PERBANDINGAN ALGORITMA PERLUASAN JARINGAN TELEKOMUNIKASI DENGAN *MOBILE ROBOT* DI DAERAH BENCANA MENGGUNAKAN *OPEN DYNAMIC ENGINE*

Abdullah Hafidh, M. Eka Suryana, Muhammad Fajar, dan Arief Ramadhan

Fakultas Ilmu Komputer, Universitas Indonesia, Kampus Baru UI, Depok, 16424, Indonesia

E-mail: abdullah.hafidh@ui.ac.id

Abstrak

Putusnya jaringan komunikasi sebagai dampak suatu bencana merupakan permasalahan yang perlu cepat diatasi agar tidak menimbulkan kerugian yang lebih besar. Komunikasi antara lokasi bencana dengan dunia luar menjadi sangat penting dalam menentukan tindakan yang tepat untuk meminimalisasi kerugian akibat bencana. Kajian penggunaan *mobile robot* otonom untuk menghubungkan menara-menara komunikasi yang terputus dapat menjadi solusi alternatif untuk mengatasi masalah ini. Dalam penelitian ini, penulis mengkaji beberapa algoritma perluasan jaringan dalam bentuk simulasi menggunakan *Library Open Dynamic Engine*. *Library Open Dynamic Engine* merupakan *library* yang telah menggunakan perhitungan fisika secara efisien dan teruji sehingga dapat mendekati keadaan dunia sebenarnya. Simulasi ini bertujuan untuk mendemontrasikan serta membandingkan bahwa algoritma ini dapat dikembangkan untuk menghubungkan menara komunikasi.

Kata Kunci: *daerah bencana, mobile robots, open dynamic engine, otonom, simulasi*

Abstract

The disconnection in communication networks as the impact of a disaster is a problem that needs to be addressed quickly in order to avoid greater losses. Communication between the disaster site with the outside world becomes very important in determining the appropriate action to minimize losses due to disasters. Study the use of autonomous mobile robots for connecting communication towers that was disconnected can be alternative solutions to address these issues. In this study, the authors examine some of the network expansion algorithm in the form of a simulation using Open Dynamics Engine library. Library of Open Dynamic Engine is a library that has used physics calculations in an efficient and tested so can approach the state of the real world. This simulation aims to demonstrate and compare that this algorithm can be developed to connect the communication towers.

Keywords: *autonomous, disaster area, mobile robots, open dynamic engine, simulation*

1. Pendahuluan

Komunikasi pasca bencana merupakan hal penting sehingga dunia luar dapat mengakses lokasi bencana dan dapat memperkirakan seberapa besar dampak kerusakan maupun korban jiwa di lokasi tersebut. Namun, suatu bencana terkadang merusak beberapa bangunan fisik komunikasi sehingga menimbulkan terputusnya komunikasi antara lokasi bencana dan lokasi di luar bencana. Terputusnya komunikasi antara lokasi bencana dan dunia luar dapat menimbulkan beberapa dampak negatif bagi lokasi bencana. Dampak negatif tersebut dapat berupa tidak ada informasi mengenai kondisi korban jiwa, penanganan korban jiwa, total kerusakan bangunan, dan lain sebagainya. Akibatnya, hal ini dapat menciptakan ketidakpastian mengenai kebutuhan penting bagi korban jiwa sehingga dapat menghambat penanganan yang tepat terhadap korban bencana seperti tertundanya bahan bantuan.

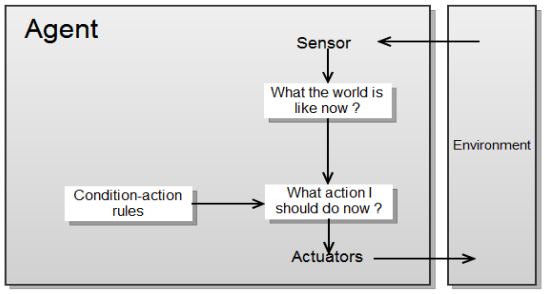
Kondisi ini menjadi pemicu peneliti-peneliti dalam mengembangkan robot sehingga dapat menanggulangi permasalahan putusnya komunikasi akibat suatu bencana. Bidang penelitian terkait permasalahan ini dapat berupa pemilihan jenis robot, penentuan posisi persebaran robot, dan algoritma-algoritma terkait seperti penentuan lokasi robot, pencegahan benturan, dan perluasan jaringan. Dalam *paper*-nya, Takahashi, Sekiyama, dan Fukuda mencoba mensimulasikan algoritma pergerakan koloni robot berdasarkan prioritas keterhubungan robot dengan menara komunikasi [1]. Penelitian lainnya dikembangkan oleh Nulad, dkk., yaitu menggunakan algoritma Position By Line untuk menghubungkan 2 menara komunikasi yang terputus [2].

Kecepatan performa dari sisi waktu pembentukan jaringan merupakan salah satu indikator penting dari setiap algoritma pembentukan jaringan tersebut. Pengukuran waktu performa pun membutuhkan keadaan lingkungan yang sesuai dan hal ini cukup sulit untuk diterapkan pada dunia nyata. Hal ini terjadi karena terkendala dengan pemodelan situasi bencana, jumlah robot untuk membentuk suatu koloni, dan biaya penelitian yang besar. Untuk itu, peneliti mencoba membandingkan beberapa algoritma pembentukan jaringan secara otonom menggunakan perangkat lunak *Open Dynamic Engine*. Alasan pemilihan perangkat lunak ini karena telah mengimplementasikan rumus-rumus fisika seperti gravitasi, momentum, dan gaya gesek. Pemodelan dengan perangkat lunak ini diharapkan dapat meminimalisasi *gap* dengan keadaan dunia nyata.

