



UNIVERSITAS INDONESIA

**PEMBUATAN SIMULATOR 3D DENGAN
OPEN DYNAMICS ENGINE DAN
PENAMBAHAN *DYNAMIC NICHE* PADA ALGORITMA
MODIFIED PARTICLE SWARM OPTIMIZATION UNTUK
PENCARIAN BANYAK SUMBER ASAP**

TESIS

**WULUNG PAMBUKO
0706193435**

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI MAGISTER ILMU KOMPUTER**

**DEPOK
JULI 2009**

**PERPUSTAKAAN
UNIVERSITAS INDONESIA**



UNIVERSITAS INDONESIA

**PEMBUATAN SIMULATOR 3D DENGAN
OPEN DYNAMICS ENGINE DAN
PENAMBAHAN *DYNAMIC NICHE* PADA ALGORITMA
MODIFIED PARTICLE SWARM OPTIMIZATION UNTUK
PENCARIAN BANYAK SUMBER ASAP**

TESIS

**Diajukan sebagai salah satu syarat untuk memperoleh gelar
Magister Ilmu Komputer**

**WULUNG PAMBUKO
0706193435**

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI MAGISTER ILMU KOMPUTER**

**DEPOK
JULI 2009**



HALAMAN PERNYATAAN ORISINALITAS

Tesis ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun yang dirujuk
telah saya nyatakan dengan benar

Nama : WULUNG PAMBUKO

NPM : 0706193435

Tanda Tangan : 

Tanggal : 31 JULI 2009

HALAMAN PENGESAHAN

Tesis ini diajukan oleh :

Nama : Wulung Pambuko
NPM : 0706193435
Program Studi : Magister Ilmu Komputer
Judul Tesis : Pembuatan Simulator 3D dengan *Open Dynamics Engine* dan Penambahan *Dynamic Niche* pada Algoritma *Modified Particle Swarm Optimization* untuk Pencarian Banyak Sumber Asap

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Magister Ilmu Komputer pada Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Ir. Wisnu Jatmiko, M.Kom., Dr. Eng. (

Penguji : Dr. Ir. Petrus Mursanto, M.Sc. (

Penguji : Setiadi Yazid, Ph.D (

Penguji : Ir. Bob Hardian Syahbuddin, Ph.D. (

Ditetapkan di : Depok

Tanggal : Juli 2009

KATA PENGANTAR

Penulis mengucapkan puji dan syukur kehadirat Allah SWT, dengan izin-Nya penulis dapat menyelesaikan kegiatan penelitian dan penyusunan laporan tugas akhir dengan judul: “Pembuatan Simulator 3D dengan *Open Dynamics Engine* dan Penambahan *Dynamic Niche* pada Algoritma *Modified Particle Swarm Optimization* untuk Pencarian Banyak Sumber Asap” ini.

Pada kesempatan ini, penulis ingin menyampaikan terima kasih yang kepada semua pihak yang telah ikut serta memberikan dukungan serta bantuan mencakup dorongan semangat dan moral, sehingga akhirnya kegiatan penelitian tugas akhir ini dapat berjalan lancar seperti sebagaimana mestinya. Penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa, Allah SWT atas rahmat dan hidayahnya kepada penulis.
2. Kedua orang tua penulis, Ayahanda Boediono Soedirman dan Ibunda Harina Yuhetty, yang selalu menyertai langkah penulis dalam setiap do'a mereka.
3. Bapak Ir. Wisnu Jatmiko, M.Kom., Dr.Eng., selaku pembimbing tugas akhir yang telah mengarahkan penulis dalam melaksanakan kegiatan penelitian ini hingga proses penyusunan laporan.
4. Dr. Ir. Petrus Mursanto M.Sc., selaku dosen penguji pertama pada sidang penelitian tugas akhir ini.
5. Setiadi Yazid Ph.D., selaku dosen penguji kedua pada sidang penelitian tugas akhir ini.
6. Ir. Bob Hardian Syahbuddin Ph.D., selaku dosen penguji ketiga pada sidang penelitian tugas akhir ini.
7. Bapak DR. M. Rahmat Widyanto dan Bapak DR. Hisar Maruli Manurung, selaku Pembimbing Akademik yang telah memberikan arahan kepada penulis selama masa studi di Fakultas Ilmu Komputer, Universitas Indonesia..
8. Bapak Drs. R. Yugo Kartono Isal M.Sc., yang telah mengajarkan konsep

struktur data dan desain algoritma, hingga penulis memiliki pengetahuan yang dapat diterapkan untuk menyelesaikan penelitian tugas akhir.

9. Melur Pinilih, adik penulis yang telah ikut memberikan dukungan moral dan dorongan semangat kepada penulis.
10. Rekan-rekan seperjuangan di Lab 3310 Robotics, yang telah berbagi pengetahuan dan pengalaman selama kegiatan penelitian tugas akhir.
11. Teman-teman seangkatan yang telah berjuang bersama penulis dalam suka dan duka selama menjalani dua tahun masa pendidikan di Fasilkom UI. Terima kasih untuk warna-warna yang tidak akan pernah penulis lupakan.
12. Dosen, staf, mahasiswa, dan seluruh keluarga besar Fasilkom UI yang namanya tidak bisa penulis sebutkan satu per satu di sini, terima kasih atas segala bentuk bantuan dan dukungannya.
13. Shinobu Izumi dan Yurie Izumi yang selalu menyemangati penulis di saat-saat senang dan susah.

Penulis menyadari masih terdapat kekurangan pada penyusunan laporan ini. Oleh karena itu, penulis dengan tangan terbuka bersedia menerima kritik dan saran yang berguna. Semoga karya ini bermanfaat bagi pembaca semua.

Depok, Agustus 2008 – Juli 2009

Penulis

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS
AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Wulung Pambuko
NPM : 0706193435
Program Studi : Magister Ilmu Komputer
Fakultas : Ilmu Komputer
Jenis Karya : Tesis

demı pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (*No-exclusive Royalty-Free Right*) atas karya ilmiah saya yang berjudul:

“Pembuatan Simulator 3D dengan *Open Dynamics Engine* dan Penambahan *Dynamic Niche* pada Algoritma *Modified Particle Swarm Optimization* untuk Pencarian Banyak Sumber Asap”

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia / format, mengelola dalam bentuk pangkalan data (*database*), merawat dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis / pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 31 Juli 2009

Yang menyatakan,



(Wulung Pambuko)

ABSTRAK

Nama : Wulung Pambuko
Program Studi : Magister Ilmu Komputer
Judul : Pembuatan Simulator 3D dengan *Open Dynamics Engine* dan Penambahan *Dynamic Niche* pada Algoritma *Modified Particle Swarm Optimization* untuk Pencarian Banyak Sumber Asap

Isi tesis ini mengenai pembuatan simulator 3D dari algoritma pencarian *Particle Swarm Optimization* untuk pencarian banyak sumber asap dengan menggunakan *Open Dynamics Engine* dan mengenai *Dynamic Niche-PSO* yang adalah algoritma baru sebagai modifikasi algoritma MPSO dari penelitian sebelumnya [1]. Versi simulator 2D untuk PSO ini telah dibuat penelitian sebelumnya ini. Pemodelan fisik 3D ini bertujuan untuk mengurangi gap antara perangkat lunak dan perangkat keras di dunia nyata.

Salah satu bab adalah bab yang menjelaskan pembuatan model robot, asap dan sumbernya, dan medan dengan *Open Dynamics Engine*. Dilanjutkan dengan bab tentang cara pemakaian GUI simulator ini.

Algoritma *Dynamic Niche-PSO* yang diajukan pada penelitian ini bertujuan untuk memperbaiki kelemahan algoritma PSO sebelumnya dimana 2 *niche* (kelompok agen) atau lebih masih ada kemungkinan untuk menuju sumber asap yang sama. Pada *Dynamic Niche-PSO* ini diperkenalkan robot baru, yaitu robot utama yang mempunyai arca ketertarikan. Pada *Dynamic Niche-PSO* ini juga robot netral dan bermuatan dapat berpindah keanggotaan dari satu *niche* ke *niche* yang lain apabila memasuki area ketertarikan atau *attract area* dari robot utama *niche* yang lain ini.

