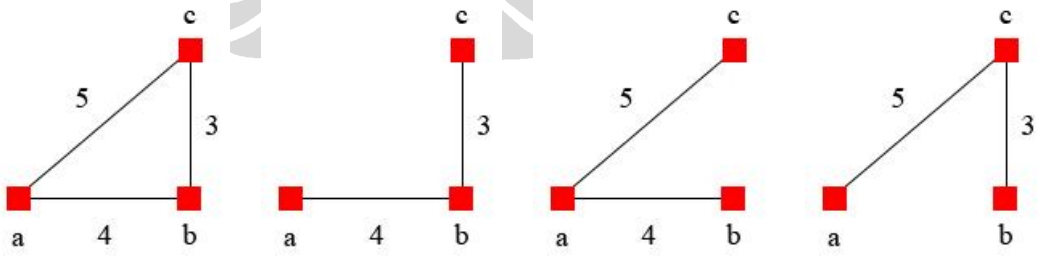
Untuk dapat menyelesaikan permasalahan dengan banyak sumber ini, penulis kemudian merancang algoritma baru yang merupakan lanjutan dari algoritma *Position By Line* yang sudah dikembangkan sebelumnya. Pada dasarnya, prinsip yang digunakan pada algoritma ini masih sama dengan algoritma *Position By Line*, yaitu dengan menemukan posisi-posisi yang layak untuk ditempati oleh semua robot, dapat ditentukan solusi bagi permasalahan konfigurasi jaringan komunikasi yang diharapkan. Adapun cara untuk menemukan solusi ini adalah dengan merepresentasikan solusi permasalahan dalam suatu *connected graph*, akan ditentukan bentuk konfigurasi graph yang baik untuk digunakan sehingga dapat diperoleh nilai *distance* minimum antara semua target yang ada, dengan batasan nilai *cost* jumlah robot yang tersedia.

Adapun langkah penyelesaian yang dilakukan pada algoritma *Extended Position By Line* ini adalah sebagai berikut.

1. Hitung jarak antara masing-masing target dalam jaringan yang akan dicari konfigurasinya. Nilai-nilai ini kemudian akan disebut sebagai *path*.

2. Tentukan semua kemungkinan *path* yang jika dijumlahkan akan bernilai lebih kecil atau sama dengan nilai *cost* jumlah robot yang tersedia.
3. Ambil setiap konfigurasi *path* yang dapat membentuk *connected graph* pada target-target yang akan dihubungkan oleh konfigurasi jaringan. Konfigurasi ini kemudian akan disebut sebagai solusi sementara.
4. Tentukan nilai *distance* dari semua target terhadap target yang lain. Nilai ini diperoleh dengan menjumlahkan *path* yang dapat menghubungkan suatu target terhadap target lain. Jika terdapat lebih dari satu kemungkinan *path*, pilih yang jumlahnya lebih minimum.
5. Jumlahkan semua nilai *distance*, dimana hasilnya akan menentukan *total distance* dari solusi sementara tersebut.
6. Pilih satu solusi sementara dengan *total distance* yang paling kecil. Solusi ini kemudian merupakan solusi akhir konfigurasi jaringan yang dicari.
7. Gunakan algoritma *Position By Line* untuk menentukan *feasible positions* pada setiap garis yang termasuk ke dalam solusi konfigurasi jaringan.

Untuk lebih jelasnya, berikut diberikan contoh permasalahan yang akan diselesaikan dengan menggunakan algoritma *Extended Position By Line*. Kondisi jaringan ditunjukkan oleh gambar 4.3 di bawah ini. Pada contoh permasalahan ini, terdapat tiga target yang akan dihubungkan oleh konfigurasi yang dicari.



(a) **(b)** **(c)** **(d)**

Gambar 4.3 - Kemungkinan Konfigurasi pada Kasus 3 Target

Misalkan pada contoh permasalahan ini, diberikan 9 unit robot yang akan digunakan untuk menemukan solusi permasalahan. Maka, langkah pertama yang harus dilakukan adalah menentukan apakah terdapat solusi atau tidak untuk permasalahan yang diberikan, yaitu sebagai berikut.

If (MST's cost <= Robots) Then TRUE

If ((3 + 4) <= 9) Then TRUE **(4.3)**

If (7 <= 9) Then TRUE

TRUE

Karena *cost* dari *Minimum Spanning* Tree, yang ditunjukkan pada gambar 4.3.b, dalam permasalahan ini lebih kecil daripada jumlah robot yang tersedia , maka permasalahan ini akan dapat ditemukan solusinya, yang fapat dilakukan dengan mengikuti langkah-langkah sebagai berikut.

1. Terdapat 3 path pada konfigurasi graph tersebut, yaitu:
   a. A – B = 4 robot.
   b. B – C = 5 robot.
   c. A – C = 3 robot.
2. Semua kemungkinan terbentuknya konfigurasi *connected graph* adalah:
   a. Konfigurasi dengan 3 path, yaitu seperti pada gambar 4.3.a. *Cost* yang dibutuhkan adalah 4 + 5 + 3 = 12 robot.
   b. Konfigurasi dengan 2 path, yaitu seperti pada gambar 4.3.b, 4.3.c dan 4.3.d. Masing-masing *cost* yang dibutuhkan adalah:
      a) Gambar 4.3.b = 4 + 3 = 7 robot.
      b) Gambar 4.3.c = 4 + 5 = 9 robot.
      c) Gambar 4.3.d = 5 + 3 = 8 robot.

3. Karena hanya tersedia 9 unit robot, maka tidak mungkin dibentuk solusi dengan menggunakan *complete* graph, sehingga konfigurasi yang mungkin menjadi solusi permasalahan adalah salah satu di antara gambar 4.3.b, 4.3.c dan 4.3.d yang mempunyai nilai *distance* paling minimum.

4. Nilai *distance* pada masing-masing konfigurasi dapat ditentukan dengan menggunakan matriks *distance* sebagai berikut.

   a. Gambar 4.3.b

   $$\text{Distance} = \begin{matrix} 0 & 4 & 7 \\ 4 & 0 & 3 \\ 7 & 3 & 0 \end{matrix} = \begin{matrix} 11 \\ 7 \\ 10 \end{matrix}$$

   Total Distance = 11 + 7 + 10 = 28

   b. Gambar 4.3.c

   $$\text{Distance} = \begin{matrix} 0 & 4 & 5 \\ 4 & 0 & 9 \\ 5 & 9 & 0 \end{matrix} = \begin{matrix} 9 \\ 13 \\ 14 \end{matrix}$$

   Total Distance = 9 + 13 + 14 = 36

   c. Gambar 4.3.d

   $$\text{Distance} = \begin{matrix} 0 & 8 & 5 \\ 8 & 0 & 3 \\ 5 & 3 & 0 \end{matrix} = \begin{matrix} 13 \\ 11 \\ 8 \end{matrix}$$

   Total Distance = 13 + 11 + 8 = 32

5. Solusi konfigurasi jaringan adalah konfigurasi graph yang memiliki nilai *total distance* yang paling minimum. Maka solusi pada contoh permasalahan ini adalah konfigurasi seperti yang ditunjukkan pada gambar 4.3.b.

6. Dengan menggunakan algoritma *Position By Line* dapat ditemukan semua kemungkinan posisi yang dapat ditempati oleh robot-robot.