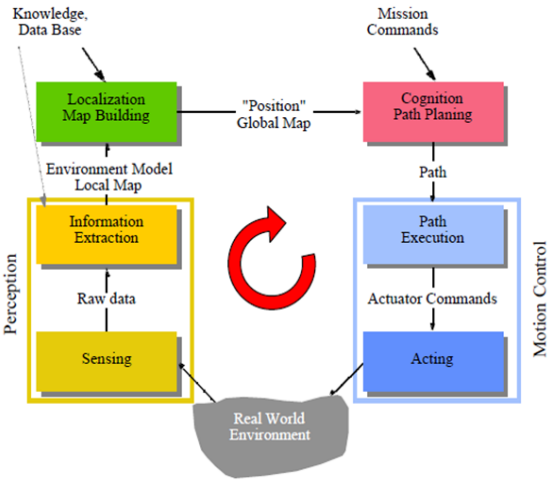
Artificial Intelligence (AI) memiliki empat pengertian utama, yaitu sistem yang berpikir seperti manusia, sistem yang berpikir secara rasional, sistem yang bertindak seperti manusia, dan sistem yang bertindak secara rasional [3]. Pengertian sistem ini kemudian secara umumnya dikenal sebagai *agent*. Dalam dunia AI, suatu *agent*, baik itu *intelligent agent* atau *rational agent*, merupakan aspek terpenting dan diperlukan untuk menjalankan suatu *task* tertentu. Dalam hubungannya dengan lingkungan,*agent* menerima masukan dari lingkungan, kemudian memproses masukan tersebut melalui *sensor*. Hasil masukan tersebut kemudian diolah sehingga menjadi suatu pengetahuan. Pengetahuan ini diasosiasikan dengan respon yang masuk akal melalui *actuators*. Respon ini dinamakan *actions*. Ilustrasi mengenai hal ini dapat dilihat di dalam gambar 1.

Konsep lainnya di dalam AI yang sangat penting adalah mendefinisikan *Performance*, *Actions*, *Goal*, dan *Environmen*t (PAGE). Kaitan antara PAGE dan *agent* diperlukan untuk menggambarkan keterkaitan antara tipe suatu *agent* dengan lingkungan dalam mencapai *target* dan perilaku *agent* tersebut.

Autonomous robot adalah robot yang dapat melakukan pekerjaan dalam lingkungan yang tidak terstruktur tanpa bantuan petunjuk dari manusia [4]. *Autonomous robot* dapat dikategorisasikan menjadi 2 bentuk, yaitu *fully-autonomous robot* dan *semi-autonomous robot* [5]. Sebuah robot dapat dikatakan sepenuhnya otonom jika memiliki kemampuan [4]: mendapatkan informasi mengenai lingkungan, bekerja dalam periode waktu tertentu tanpa interfensi dari luar, dapat bergerak tanpa interfensi manusia, dan menghindari situasi yang berbahaya bagi manusia, properti atau dirinya sendiri.



Gambar 1. Ilustrasi interaksi antara *Agent* dan *Environment* [3].



Gambar 2. *Control scheme* untuk *mobile robot* [5].

Sedangkan, sebuah robot dikatakan *semi-autonomous* karena ketika robot melakukan pengambilan keputusan terdapat interfensi dari luar dalam menentukan keputusan tersebut [5]. *Mobile robot* merupakan mesin otomatis yang dapat bergerak dalam suatu lingkungan [6]. Referensi skema kontrol dari *mobile robot* yang akan digunakan oleh penulis digambarkan pada gambar 2.

Sensing berupa interpretasi terhadap masukan dari *sensor* yang terhubung dengan *mobile robot*. *Information Extraction and Interpretarion* merupakan tahapan pengolahan data mentah berdasarkan keluaran dari *sensing*. Kedua tahapan ini dapat digeneralisasi sebagai tahapan *sensing*. Tahapan ini merupakan tahapan penting karena suatu *mobile robot* harus dapat mengenali lingkungannya agar dapat menyelesaikan pekerjaannya dengan tepat.

Berdasarkan *sensing* dari lingkungan, *mobile robot* memperbaharui pengetahuannya antara lain adalah posisi *mobile robot* baik posisi global maupun posisi lokal terhadap

target. Informasi yang sudah diolah menjadi suatu pengetahuan ini kemudian menjadi masukan untuk menentukan rencana pergerakan *mobile robot* dalam mencapai *target*-nya. Penentuan ini dilakukan pada tahapan *Cognition Path Planning*. Setelah terbentuk keputusan mengenai jalur yang akan diambil maka *mobile robot*akan bergerak sesuai jalur tersebut. Pergerakan ini dilakukan oleh bagian *Motion Control*. Proses ini berulang hingga *target*dari *mobile robot* telah tercapai.

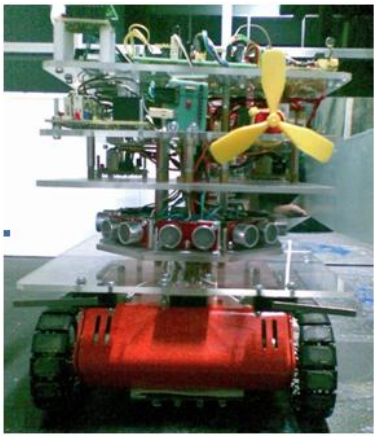
2. Metodologi

Perancangan penelitian yang dibangun diawali dengan memodelkan objek-objek pada *Open Dynamic Engine*. Setelah melakukan pemodelan objek, maka penulis akan mengimplementasikan algoritma perluasan jaringan yaitu algoritma *Position By Line* dan *Self Deployment*. Kemudian, tahapan selanjutnya adalah membandingkan algoritma tersebut dalam sisi waktu pembentukan jaringan komunikasi.