Kata kunci:

Particle Swarm Optimization, Open Dynamics Engine, OpenGL

ABSTRACT

Name : Wulung Pambuko
Study Program : Magister Computer Science
Title : The Development of 3D Simulator by Open Dynamics Engine and The Additional Dynamic Niche to The Modified Particle Swarm Optimization Algorithm for Multi Odor Source Localization

The contents of this thesis are the development of 3D simulator for visualizing Particle Swarm Optimization algorithm for multi odor source localization using Open Dynamics Engine, and Dynamic Niche-PSO as modification of MPSO algorithm from previous research [1]. The 2D version of this MPSO is made in previous research. This 3D modeling has a purpose to reduce gap between software and hardware in the real world.

One of chapters is explaining about how to make the model of robots, plumes and its sources, and field with Open Dynamics Engine. Continued with chapter explaining about how to use the GUI of this simulator.

Dynamic Niche-PSO algorithm proposed in this research has a purpose to refine the weakness of previous algorithm where 2 niches (group of agents) or more still have a probability to move toward the same odor source. There is newly introduced robot in this Dynamic Niche-PSO algorithm called main robot which has an attract area. In this Dynamic Niche-PSO also a neutral robot or a charge robot could become a member of another niche if it entered the attract area of main robot of this other niche.

Keywords:

Particle Swarm Optimization, Open Dynamics Engine, OpenGL

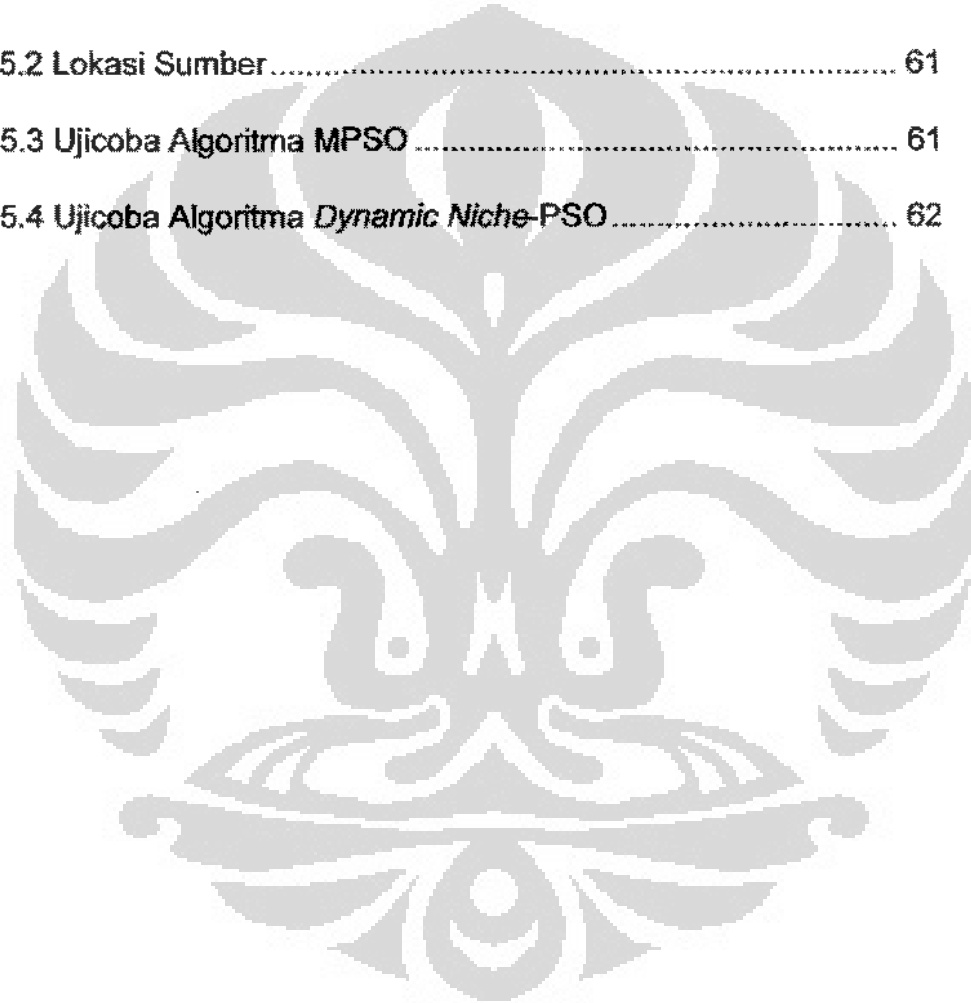
DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS	II
HALAMAN PENGESAHAN	III
KATA PENGANTAR	IV
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS	VI
ABSTRAK.....	VII
ABSTRACT.....	VIII
DAFTAR ISI.....	IX
DAFTAR GAMBAR.....	XII
BAB 1 PENDAHULUAN.....	1
1.1 LATAR BELAKANG.....	1
1.2 RUMUSAN MASALAH	4
1.3 TUJUAN PENELITIAN	5
1.4 RUANG LINGKUP PENELITIAN	5
1.5 TAHAPAN PENELITIAN.....	5
1.6 SISTEMATIKA PENULISAN LAPORAN.....	6
BAB 2 PEMODELAN ODE.....	8
2.1 PEMODELAN ROBOT.....	8
2.2 PEMODELAN MEDAN.....	16
2.3 PEMODELAN ASAP	18
2.4 PEMODELAN SUMBER ASAP.....	20
BAB 3 PETUNJUK PEMAKAIAN SIMULATOR	22
3.1 ANTAR MUKA.....	22
3.2 MENJALANKAN SIMULATOR	24
3.3 MENU SIMULATOR	27
3.4 TAMPILAN ANIMASI SIMULASI.....	30
3.5 TAMPILAN INFORMASI SIMULATOR	31
3.6 PENGATURAN KAMERA.....	32
3.7 PENGATURAN LINGKUNGAN	36
3.8 PENGATURAN SUMBER ASAP.....	37

3.9 PENGATURAN ANGIN	39
3.10 PENGATURAN ROBOT NETRAL	40
3.11 PENGATURAN ROBOT BERMUATAN	43
3.12 PENGATURAN KELOMPOK PENCARIAN	44
3.13 PENGATURAN ROBOT UTAMA	45
3.14 PENGATURAN RASIO KESALAHAN SENSOR	45
BAB 4 DYNAMIC NICHE-PSO.....	47
4.1 ALGORITMA ORIGINAL	47
4.2 KELEMAHAN ALGORITMA ORIGINAL.....	49
4.3 DYNAMIC NICHE-PSO	51
4.3.1 Robot Utama	51
4.3.2 Pengaruh Perpindahan Keanggotaan	56
4.4 HASIL.....	58
4.5 ANALISIS	59
BAB 5 PENUTUP.....	63
5.1 KESIMPULAN.....	63
5.2 SARAN	63
DAFTAR PUSTAKA.....	65

DAFTAR TABEL

Tabel 2.1.1 Penetapan <i>Joint</i>	11
Tabel 2.1.2 Penetapan Warna Robot	14
Tabel 4.3.1 Keanggotaan Niche (Sebelum Berpindah).....	53
Tabel 4.3.2 Keanggotaan Niche (Sesudah Berpindah).....	53
Tabel 4.5.1 Jumlah Robot/ <i>Niche</i>	60
Tabel 4.5.2 Lokasi Sumber.....	61
Tabel 4.5.3 Ujicoba Algoritma MPSO	61
Tabel 4.5.4 Ujicoba Algoritma <i>Dynamic Niche</i> -PSO.....	62



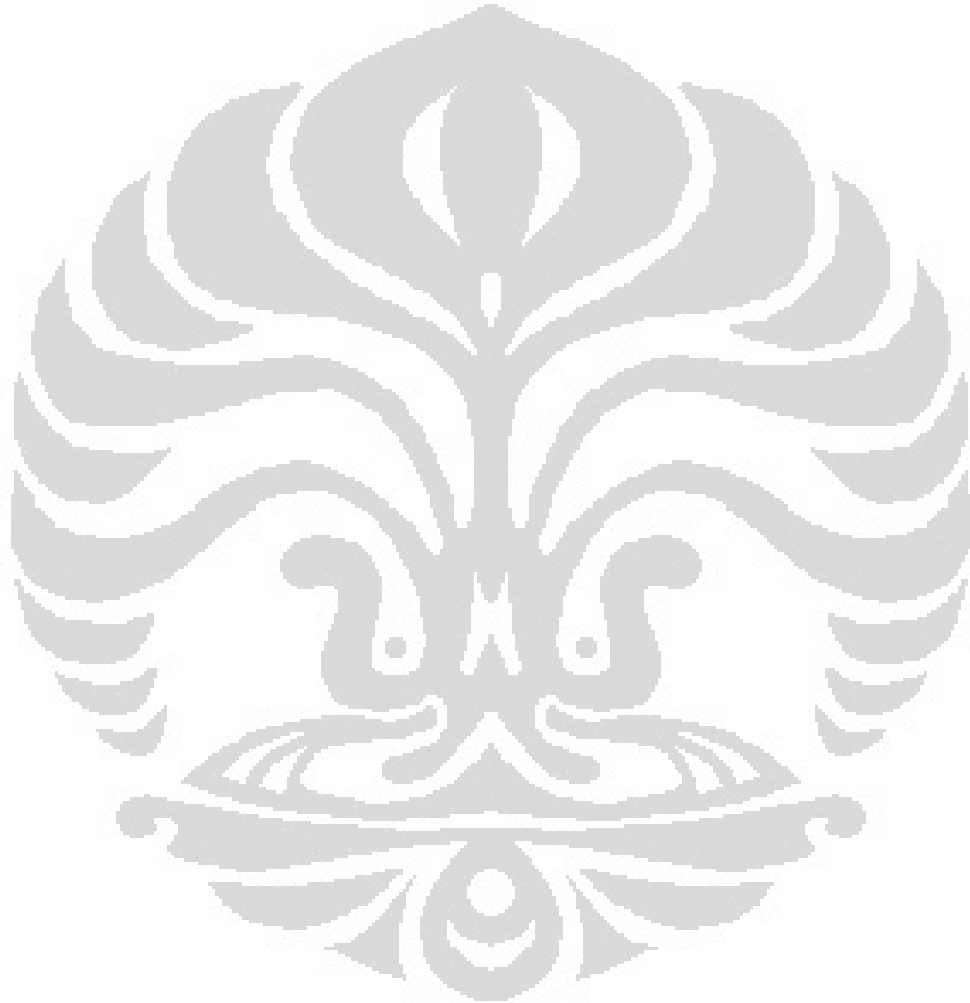
DAFTAR GAMBAR

Gambar 1.1.1 Simulator 2D Niche-PSO	3
Gambar 1.1.2 Perputaran robot.....	4
Gambar 2.1.1 Geometri Robot.....	8
Gambar 2.1.2 Robot Bermuatan.....	15
Gambar 2.2.1 Dinding.....	18
Gambar 2.3.1 Asap.....	19
Gambar 2.4.1 Sumber Asap (Sebelum ditemukan)	21
Gambar 2.4.2 Sumber Asap (Setelah ditemukan)	21
Gambar 3.1.1 GUI Simulator	22
Gambar 3.2.1 Penambahan Sumber Asap.....	25
Gambar 3.2.2 Tombol "Pause"	26
Gambar 3.2.3 Simulasi	26
Gambar 3.3.1 Menu Simulator.....	27
Gambar 3.3.2 Bagian dari Data.txt	29
Gambar 3.4.1 Ruang Pencarian	30
Gambar 3.4.2 Robot Netral dan Robot Bermuatan.....	31
Gambar 3.5.1 Informasi Simulasi.....	32
Gambar 3.6.1 Hasil Pengaturan Kamera.....	33
Gambar 3.6.2 Rotasi Vertikal Pandangan	34
Gambar 3.6.3 Rotasi Pandangan	34
Gambar 3.6.4 Pergeseran Horizontal Pandangan.....	35
Gambar 3.6.5 Pergeseran Pandangan.....	36

Gambar 3.7.1 Pengaturan Lingkungan.....	37
Gambar 3.8.1 Pengaturan Sumber Asap.....	37
Gambar 3.8.2 Animasi Asap.....	38
Gambar 3.8.3 Form Penambahan Asap.....	38
Gambar 3.9.1 Pengaturan Angin.....	39
Gambar 3.10.1 Pengaturan Robot Netral.....	41
Gambar 3.11.1 Pengaturan Robot Bermuatan.....	43
Gambar 3.12.1 Pengaturan Kelompok Pencarian.....	44
Gambar 3.13.1.....	45
Gambar 3.14.1 Pengaturan Rasio Kesalahan Sensor.....	46
Gambar 4.1.1 Pergerakan dalam PSO.....	47
Gambar 4.1.2 Robot Bermuatan.....	48
Gambar 4.2.1 PSO Algoritma Original (Keadaan Awal).....	50
Gambar 4.2.2 PSO Algoritma Original (Hasil).....	50
Gambar 4.3.1 Robot Utama.....	51
Gambar 4.3.2 Perpindahan Keanggotaan.....	52
Gambar 4.3.3 Robot Utama – R Core.....	54
Gambar 4.3.4 Gaya Tolak Robot Utama.....	55
Gambar 4.3.5 Pengaruh Perpindahan Keanggotaan (Keadaan awal).....	56
Gambar 4.3.6 Pengaruh Perpindahan Keanggotaan (Hasil).....	57
Gambar 4.4.1 Dynamic Niche-PSO (Keadaan Awal).....	58
Gambar 4.4.2 Dynamic Niche-PSO (Hasil).....	59
Gambar 4.5.1 Susunan Robot (MPSO).....	60

Gambar 4.5.2 Susunan Robot (*Dynamic Niche-PSO, 2 niche*) 60

Gambar 4.5.3 Susunan Robot (*Dynamic Niche-PSO, 5 niche*) 60



DAFTAR PROGRAM

Program 2.1.1 Definisi Geometri Robot.....	9
Program 2.1.2 Pemodelan Chassis Robot	9
Program 2.1.3 Pemodelan Upper Robot	9
Program 2.1.4 Pemodelan Wheel Robot.....	10
Program 2.1.5 Pemodelan Help Wheel Robot.....	10
Program 2.1.6 Deklarasi Variabel Bagian Badan Robot.....	11
Program 2.1.7 Chassis - Upper.....	12
Program 2.1.8 Chassis - Wheel.....	12
Program 2.1.9 Chassis – Help Wheel	12
Program 2.1.10 Ruang Robot.....	13
Program 2.1.11 Penggambaran Robot (Robot Netral)	14
Program 2.1.12 Penggambaran Area Ketertarikan	15
Program 2.2.1 <i>Constructor</i> Kelas Wall	16
Program 2.2.2 Perhitungan Sudut 2 Titik.....	16
Program 2.2.3 Penggambaran Wall	17
Program 2.4.1 Penggambaran Asap	20
Program 2.4.2 Perubahan Keanggotaan Sumber Asap	21

DAFTAR PROGRAM

Program 2.1.1 Definisi Geometri Robot.....	9
Program 2.1.2 Pemodelan Chassis Robot.....	9
Program 2.1.3 Pemodelan Upper Robot	9
Program 2.1.4 Pemodelan Wheel Robot	10
Program 2.1.5 Pemodelan Help Wheel Robot.....	10
Program 2.1.6 Deklarasi Variabel Bagian Badan Robot.....	11
Program 2.1.7 Chassis - Upper	12
Program 2.1.8 Chassis - Wheel.....	12
Program 2.1.9 Chassis – Help Wheel.....	12
Program 2.1.10 Ruang Robot.....	13
Program 2.1.11 Penggambaran Robot (Robot Netral)	14
Program 2.1.12 Penggambaran Area Ketertarikan.....	15
Program 2.2.1 <i>Contractor</i> Kelas Wall	16
Program 2.2.2 Perhitungan Sudut 2 Titik.....	16
Program 2.2.3 Penggambaran Wall	17
Program 2.4.1 Penggambaran Asap	20
Program 2.4.2 Perubahan Keanggotaan Sumber Asap	21

BAB 1 PENDAHULUAN

Bab ini akan menjelaskan tentang latar belakang dilakukannya penelitian tugas akhir ini, rumusan masalah, tujuan yang ingin dicapai dengan pelaksanaan penelitian, ruang lingkup yang akan membatasi penelitian tugas akhir, tahapan dan metodologi yang penulis gunakan dalam melaksanakan penelitian, serta sistematika penulisan laporan.