Pada tahap ini penulis akan memodelkan objek-objek yang akan disimulasikan seperti robot dan lingkungannya. Pemodelan robot terdiri atas pemodelan *chassis*, 4 buah roda, *wireless transmitter*/*device*, dan 4 buah *distance sensor*. Sedangkan, pemodelan lingkungan terdiri atas pemodelan rintangan, lantai, dan pemodelan dinding. Setelah pemodelan diselesaikan, maka akan dibangun engsel-engsel yang bersesuaian antar bagian-bagian robot dan juga antara rintangan/dinding dengan lantai. Hasil pada tahapan ini berupa bentuk *3D robot* Al-Fath dalam dunia *Open Dynamic Engine*.

Pada tahap ini akan diimplementasikan algoritma pergerakan robot menuju suatu titik tertentu. Pergerakan robot akan disesuaikan dengan pergerakan robot Al-Fath pada dunia sebenarnya. Selain itu, penulis juga akan mengimplementasikan proses pembentukan koloni untuk menghubungkan antar menara komunikasi. Hasil pada tahapan ini berupa pergerakan robot Al-Fath menuju suatu titik tertentu dan pergerakan koloni robot Al-Fath dalam menghubungkan antar menara komunikasi.

Pemodelan Objek pada *Open Dynamic Engine* dibagi menjadi 2 bagian utama, yaitu pemodelan robot dan pemodelan lingkungan. Robot yang dipergunakan dalam perbandingan algoritma perluasan jaringan ini adalah robot Al-Fath. Robot Al-Fath merupakan robot yang dirakit oleh Universitas Indonesia dengan visualisasi yang dapat dilihat pada gambar 3 [7]. Sedangkan, lingkungan yang dimodelkan berupa dinding dan rintangan berupa kubus dengan lokasi sesuai informasi dari *file* .txt.



Gambar 3. Robot Al-Fath.

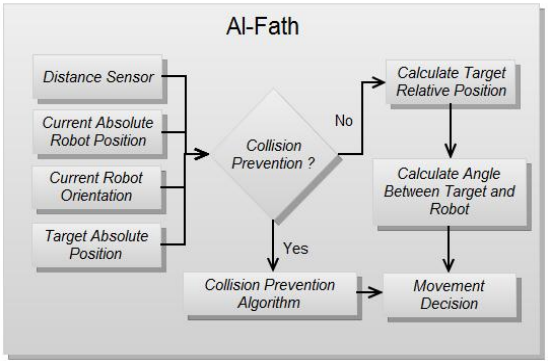
Untuk keperluan simulasi, penulis membagi robot Al-Fath menjadi beberapa komponen, yaitu *chassis*, roda, *distance sensor device*, dan *wireless sensor device*. Sedangkan, pemodelan lingkungan adalah pemodelan lantai, rintangan, dan dinding. Pemodelan lantai pada simulasi diasumsikan datar. Kemudian, pemodelan rintangan dan dinding dengan posisi mengikuti format *file* .txt berukuran *nxn*. Hasil implementasi pemodelan objek pada *Open Dynamic Engine* dapat dilihat pada gambar 4.

Gambar 4. Hasil pemodelan objek pada *Open Dynamic Engine.*

Implementasi algoritma perluasan jaringan pada *Open Dynamic Engine* mengikuti alur seperti pada gambar 5. Penentuan target pada masing-masing robot otonom mengikuti algoritma-algoritma perluasan jaringan. Salah satunya adalah algoritma *Position By Line*.

Pada algoritma *Position By Line* terdapat 2 (dua) fase utama yaitu fase inisialisasi dan fase iterasi. Fase inisialisasi merupakan fase pemberian informasi awal kepada masing-masing robot. Informasi tersebut adalahposisi menara komunikasi, posisi robot, arah robot, dan peta dari lokasi. Peta dari lokasi merupakan

informasi pilihan karena implementasi beberapa algoritma berbeda dengan tanpa adanya informasi ini. Namun, implementasi algoritma tersebut tidak mengurangi konsep algoritma utama pada *Position By Line* ini. Sedangkan, pada fase iterasi adalah fase jalannya simulasi dengan serangkaian proses sesuai skema kontrol robot atau skema pergerakan robot.



Gambar 5. Skema pergerakan robot menuju suatu target.

Penerapan algoritma *Position By Line* dibedakan berdasarkan ada atau tidaknya informasi peta dari lokasi. Ada atau tidaknya peta lokasi dan informasi posisi menara komunikasi menentukan jalur antar menara komunikasi. Pada skenario tidak diketahuinya keadaan lingkungan melalui peta, pembentukan jalur dilakukan dengan menarik garis lurus antar menara komunikasi. Sedangkan ketika informasi peta diketahui, maka jalur yang dibentuk disesuaikan dengan peta dengan melihat apakah ada rintangan atau tidak di suatu titik tertentu. Jalur yang dibentuk pun merupakan jalur terpendek antar menara komunikasi yang hendak dihubungkan.