1.1 Latar Belakang

Upaya pengembangan sistem deteksi kebocoran gas telah menjadi salah satu topik utama bagi banyak ahli di bidang komputasi, matematika, informatika, dan mekatronik atau robotik. Sistem deteksi kebocoran gas adalah kebutuhan yang krusial di era modern ini mengingat banyaknya ancaman yang mungkin muncul di dalam kehidupan masyarakat, seperti ancaman bom, kebocoran pipa gas, kebakaran hutan dan lain sebagainya. Sementara itu penggunaan anjing pelacak sebagai metode konvensional yang biasa digunakan untuk mendeteksi sumber gas memiliki resiko yang tinggi, terutama dalam hal keselamatan sang anjing, karena bisa saja gas yang bocor mengandung racun.

Seiring dengan perkembangan teknologi informasi dan mekatronik, berbagai upaya dilakukan untuk memecahkan permasalahan deteksi sumber gas. Pemanfaatan robot sebagai media atau agen pendeteksi sumber kebocoran gas menjadi salah satu solusi yang tepat. Penciptaan robot dengan perangkat lunak sehingga mampu mendeteksi keberadaan gas telah dikembangkan intensif satu dasawarsa terakhir. Para peneliti berupaya menciptakan sistem “indera penciuman” pada robot, seperti halnya pada makhluk hidup. Beberapa penelitian berhasil mengembangkan prototipe robot pendeteksi dengan menanamkan algoritma tertentu dan seprangkat perhitungan matematis di dalam sistem. Semua upaya tersebut bertujuan agar robot dapat mengenali keberadaan gas dari lingkungan sekitar kemudian robot mampu menelusuri arah datangnya konsentrasi gas dan menemukan keberadaan sumber gas tersebut.

Algoritma *Particle Swarm Optimization* (PSO) adalah salah satu metode yang telah diujicoba dan berhasil dengan baik untuk pencarian sumber gas. PSO diadaptasi dari kebiasaan hewan yang bergerak dalam koloninya untuk mencari makanan. Dalam pengembangannya, algoritma PSO telah dimodifikasi ke dalam sejumlah metode untuk menyelesaikan permasalahan deteksi sumber gas termasuk permasalahan dalam lingkungan dinamis. Upaya modifikasi tersebut telah dirintis melalui serangkaian penelitian berkelanjutan sejak beberapa tahun terakhir.

Wisnu [5] dalam penelitiannya, berhasil mengembangkan sistem deteksi satu sumber gas melalui modifikasi PSO dengan memanfaatkan mekanisme *Detect and Respons*, penggunaan *Charge Robot*, dan pemanfaatan prinsip *Wind Utilities*. Proses deteksi sumber yang telah dihasilkan hanya mampu diimplementasikan untuk proses pendeteksian keberadaan sumber gas tunggal. Padahal dalam lingkungan nyata jumlah sumber yang harus dideteksi sering kali lebih dari satu.

Pada penelitian selanjutnya, Aditya [1] berhasil mengembangkan sistem deteksi multi sumber gas pada lingkungan dinamis. Sistem yang dibangun berbentuk simulasi dengan memanfaatkan kumpulan-kumpulan robot yang akan bergerak secara paralel sehingga mampu mendeteksi sumber gas yang berjumlah lebih dari satu. Modifikasi algoritma PSO yang dikembangkan cukup berhasil memenuhi kebutuhan yang diharapkan, dimana robot mampu menemukan seluruh sumber gas yang disebar secara random. Pada penelitian juga dibuat simulator 2D seperti terlihat pada Gambar 1.1.1.

Penelitian ini merupakan penelitian lanjutan dari penelitian di atas. Fokus dari penelitian ini adalah untuk mengganti simulator 2D yang sudah ada dengan simulator 3D dan memodifikasi lagi algoritma PSO. Akan dilakukan uji coba dengan beberapa kombinasi percobaan terhadap sistem deteksi banyak sumber asap untuk melihat kelebihan algoritma baru yang diajukan. Sejauh mana kemampuan sistem menghasilkan kinerja optimum dalam ragam kombinasi percobaan, maka hal tersebut akan menjadi informasi berguna bagi penyempurnaan sistem di masa mendatang. Dari penelitian ini, juga tidak tertutup

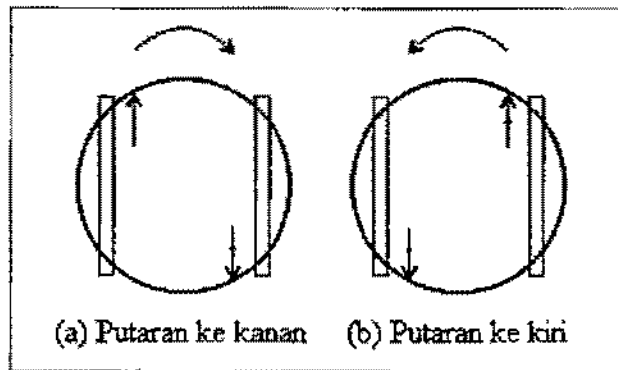
kemungkinan dikembangkannya modifikasi lanjutan atas sistem yang telah ada sehingga menghasilkan kinerja pencarian yang lebih optimal.



Gambar 1.1.1 Simulator 2D Niche-PSO

Dalam pemodelan 2D seperti di atas, robot hanya dimodelkan dengan gambar lingkaran. Mungkin robot akan tetap bertabrakan apabila bersinggungan dengan robot lain, tetapi proses pemantulan robot tidak mempedulikan mana bagian depan dan mana bagian belakang. Robot hanya akan berpantulan sebagaimana layaknya bola billiar.

Robot dalam dunia nyata perlu roda atau kaki untuk berpindah dari satu tempat ke tempat yang lain. Robot beroda apabila ingin mengubah haluan, maka robot tersebut harus memutar roda sebelah kiri dan kanan dengan arah yang berlawanan seperti ditunjukkan oleh Gambar 1.1.2.



Gambar 1.1.2 Perputaran robot

Dengan ini, kita memerlukan pemodelan 3D untuk mengurangi gap antara simulator dengan dunia nyata.

Algoritma PSO pada penelitian sebelumnya tidak bisa menjamin *niche* (kelompok robot) pada pencarian banyak sumber asap bergerak mencari sumber yang berbeda. Hal ini sangat tidak efisien karena ada kemungkinan lebih dari satu *niche* yang mencari sumber asap yang sama. Maka dari itu diperlukan algoritma baru yang bisa memberikan sifat divergen lebih terhadap *niche*, yang bisa menjamin *niche* bergerak mencari sumber yang berbeda sehingga pencarian akan menjadi lebih efisien karena tidak ada persaingan antar *niche*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, penulis merumuskan beberapa masalah yang akan dibahas pada penelitian tugas akhir ini adalah sebagai berikut.

1. Bagaimana cara membuat simulator versi 3D dari simulator versi 2D yang sudah ada?
2. Dimana kesalahan dan bagaimana cara memperbaiki algoritma yang sudah ada agar dapat mewujudkan *niche*-PSO yang lebih divergen?

1.3 Tujuan Penelitian

Adapun tujuan dilakukannya penelitian tugas akhir ini adalah sebagai berikut.

1. Meningkatkan level simulator dari 2D menjadi 3D untuk mengurangi gap antara perangkat keras dan dunia nyata.
2. Meningkatkan sifat divergen PSO agar didapat hasil pencarian yang lebih efisien.

1.4 Ruang Lingkup Penelitian

Berdasarkan tujuan yang ingin dicapai pada tugas akhir ini, maka penelitian akan dibatasi pada ruang lingkup sebagai berikut.

1. Ruang berbentuk bujur sangkar dengan panjang sisi yang bisa diubah-ubah.
2. Tidak ada halangan atau tanjakan.
3. Partikel-partikel asap dimodelkan dengan bentuk bola. Asap di dunia riil adalah kumpulan dari debu-debu yang jumlahnya sangat banyak. Model ini tidak diwujudkan berkenaan dengan memori komputer.
4. Evaluasi kinerja algoritma yang telah dikembangkan untuk melakukan penilaian secara objektif terhadap kualitas algoritma tersebut.

1.5 Tahapan Penelitian

Penelitian tugas akhir ini dilaksanakan dengan mengikuti tahapan-tahapan sebagai berikut.