Setelah ditentukan jalur antar menara komunikasi, hal yang dilakukan selanjutnya adalah penentuan lokasi tujuan robot. Penerapan penentuan lokasi tujuan tanpa adanya informasi peta lokasi dapat diilustrasikan pada gambar 6. Pada mulanya, Al-Fath akan mengolah informasi letak 2 buah menara komunikasi yang hendak dihubungkan menjadi titik-titik posisi target (p1, p2, ..., pn). Masing-masing titik posisi target akan diurutkan berdasarkan jarak posisi tersebut dengan robot. Posisi target terdekatlah yang akan menjadi target utama robot. Persamaan perhitungan titik posisi target ini berdasarkan gambar 6 adalah persamaan 1-4 berikut:

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \qquad (1)$$

$$x_{(p_n)} = \frac{\left(((x_2 - x_1) \times r \times n)\right)}{d} + x_1 \qquad (2)$$

$$y_{(p_n)} = \frac{\left(((y_2 - y_1) \times r \times n)\right)}{d} + y_1 \qquad (3)$$

$$min_{robot} = \left\lceil \frac{d}{r} \right\rceil - 1 \qquad (4)$$

di mana:
(x1, y1) = posisi menara komunikasi *sink*.
(x2, y2) = posisi menara komunikasi target.
d = jarak antara *sink* dan target.
r = *wireless range* dari suatu robot.
n = posisi ke-n.
min_{robot} = jumlah robot minimalyang dibutuhkan.

Sedangkan untuk perhitungan target dengan diketahui peta dalam bentuk *grid*, maka perhitungan dari target tujuan untuk setiap robot akan menjadi sedikit berbeda. Ilustrasi mengenai posisi menara komunikasi dan posisi target berdasarkan jalur yang dibentuk dapat dilihat di dalam gambar 7. Persamaan perhitungan dari target tersebut dapat dilihat pada persamaan 5-9 berikut:

Sistem persamaan lingkaran

$$(x - a)^2 + (y - b)^2 = r^2 \qquad (5)$$

dimana :
(a, b) = titik pusat lingkaran (titik pusat *wireless device*)
r = jari-jari lingkaran (*wireless range*)

Sistem persamaan garis dengan gradien m

$$y = mx + c \qquad (6)$$

Sistem persamaan garis dengan gradien ∞

$$x = c \qquad (7)$$

Perhitungan titik-titik *target* yang diturunkan dari persamaan lingkaran dan garis (persamaan garis dibentuk dari pasangan titik pusat *tile*).

$$\underbrace{(1 + m^2)x^2}_{A} + \underbrace{(-2a + 2mc - 2mb)x}_{B} + \underbrace{(a^2 + c^2 - 2ab + b^2 - r^2)}_{C} = 0 \qquad (8)$$

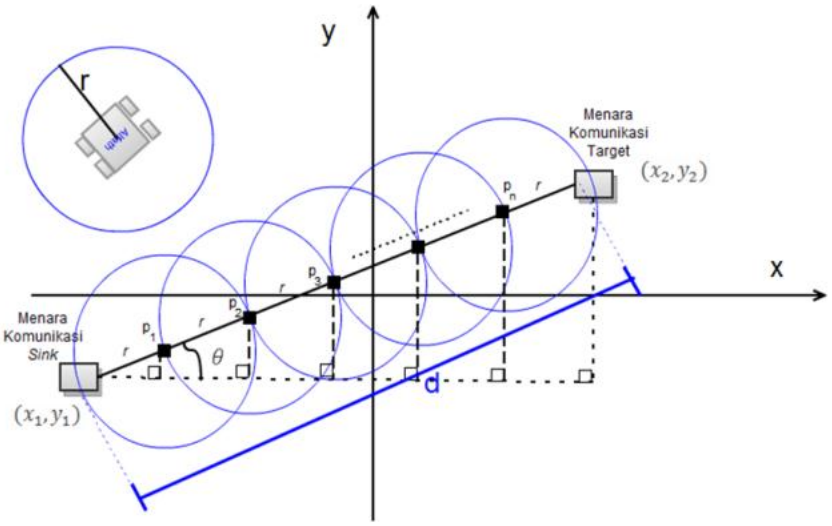
Karena persamaan 8 adalah persamaan kuadrat sehingga dapat dicari solusinya dengan menyubsitusikan nilai A, B, dan C ke persamaan:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \qquad (9)$$

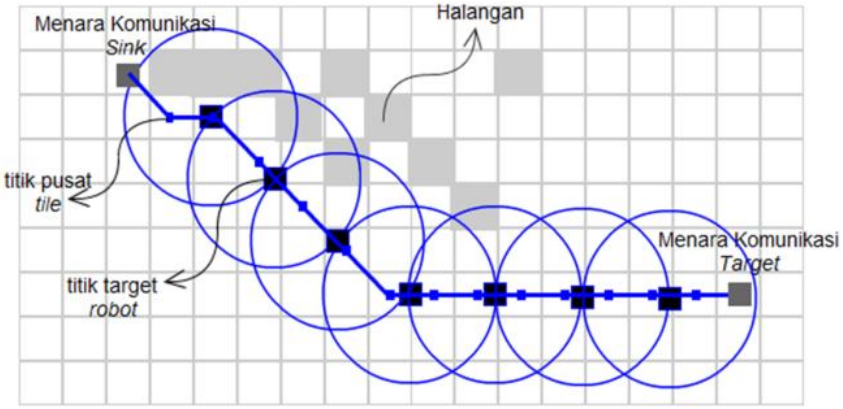
Oleh karena persamaan 9 akan memberikan 2 (dua) buah nilai x, maka diperlukan pengecekan terhadap nilai x yang terletak di antara pasangan titik pusat *tile* yang dipilih. Setelah nilai x telah ditemukan, maka nilai y dapat diperoleh hanya den- gan dengan mensubstitusi nilai x ke persamaan garis awal atau persamaan lingkaran.

Setelah melakukan perhitungan jalur serta posisi *target* pada masing-masing robot, maka robot akan bergerak menuju posisi *target* berdasarkan jalur terpendek ke posisi terdekat. Pada skenario peta yang tidak diketahui, robot akan bergerak berdasarkan garis lurus menuju *target* posisi. Sedangkan pada skenario peta yang diketahui berupa *grid*, robot akan menghitung jalur terpendek menuju posisi yang terdekat. Posisi terdekat dihitung berdasarkan algoritma *flood fill* dan *backtracking* ke semua titik *target* yang diperoleh.