1. Mengumpulkan bahan studi pustaka dan literatur yang terkait dengan permasalahan yang akan diteliti, kemudian mempelajarinya.
2. Mempelajari penelitian terdahulu yang terkait dengan topik yang akan diteliti pada penelitian tugas akhir ini. Salah satu penelitian yang menjadi

rujukan pada penelitian ini berjudul "*Cooperative Object Tracking with Mobile Robotic Sensor Network*" yang merupakan penelitian yang dilakukan oleh Aditya Nugraha dari Fakultas Ilmu Komputer, Universitas Indonesia, Indonesia.

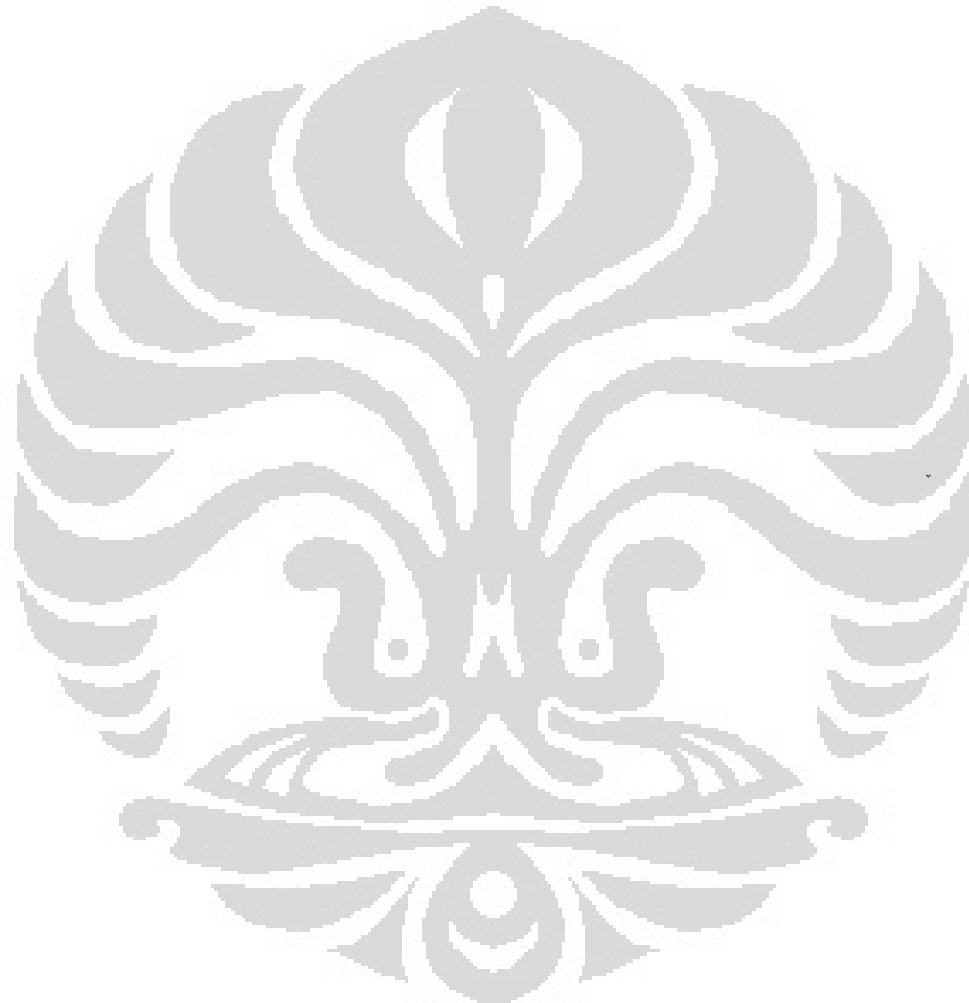
3. Melakukan perancangan simulator 3D dari simulator 2D yang sudah ada.
4. Melakukan perancangan metode penyelesaian baru yang dapat memberikan solusi yang lebih baik. Perancangan yang dilakukan disini bukan sekedar melakukan perubahan terhadap algoritma yang sudah diimplementasikan sebelumnya, tetapi juga merancang algoritma baru yang dapat menyelesaikan permasalahan yang sama dengan lebih baik.
5. Melakukan implementasi algoritma supaya penulis dapat melakukan pengujian secara objektif.

1.6 Sistematika Penulisan Laporan

Laporan kegiatan penelitian tugas akhir ini akan disusun berdasarkan sistematika penulisan sebagai berikut.

1. BAB 1 PENDAHULUAN, menjelaskan latar belakang dilakukannya penelitian tugas akhir ini, rumusan masalah, tujuan yang ingin dicapai dengan pelaksanaan penelitian, ruang lingkup yang akan membatasi penelitian tugas akhir, tahapan dan metodologi yang penulis gunakan dalam melaksanakan penelitian, serta sistematika penulisan laporan.
2. BAB 2 PEMODELAN ODE, menjelaskan tentang pembuatan model 3D. Dimulai dari model ruang, robot, dan asap.
3. BAB 3 PETUNJUK PEMAKAIAN SIMULATOR, adalah tutorial tentang cara menggunakan GUI simulator.
4. BAB 4 DYNAMIC NICHE-PSO, menjelaskan tentang algoritma baru yang diajukan pada penelitian kali ini. Menjelaskan tentang fungsi robot utama, dan hasil perbandingan dengan algoritma yang sebelumnya.

5. BAB 5 PENUTUP, menjelaskan tentang kesimpulan dan saran yang diperoleh melalui kegiatan penelitian tugas akhir ini. Pada bagian kesimpulan, penulis akan menyampaikan rangkuman hasil yang telah dicapai pada kegiatan penelitian yang dilakukan. Pada bagian saran, penulis mengemukakan usulan penelitian yang dapat dilaksanakan pada tahap pengembangan yang selanjutnya.

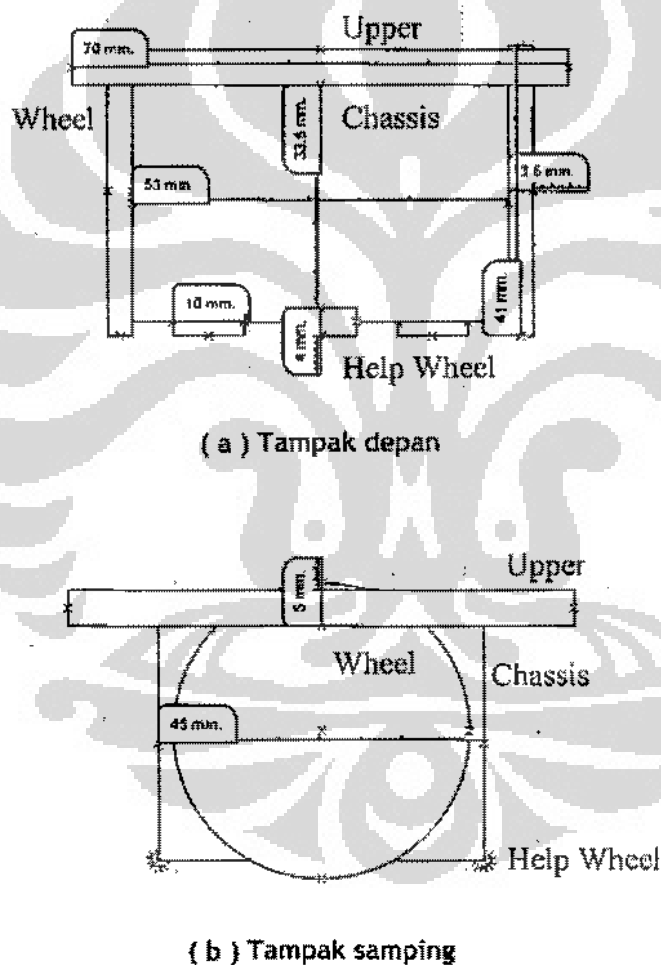


BAB 2 PEMODELAN ODE

Bab ini akan menjelaskan tentang pemodelan objek-objek seperti robot, medan, sumber asap, asap, dan posisi *Global Best*. Referensi pemrograman ODE dapat dilihat di manual yang bisa didapatkan dari <http://ode.org>.

2.1 Pemodelan Robot

Dimensi robot ditentukan seperti ditunjukkan oleh Gambar 2.1.1.



Gambar 2.1.1 Geometri Robot

Program untuk pemodelan pada Gambar 2.1.1 ditunjukkan oleh Program 2.1.1.

```

*****
Robot size
*****
/ Chassis
#define CHASSIS_LENGTH      0.045
#define CHASSIS_WIDTH      0.053
#define CHASSIS_HEIGHT     0.0335
// Upper body
#define UPPER_RADIUS 0.035
#define UPPER_HEIGHT 0.005
/ Wheel
#define WHEEL_RADIUS      0.0205
#define WHEEL_THICKNESS   0.0035
// Help wheel
#define HELPWHEEL_RADIUS  0.004
#define HELPWHEEL_THICKNESS 0.01

```

Program 2.1.1 Definisi Geometri Robot

Berikut Program 2.1.2 ~ Program 2.1.5 adalah program pemodelan badan-badan robot.

```

chassisBody = dBodyCreate( world ) ;
dBodySetPosition( chassisBody, xpos, ypos, STARTZ + 0.01375 ) ;
dMassSetBox( &m, 1, CHASSIS_LENGTH, CHASSIS_WIDTH, CHASSIS_HEIGHT ) ;
dMassAdjust( &m, CMASS ) ;
dBodySetMass( chassisBody, &m ) ;
chassisGeom = dCreateBox( 0, CHASSIS_LENGTH, CHASSIS_WIDTH, CHASSIS_HEIGHT ) ;
dGeomSetBody( chassisGeom, chassisBody ) ;

```

Program 2.1.2 Pemodelan Chassis Robot

```

upperBody = dBodyCreate( world ) ;
dBodySetPosition( upperBody, xpos, ypos, STARTZ + 0.033 ) ;
dMassSetCylinder( &m, 1, 3, UPPER_RADIUS, UPPER_HEIGHT ) ;
dMassAdjust( &m, UMASS ) ;
dBodySetMass( upperBody, &m ) ;
upperGeom = dCreateBox( 0, UPPER_RADIUS * 2, UPPER_RADIUS * 2,
UPPER_HEIGHT ) ;
dGeomSetBody( upperGeom, upperBody ) ;

```

Program 2.1.3 Pemodelan Upper Robot

```

for ( i = 0; i < 2; i++ ) {
    WheelBody[ i ] = dBodyCreate( world ) ;
    dQuaternion q ;
    dQFromAxisAndAngle( q, 1, 0, 0, M_PI * 0.5 ) ;
    dBodySetQuaternion( WheelBody[ i ], q ) ;
    dMassSetSphere( &m, 1, WHEEL_RADIUS ) ;
    dMassAdjust( &m, WMASS ) ;
    dBodySetMass( WheelBody[ i ], &m ) ;
    WheelGeom[ i ] = dCreateSphere( 0, WHEEL_RADIUS ) ;
    dGeomSetBody( WheelGeom[ i ], WheelBody[ i ] ) ;
}
dBodySetPosition( WheelBody[ 0 ], xpos,
    ypos + ( 0.5 * CHASSIS_WIDTH + WHEEL_THICKNESS ), STARTZ + 0.0155 ) ;
dBodySetPosition( WheelBody[ 1 ], xpos,
    ypos - ( 0.5 * CHASSIS_WIDTH + WHEEL_THICKNESS ), STARTZ +

```

Program 2.1.4 Pemodelan Wheel Robot

```

for ( i = 0; i < 3; i++ ) {
    helpWheelBody[ i ] = dBodyCreate( world ) ;
    dQuaternion q ;
    dQFromAxisAndAngle( q, 1, 0, 0, M_PI * 0.5 ) ;
    dBodySetQuaternion( helpWheelBody[ i ], q ) ;
    dMassSetSphere( &m, 1, HELPWHEEL_RADIUS ) ;
    dMassAdjust( &m, WMASS ) ;
    dBodySetMass( helpWheelBody[ i ], &m ) ;
    helpWheelGeom[ i ] = dCreateSphere( 0, HELPWHEEL_RADIUS ) ;
    dGeomSetBody( helpWheelGeom[ i ], helpWheelBody[ i ] ) ;
}
dBodySetPosition( helpWheelBody[ 0 ], xpos + ( 0.5 * CHASSIS_LENGTH ), ypos,
    STARTZ - 0.003 ) ;
dBodySetPosition( helpWheelBody[ 1 ], xpos - ( 0.5 * CHASSIS_LENGTH ),
    ypos + ( 0.25 * CHASSIS_WIDTH ), STARTZ - 0.003 ) ;
dBodySetPosition( helpWheelBody[ 2 ], xpos - ( 0.5 * CHASSIS_LENGTH ),
    ypos - ( 0.25 * CHASSIS_WIDTH ), STARTZ - 0.003 ) ;

```

Program 2.1.5 Pemodelan Help Wheel Robot

Robot dalam simulasi ini semua mempunyai bentuk yang sama, jadi program diatas semua diletakkan di dalam *constructor* kelas Robot. Parameter yang dioper adalah parameter posisi robot yaitu variabel *xpos* dan *ypos*, karena robot bisa ada lebih dari 1.

Variabel-variabel untuk badan robot pada program diatas dideklarasikan di dalam kelas Robot yang ditunjukkan oleh Program 2.1.6.

```
public:
    dBodyID upperBody ;
    dGeomID upperGeom ;
    dBodyID middleBody ;
    dGeomID middleGeom ;
    dBodyID chasisBody ;
    dGeomID chasisGeom ;
    dBodyID WheelBody[ 2 ] ;
    dGeomID WheelGeom[ 2 ] ;
    dBodyID helpWheelBody[ 3 ] ;
    dGeomID helpWheelGeom[ 3 ] ;
```

Program 2.1.6 Deklarasi Variabel Bagian Badan Robot

Setelah bagian badan robot dibuat, maka harus dipasangkan. Pemasangan bagian badan robot dilakukan dengan *joint*. Jenis *joint* yang digunakan ditunjukkan pada Tabel 2.1.1.

Tabel 2.1.1 Penetapan *Joint*

Bagian Robot		Joint
Chassis	Upper	Fixed
Chassis	Wheel	Hinge2
Chassis	Help Wheel	Hinge

Fixed joint dibuat untuk menggabungkan 2 objek. Tidak ada sumbu putar pada *fixed joint*. Perbedaan *hinge2 joint* dan *hinge joint* adalah pada *hinge2 joint* terdapat suspensi yang cocok diimplementasikan untuk roda robot beroda.

Program penggabungan bagian robot dengan *joint* ditunjukkan pada Program 2.1.7 ~ Program 2.1.9.

```

cha_up_joint = dJointCreateFixed( world, 0 ) ;
dJointAttach( cha_up_joint, chassisBody, upperBody ) ;
dJointSetFixed( cha_up_joint ) ;

```

Program 2.1.7 Chassis - Upper

dJointAttach hanya memberikan informasi bahwa bagian yang dihubungkan adalah bagian yang dioper di parameter pertama dan kedua, kedua bagian ini belum tersambung. Ketika fungsi dJointSetFixed dipanggil maka tersambunglah kedua bagian ini.

```

for ( i = 0; i < 2; i++ ) {
    joint[ i ] = dJointCreateHinge2( world, 0 ) ;
    dJointAttach( joint[ i ], chassisBody, WheelBody[ i ] ) ;
    const dReal *a = dBodyGetPosition( WheelBody [ i ] ) ;
    dJointSetHinge2Anchor( joint[ i ], a[ 0 ], a[ 1 ], a[ 2 ] ) ;
    dJointSetHinge2Axis1( joint[ i ], 0, 0, 1 ) ;
    dJointSetHinge2Axis2( joint[ i ], 0, 1, 0 ) ;
    dJointSetHinge2Param( joint[ i ], dParamLoStop, 0 ) ;
    dJointSetHinge2Param( joint[ i ], dParamHiStop, 0 ) ;
    dJointSetHinge2Param( joint[ i ], dParamSuspensionERP, 0.01 ) ;
    dJointSetHinge2Param( joint[ i ], dParamSuspensionCFM, 0.01 ) ;
}

```

Program 2.1.8 Chassis - Wheel

```

for ( i = 0; i < 3; i++ ) {
    help_joint[ i ] = dJointCreateHinge( world, 0 ) ;
    dJointAttach( help_joint[ i ], chassisBody, helpWheelBody[ i ] ) ;
    const dReal *a = dBodyGetPosition( helpWheelBody [ i ] ) ;
    dJointSetHingeAnchor( help_joint[ i ], a[ 0 ], a[ 1 ], a[ 2 ] ) ;
    dJointSetHingeAxis( help_joint[ i ], 0, 1, 0 ) ;
    dJointSetHingeParam( help_joint[ i ], dParamLoStop, 0 ) ;
    dJointSetHingeParam( help_joint[ i ], dParamHiStop, 0 ) ;
}

```

Program 2.1.9 Chassis - Help Wheel

dParamLoStop dan dParamHiStop adalah batas perputaran ke arah kiri dan ke arah kanan. Apabila nilai kedua parameter ini sama, maka artinya tidak ada batas perputaran baik yang ke arah kiri maupun yang ke arah kanan. Arah perputaran

dilihat dari badan pertama yang menjadi pivot. Badan ini adalah badan yang dioper di parameter pertama pada fungsi `dJointAttach`.