Setelah menempati posisi stabil, terdapat kemungkinan bahwa robot lainnya menuju posisi yang sama dengan robot ini. Oleh karena itu, antar robot diperlukan komunikasi untuk menentukan apakah posisi *target* dari suatu robot telah ditempati oleh robot lain atau tidak. Prinsip komunikasi antar robot pada *Open Dynamic Engine* telah dijelaskan pada bagian *collision detection*. Ketika suatu robot menemukan bahwa lokasi yang ditujunya sudah ditempati robot lain, maka robot akan menuju target lainnya yang terdekat dengan posisi robot saat itu. Penentuan posisi terdekat ditentukan dari jarak robot dengan posisi target berikutnya, tetapi pada kondisi peta *grid* yang diketahui, jarak terdekat ditentukan dari algoritma *flood fill*. Implementasi Algoritma Perluasan: Agoritma *Self Deployment.* Penerapan algoritma *Self Deployment* berbeda dengan algoritma *Position By Line* dalam hal strategi pembentukan jaringan. Pada algoritma *Position By Line*, target posisi robot telah ditentukan ketika inisialisasi atau awal proses simulasi. Sedangkan penentuan target pada algoritma *Self Deployment* ditentukan sepanjang simulasi terjadi. *Pseudocode* penerapan algoritma ini pada *Open Dynamic Engine* adalah sebagai berikut. Pada mulanya, robot akan menuju prioritas utama yaitu *node* yang terdekat dengan *sink*. Jika tidak terdapat *node* yang terdekat dengan *sink*, maka robot akan menuju *sink*.



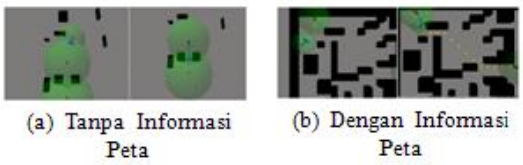
Gambar 6. Ilustrasi target robot tanpa informasi peta lokasi.



Gambar 7. Ilustrasi target robot dengan informasi peta lokasi.

Setelah robot mencapai *sink* dalam jarak tertentu, maka robot akan mengubah *target* ke prioritas utama berikutnya yaitu *node* yang terdekat dengan target. Jika tidak ditemukan *node* yang terdekat dengan target, maka robot akan menuju menara komunikasi target. Setelah mencapai jarak *threshold* untuk komunikasi *wireless*, maka robot akan berhenti dan menyebarkan kepada tetangganya bahwa ia merupakan *node* dengan prioritas tertinggi yang dekat dengan target. Sepanjang simulasi, terdapat kemungkinan bahwa robot bertemu dengan robot lainnya yang telah berada dalam posisi stabil. Ketika hal ini terjadi, maka robot akan berkomunikasi dengan robot tersebut mengenai *sink*, target, dan *node* terdekat dengan target. Jika ternyata informasi ini sesuai dengan keadaan robot yang bertanya (*sink* dan target), maka robot tersebut akan langsung bergerak menuju prioritas utama berikutnya yaitu*node* terdekat dengan target. Sepanjang simulasi, terdapat kemungkinan bahwa robot mendapatkan informasi dari *distance sensor* mengenai rintangan. Untuk itu, robot akan menjadikan algoritma pencegahan benturan sebagai prioritas utama.

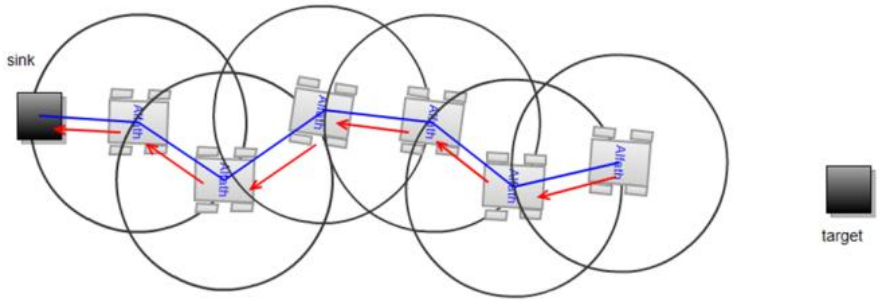
Penentuan *sink* dan target dari suatu robot diperoleh dengan memperhatikan jarak terpendek yang dibentuk antara robot dengan target dan robot dengan *sink* ketika proses inisialisasi. Menara komunikasi dengan jarak terpendek ke robot akan menjadi *sink* dan menara komunikasi lainnya akan menjadi target. Sedangkan jika dilihat dari sisi pergerakannya, algoritma *Self Deployment* ini juga memiliki perbedaan dalam penerapannya bergantung dari ada atau tidaknya informasi peta lokasi. Dengan tidak adanya informasi peta lokasi, pergerakan robot akan menuju target dalam bentuk garis lurus. Namun, jika informasi peta *grid* diketahui, robot akan bergerak menuju suatu target dengan jalur terpendek. Kedua hal ini serupa pada penerapan algoritma *Position By Line*. Hasil simulasi pergerakan ini dapat dilihat pada gambar 8.