Dalam ODE, semua objek yang menempel akan dihitung tumbukannya. Kemana objek akan dipentalkan dan berapa kecepatannya. Tetapi sebelum melakukan perhitungan tumbukan ini, ODE melihat apakah objek tersebut mempunyai *joint* dengan objek yang menempel dengannya. Jika ada *joint*, maka tidak akan dilakukan proses perhitungan tumbukan. Tetapi ada cara lain untuk menghemat proses ini yaitu dengan memasukkan semua bagian-bagian robot ke dalam ruang robot. Dengan begini, ODE tidak akan bahkan melihat apakah ada *joint* atau tidak, karena bagian-bagian tersebut sudah menjadi suatu kesatuan. Program ini ditunjukkan oleh Program 2.1.10.

```
robotSpace = dlashSpaceCreate (space);
dSpaceSetCleanup (robotSpace, 0);

dSpaceAdd( robotSpace, chasisGeom ) ;
dSpaceAdd( robotSpace, upperGeom ) ;
dSpaceAdd( robotSpace, wheelGeom[ 0 ] ) ;
dSpaceAdd( robotSpace, wheelGeom[ 1 ] ) ;
dSpaceAdd( robotSpace, helpWheelGeom[ 0 ] ) ;
dSpaceAdd( robotSpace, helpWheelGeom[ 1 ] ) ;
dSpaceAdd( robotSpace, helpWheelGeom[ 2 ] ) ;
```

Program 2.1.10 Ruang Robot

Dalam ODE, objek yang sudah dibuat badannya tidak akan terlihat jikalau tidak digambarkan. Dibawah ini ditunjukkan program penggambaran robot. Dalam penelitian ini terdapat 3 jenis robot yaitu robot netral, robot bermuatan, dan robot utama. Untuk membedakan mereka, maka warna bagian Upper dibedakan yang ditunjukkan oleh Tabel 2.1.2.

Tabel 2.1.2 Penetapan Warna Robot

Jenis Robot	Warna	(R, G, B)
Netral	Merah	(1.0, 0.0, 0.0)
Bermuatan	Kuning	(1.0, 1.0, 0.0)
Utama	Ungu	(0.8, 0.2, 1.0)

```

void RobotPS0::draw() {
    int i;

    dsSetColor( 1.0, 0.0, 0.0 ); ①
    dsDrawCylinder( dBodyGetPosition( upperBody ),
                   dBodyGetRotation( upperBody ),
                   UPPER_HEIGHT, UPPER_RADIUS );

    dsSetColor( 0.7, 0.7, 0.7 );
    const dReal sides[ 3 ] = { CHASSIS_LENGTH, CHASSIS_WIDTH,
                               CHASSIS_HEIGHT };

    dsDrawBox( dBodyGetPosition( chasisBody ),
              dBodyGetRotation( chasisBody ),
              sides );

    dsSetColor( 1.3, 1.3, 1.3 );
    for (i=0; i<2; i++) {
        dsDrawCylinder( dBodyGetPosition( WheelBody[ i ] ),
                       dBodyGetRotation( WheelBody[ i ] ),
                       WHEEL_THICKNESS, WHEEL_RADIUS );
    }

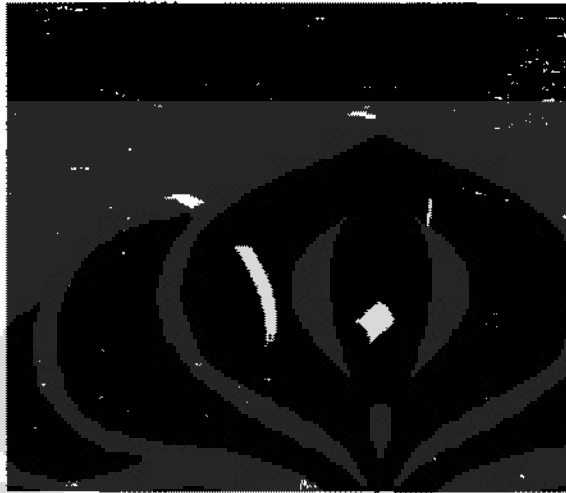
    for ( i = 0; i < 3; i++ ) {
        const dReal *temp = dBodyGetPosition( helpWheelBody[ i ] );
        dsDrawCylinder( dBodyGetPosition( helpWheelBody[ i ] ),
                       dBodyGetRotation( helpWheelBody[ i ] ),
                       HELPWHEEL_THICKNESS, HELPWHEEL_RADIUS );
    }
}

```

Program 2.1.11 Penggambaran Robot (Robot Netral)

Untuk robot bermuatan dan robot utama, tinggal mengganti nilai RGB yang ditunjukkan oleh ① pada Program 2.1.11 diatas.

Gambar salah satu robot yaitu robot bermuatan ditunjukkan oleh di bawah.



Gambar 2.1.2 Robot Bermuatan

Robot utama mempunyai area ketertarikan. Area ketertarikan ini ditandai oleh lingkaran berwarna biru. Radius lingkaran ini menunjukkan radius ketertarikan robot utama. Maka dari itu titik tengah dari koordinat x dan y lingkaran sama dengan robot.

Dibawah adalah bagian dari penggambaran robot utama, yaitu bagian penggambaran area ketertarikan. Bagian lain sama dengan Program 2.1.11.

```
dsSetColorAlpha( 0.0, 0.0, 1.0, 0.3 ) ;
dsDrawCylinder( dBodyGetPosition( upperBody ),
                dBodyGetRotation( upperBody ),
                0.001, MRobotPSO::attractRadius ) ;
```

Program 2.1.12 Penggambaran Area Ketertarikan

Lingkaran area ketertarikan tetap digambarkan dengan silinder tetapi dengan ketebalan yang sangat kecil, yaitu 0.001 m.

2.2 Pemodelan Medan

Medan dalam penelitian ini berupa bujur sangkar. Karena bujur sangkar mempunyai panjang sisi yang sama, maka objek medan disederhanakan menjadi 1 objek dinding.

Di program dinding disebut sebagai Wall. *Constructor* kelas Wall ditunjukkan oleh Program 2.2.1.

```

Wall::Wall( dWorldID world, dSpaceID space,
           dReal x1, dReal y1, dReal x2, dReal y2)
{
    dReal range = sqrt( pow( x1 - x2, 2 ) + pow( y1 - y2, 2 ) ) ; ①
    wallGeom = dCreateBox ( space, range, 0.1, 0.2 ) ;
    dMatrix3 R ;
    double angle = getAngleRad( x1, y1, x2, y2 ) ;
    dRFromAxisAndAngle( R, 0, 0, 1, angle ) ;
    dGeomSetPosition( wallGeom, x1 - ( ( x1 - x2 ) / 2 ),
                    y1 - ( ( y1 - y2 ) / 2 ), 0.1 ) ; ②
    dGeomSetRotation( wallGeom, R ) ;
}

```

Program 2.2.1 *Constructor* Kelas Wall

Parameter yang dioper adalah koordinat ujung Wall. Posisi dalam ODE adalah titik tengah objek. Koordinat titik tengah objek ditentukan dengan menghitung titik tengah antara 2 koordinat ujung Wall sebagai elemen x atau y yang pada Program 2.2.1 ditunjuk oleh ①, dan setengah dari tinggi Wall sebagai elemen z. Karena tinggi dinding adalah 0.2 m, maka elemen z koordinat titik tengah objek Wall menjadi 0.1 seperti ditunjukkan oleh ② pada Program 2.2.1.