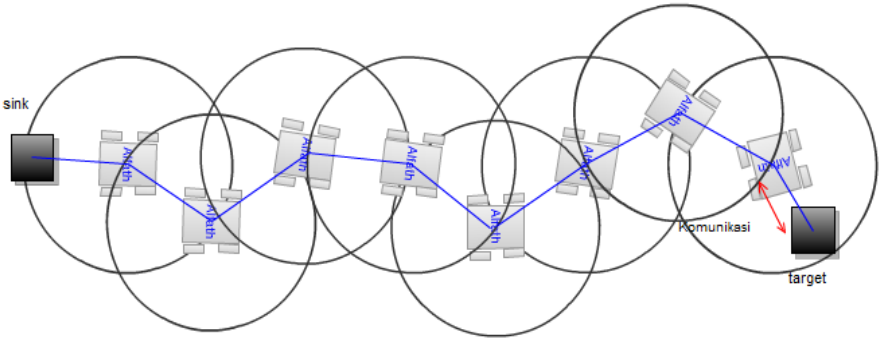
Gambar 8. Implementasi algoritmaTakahashi, Sekiyama, dan Fukuda pada *Open Dynamic Engine*.

Pemberitahuan oleh robot ketika telah menempati posisi stabil dilakukan secara rekursif terhadap seluruh *node* yang berada dalam posisi stabil. Ilustrasi ini dapat dilihat pada gambar 9.Terdapat 2 (dua) skenario pada algoritma sehingga suatu menara komunikasi dapat dikatakan terhubung satu sama lain. Pada skenario pertama, robot berusaha menghubungkan antara menara komunikasi *sink* menuju menara komunikasi target, tanpa ada robot lain yang menghubungkan dari menara komunikasi sebaliknya. Pada skenario ini, suatu menara dianggap telah terkoneksi ketika robot terakhir pada prioritas telah terhubung dengan menara komunikasitarget. Ilustrasi gambar dapat dilihat pada gambar 10.

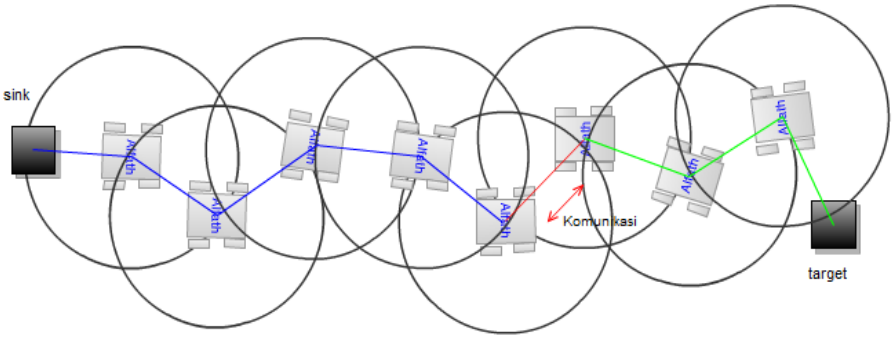
Sedangkan skenario berikutnya adalah ketika terdapat beberapa robot yang menghubungkan menara komunikasi antara*sink* dan target serta beberapa robot lainnya yang menghubungkan menara komunikasi antaratarget dan *sink*. Akibatnya, terdapat kemungkinan bahwa robot-robot tersebut bertemu pada suatu titik diantara menara komunikasi *sink* dan target.



Gambar 9. Proses pemberitahuan posisi robot.



Gambar 10. Skenario pertama terkait keterhubungan antara menara komunikasi pada algoritma *Self Deployment*.



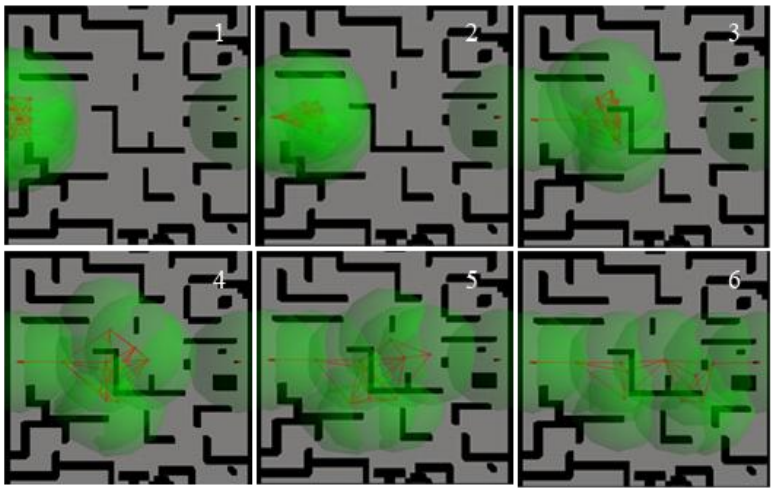
Gambar 11. Skenario kedua terkait keterhubungan antara menara komunikasi pada algoritma *Self Deployment*.

Dalam skenario ini, antar robot akan berkomunikasi untuk menyatakan bahwa menara komunikasi telah terhubung dan siap melakukan pertukaran informasi antar menara komunikasi. Ilustrasi gambar dari skenario ini dapat dilihat pada gambar 11.

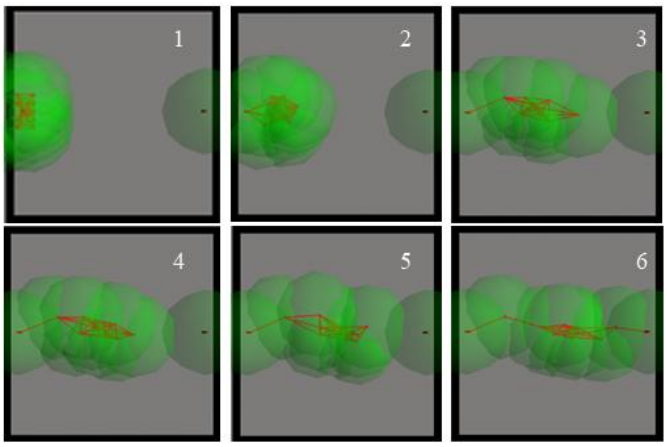
3. Hasil dan Analisis

Desain dan Skenario Uji Coba. Proses uji coba yang dilakukan terdiri dari beberapa skenario yang merupakan kombinasi dari persebaran robot (tersebar atau terpusat dekat menara komunikasi), ada atau tidaknya rintangan, dan pemberian informasi peta. Oleh karena skenario dibentuk dari kombinasi ketiga hal tersebut, maka total skenario adalah 8 buah skenario. Masing-masing skenario dilakukan iterasi sebanyak 10 kali dan dihitung nilai rata-rata dari setiap iterasi tersebut.

Hasil dari sebuah uji coba suatu skenario adalah jumlah robot yang dibutuhkan serta waktu pembentukan jaringan komunikasi stabil antar menara komunikasi. Uji coba seluruh skenario dari setiap algoritma akan dilakukan pada komputer dengan spesifikasi sebagai berikut: *Processor* Intel(R) Core(TM) i5-2400 CPU @3.10Ghz; *Installed memory* (RAM) 2.99 GB; *Operating System* Windows 7 Professional.



Gambar 12. Algoritma *Position By Line*pada skenario keempat.



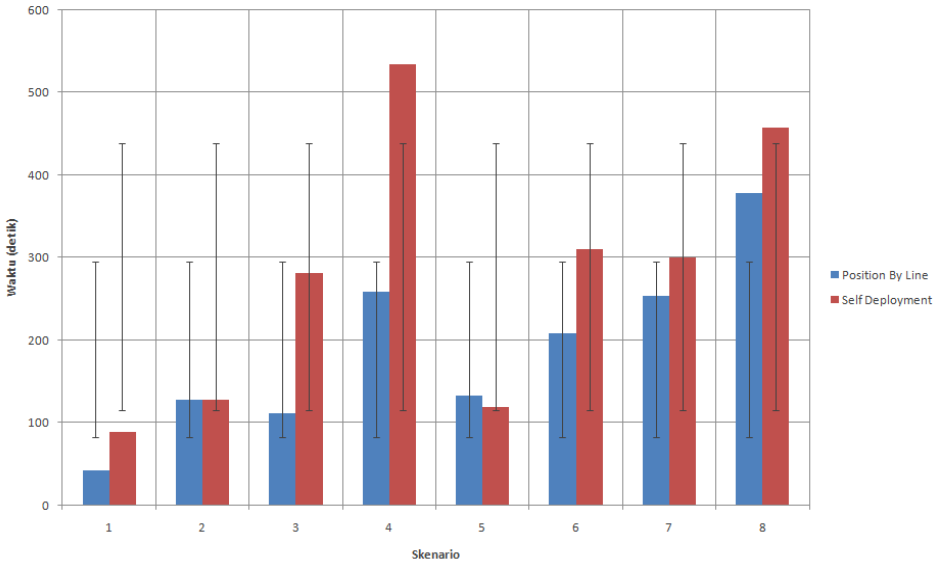
Gambar 13. Algoritma *Self Deployment*pada skenario kedua.

Tabel hasil uji coba algoritma *Position By Line* pada *Open Dynamic Engine* dapat dilihat pada tabel I. Sedangkan, salah satu contoh *screenshot* uji coba dapat dilihat pada gambar 12.

Tabel hasil uji coba algoritma *Self Deployment* pada *Open Dynamic Engine* dapat dilihat pada tabel II. Sedangkan, salah satu contoh *screenshot* uji coba dapat dilihat pada gambar 13.

Hasil perbandingan algoritma *Position By Line* dan *Self Deployment* dapat diperlihatkan pada gambar 14. Berdasarkan gambar 14, dapat dilihat bahwa dari ke-8 skenario yang dibentuk, algoritma *Position By Line* hampir unggul disetiap skenario. Hipotesis yang sama juga diperlihatkan dengan data standar deviasi-nya. Namun, untuk lebih memastikan, penulis melakukan uji statistik dengan uji Mann-Whitney U dari keseluruhan data untuk menguji hipotesa ini. Pengujian dengan tipe statistik ini dipilih karena data tersebar dengan tidak terdistribusi normal. Pengujian data terdistribusi normal atau tidaknya menggunakan uji Kolmogorov-Smirnov.

Uji Kolmogorov merupakan jenis uji normalisasi yang berfungsi untuk menguji apakah hipotesis nol merupakan suatu distribusi tertentu atau tidak. Distribusi yang hendak diuji adalah distribusi normal agar kemudian dapat ditentukan uji statistik yang tepat berdasarkan distribusinya. Penulis menentukan hipotesis sebagai berikut, Ho sebagai data terdistribusi normal, dan H1 sebagai data tidak terdistribusi normal. Sedangkan, nilai signifikansi (α) yang dipilih adalah $\alpha = 0.0$. Hasil uji coba ini dapat dilihat pada tabel III dan IV



Gambar 14. Perbandingan waktu rata-rata setiap skenario antara algoritma *Position By Line* dan *Self Deployment*.

TABEL I
HASIL UJI COBA SKENARIO PADA ALGORITMA *POSITION BY LINE*

Skenario ke-	Jumlah Robot pada Simulasi	Jumlah Robot yang Dibutuhkan	Waktu Rata-Rata	Standar Deviasi
1	9	4	42.0722	0.687
2	9	4	126.9623	5.458
3	9	4	111.385	1.071
4	9	4	258.575	2.542
5	9	4	131.8877	8.242
6	9	4	208.1863	1.321
7	9	4	253.1318	3.107
8	9	4	377.8814	2.700

TABEL II
HASIL UJI COBA SKENARIO PADA ALGORITMA *SELF DEPLOYMENT*

Skenario ke-	Jumlah Robot pada Simulasi	Jumlah Robot yang Dibutuhkan	Waktu Rata- Rata	Standar Deviasi
1	9	4	88.420	0.287
2	9	4	127.673	0.787
3	9	4	281.598	3.568
4	9	6	534.60	4.821
5	9	4	118.547	1.244
6	9	4	310.149	0.937
7	9	4	299.530	3.697
8	9	4	457.490	3.290

Karena nilai *sig* pada kedua hasil pada kedua tabel kurang dari 0.05 sehingga kesimpulan yang diperoleh adalah Ho ditolak. Kesimpulan bahwa nilai Ho ditolak menunjukan bahwa distribusi dari data percobaan tidak normal sehingga uji statistik yang tepat digunakan adalah uji non-*parametrik* yaitu Uji Mann-Whitney U. Uji Mann-Whitney U ini dipilih karena sebelumnya diperoleh hasil pengujian terhadap data yang tidak terdistribusi normal.

TABEL III
HASIL UJI KOLMOGOROV-SMIRNOV ALGORITMA POSITION BY LINE

Position By Line		
N		80
Normal Parameters[a]	Mean	188.7602
	Std. Deviation	100.52509
Most Extreme Differences	Absolute	0.170
	Positive	0.170
	Negative	-0.103
Kolmogorov-Smirnov Z		1.519
Asymp. Sig. (2-tailed)		0.020

TABEL IV
HASIL UJI KOLMOGOROV-SMIRNOV ALGORITMA SELF DEPLOYMENT

Self Deployment		
N		80
Normal Parameters[a]	Mean	277.2520
	Std. Deviation	152.44411
Most Extreme Differences	Absolute	0.210
	Positive	0.210
	Negative	-0.126
Kolmogorov-Smirnov Z		1.875
Asymp. Sig. (2-tailed)		0.002

TABEL V
HASIL UJI MANN-WHITNEY – TEST STATISTICS

Group	N	Mean Rank	Sum of Ranks
Position by line	80	66.66	5333.00
Self Deployment	80	94.34	7547.00
Total	160		

TABEL VI
HASIL UJI MANN-WHITNEY - RANKS

	Value
Mann-Whitney U	2093.0000
Wilcoxon W	5333.0000
Z	-3.778
Asymp. Sig. (2-tailed)	0.000

Selanjutnya, penulis menentukan hipotesis awal adalah sebagai berikut: Ho dengan tidak terdapat perbedaan rata-rata antara Algoritma *Position By Line* dan *Self Deployment* dan H1 dengan terdapat perbedaan rata-rata antara Algoritma *Position By Line* dan *Self Deployment*. Selanjutnya, nilai α yang dipilih yaitu 0.05. Hasil uji Mann-Whitney U diperlihatkan dengan tabel V dan VI.

Hasil pada tabel V dan VI memperlihatkan bahwa Z bernilai $-3.778 < 0.05(\alpha)$ sehingga dapat diperoleh kesimpulan bahwa Ho ditolak dan H1 diterima. Kesimpulan dari hasil uji statistik ini adalah terdapat perbedaan rata-rata antara kedua algoritma. Penentuan algoritma yang lebih unggul dapat diperlihatkan dengan membandingkan data *mean rank*. Seperti yang diperlihatkan dalam tabel V dan VI, bahwa nilai *mean rank* dari algoritma *Position By Line* lebih kecil daripada Self Deployment sehingga dapat diperoleh kesimpulan bahwa algoritma *Position By Line*lebih baik dibandingkan dengan algoritma *Self Deployment*.

4. Kesimpulan

Kesimpulan dari penelitian ini adalah sebagai berikut. Pertama, algoritma *Position By Line* lebih baik hampir dalam setiap skenario dibandingkan dengan algoritma *Self Deployment* dilihat dari sisi waktu robot mencapai posisi stabil. Kedua, penggunaan *Autonomous Mobile* robot dapat menjadi solusi alternatif dalam menghubungkan menara komunikasi yang terputus berdasarkan simulasi bencana. Ketiga, proses pertukaran komunikasi antar robot sebagai bentuk koordinasi dalam mencapai formasi yang sesuai diperlukan oleh setiap algoritma yang diujicobakan (*Position By Line* dan *Self Deployment*). Kesimpulan terakhir, persebaran posisi robot, posisi rintangan, serta penggunaan informasi peta bagi setiap robot mempengaruhi waktu pembentukan jaringan antar menara komunikasi.

Rerefensi

[1] J. Takahashi, K. Sekiyama, &T. Fukuda,"Self- deployment algorithm for mobile sensor network based on connection priority criteria withobstacle avoidance"*In IEEE International Conference*, pp. 1434–1439, 2007.

[2] N. W. Prambudi, "Laporan kemajuan penelitian Position By Line dan Extended By Line", *Technical report*, Fakultas Ilmu Komputer, Universitas Indonesia, 2009.

[3] S. Russell&P. Norvig, *Artificial Intelligence, A Modern Approach*,2nds Edition, Pearson Education, Inc., New Jersey, 2003.

[4] G.A. Bekey,*Autonomous Robot, From Biological Inspiration to Implementation and Control*, The MIT Press, Massachusetts, 2005.

[5] R. Siegwart&I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, Massachusetts Institute of Technology, Massachusetts, 2004.

[6] F. Baras,"Design of a Mars Rover Suspension Mechanism," Ph.D Thesis, Department of Mechanical Engineering, Izmir Institute of Technology, 2004.

[7] M.S. Alvissalim& F. Jovan, Hardware Prototype untuk Pengembangan Swarm Robot untuk Mendeteksi Kebocoran Gas dan Mendeteksi Bom di Gedung/Pabrik di Indonesia,*Technical Report*, Fakultas Ilmu Komputer, Depok, Universitas Indonesia, 2009.