Fungsi `getAngleRad` pada Program 2.2.1 ditunjukkan oleh Program 2.2.2 dibawah.

```

double Wall::getAngleRad( double x1, double y1, double x2, double y2)
{
    double angle = ( y1 - y2 ) / ( x1 - x2 ) ;
    angle = atan( angle ) ;
    return angle ;
}

```

Program 2.2.2 Perhitungan Sudut 2 Titik

Fungsi ini dilanjutkan oleh fungsi ODE `dRFromAxisAndAngle` untuk menghitung objek berapa derajat dirotasikan dari posisi inisial objek dibuat dalam ODE yang lurus dengan garis $y = 1$.

Objek Wall tidak perlu untuk dibuatkan *body*, karena antara 1 dinding dengan yang lain tidak disambung dengan *joint*. Dinding hanya ditaruh sedemikian sehingga membentuk bujur sangkar. Objek Wall hanya memerlukan *geometry* agar diketahui apakah robot berkolisi dengannya atau tidak. Karena dalam ODE *geometry* digunakan dalam perhitungan pendeteksian kolisi.

Program penggambaran objek ditunjukkan oleh Program 2.2.3.

```
void Wall::draw(void)
{
    dVector3 ss ;

    dsSetColorAlpha( 0.8, 0.8, 0.8, 0.7 ) ;
    dGeomBoxGetLengths( wallGeom, ss ) ;
    dsDrawBoxD( dGeomGetPosition( wallGeom ),
                dGeomGetRotation( wallGeom ), ss ) ;
}
```

Program 2.2.3 Penggambaran Wall

Wall digambarkan dengan transparansi 70%. 70% bisa dilihat dari parameter transparansi pada fungsi `dsSetColorAlpha` yang bernilai 0.7. Ini bertujuan agar apabila kamera diletakkan pada ketinggian yang tidak melebihi tinggi wall, robot, asap, sumber asap, posisi *Global Best* masih tetap terlihat.

Gambar di bawah adalah hasil penggambaran dinding. Hal yang tidak realistis disini adalah asap yang menembus dinding. Di dalam dunia nyata, asap akan berkumpul di dinding. Di bagian dinding konsentrasi asap akan lebih tinggi. Di sini dinding hanya berfungsi sebagai pembatas ruang pencarian. Maka dari itu hal realistis semacam ini diabaikan.



Gambar 2.2.1 Dinding

2.3 Pemodelan Asap

Program penggambaran asap ditunjukkan oleh

```

if ( draw_plumes ) {
  for ( int ii = 0 ; ii < N_PUFF ; ii++ ) {
    // if puff is in its source,
    // it shouldn't be draw to make it efficient
    bool check = true ;
    for ( int j = 0 ; j < x_s.size() ; j++ ) {
      if ( puff[ii].xx == x_s[ j ] &&
          puff[ii].yy == y_s[ j ] ) {
        check = false ;
        break ;
      }
    }
    if ( check ) {
      double transparent = ( 1 / ( puff[ ii ].R3 * 200 ) ) - 0.01 ;
      dsSetColorAlpha( 0.8f, 0.0f, 1.0f, transparent ) ;
      double range = arena_width_meter / 2 ;
      const dReal pos[ 3 ] = { puff[ ii ].xx + range,
                             puff[ ii ].yy + range,
                             puff[ ii ].zz } ;

      dMatrix3 R ;
      dRFromAxisAndAngle( R, 0, 0, 0 ) ;
      dsDrawSphereD( pos, R, puff[ ii ].R3 * 5 ) ;
    }
  }
}

```

Asap dalam penelitian ini mempunyai sifat sebagai berikut :

1. Transparan.
2. Tidak berkolisi dengan objek manapun.
3. Digambarkan dengan bentuk bola.

Transparansi asap dimulai dari nilai 0% atau tidak transparan sama sekali ketika koordinatnya sama dengan koordinat sumber asap, dimana artinya asap baru saja muncul. Tetapi untuk mengurangi beban prosesor, proses penggambaran asap pada saat asap pertama kali muncul ini ditiadakan. Karena walaupun asap digambarkan juga tidak akan terlihat. Seiring dengan semakin waktu berjalan asap akan semakin transparan, sampai pada akhirnya tidak kelihatan atau nilai transparansi 100%.

Asap pada penelitian ini tidak mempunyai kolisi dengan objek manapun. Dalam bahasa ODE, asap hanya memiliki *body* tapi tidak memiliki *geometry*.

Penggambaran asap adalah dengan sebuah bola yang semakin membesar seiring dengan waktu yang berjalan semenjak asap muncul dari sumber asap. Asap dalam dunia nyata adalah kumpulan partikel debu, dimana jumlah partikel debu ini sangat banyak sekali. Karena keterbatasan komputer, asap tidak mungkin untuk digambarkan sesuai dengan asap di dunia nyata.



Gambar 2.3.1 Asap

2.4 Pemodelan Sumber Asap

Sumber asap dimodelkan sebagai setengah bola. Bentuk sebenarnya adalah bola penuh, tetapi hanya bagian atas yang terlihat. Dibawah adalah program penggambaran asap.

```

// draw source
For (int ii = 0; ii < ( signed )x_s.size(); ii++) {
    dsSetColorAlpha(0.8f, 0.0f, 1.0f, 0.9) ;
    double range = arena_width_meter / 2 ;
    const dReal pos[ 3 ] = { x_s[ ii ] + range, y_s[ ii ] + range, 0 } ;
    dMatrix3 R ;
    dRFromAxisAndAngle( R, 0, 0, 0, 0 ) ;
    dsDrawSphereD( pos, R, 0.03 ) ;
}

// draw found source
For (int ii = 0; ii < ( signed )x_found.size(); ii++) {
    dsSetColorAlpha( 1.0f, 1.0f, 1.0f, 0.9 ) ;
    double range = arena_width_meter / 2 ;
    const dReal pos[ 3 ] = { x_found[ ii ] + range, y_found[ ii ] + range, 0 } ;
    dMatrix3 R ;
    dRFromAxisAndAngle( R, 0, 0, 0, 0 ) ;
    dsDrawSphereD( pos, R, 0.03 ) ;
}

```

Program 2.4.1 Penggambaran Asap

Pada Program 2.4.1 terlihat bahwa sumber asap ditunjukkan dengan koordinat, yaitu x_s , y_s , x_{found} , dan y_{found} . Dimana x_s dan y_s adalah sumber asap sebelum ditemukan, dan x_{found} dan y_{found} adalah sumber asap setelah ditemukan yang berupa vektor. Dibawah adalah program pemindahan sumber asap dari vektor sumber asap yang belum ditemukan ke vektor sumber asap yang telah ditemukan.

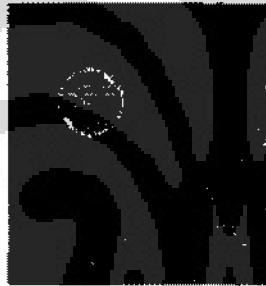
```

source which is found is saved in another vector
void Wind::setFoundSource(int index)
{
    x_found.push_back(x_s[index]);
    y_found.push_back(y_s[index]);
    x_s.erase(x_s.begin() + index);
    y_s.erase(y_s.begin() + index);
}

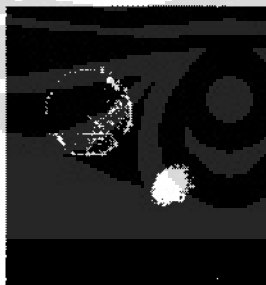
```

Program 2.4.2 Perubahan Keanggotaan Sumber Asap

Gambar 2.4.1 dan Gambar 2.4.2 di bawah adalah gambar sumber asap. Gambar 2.4.1 adalah gambar sumber asap sebelum ditemukan dan Gambar 2.4.2 adalah gambar sumber asap setelah ditemukan. Robot dalam penelitian ini dalam menutup sumber asap hanya digambarkan dengan merubah warna dari ungu ke abu-abu.



Gambar 2.4.1 Sumber Asap (Sebelum ditemukan)

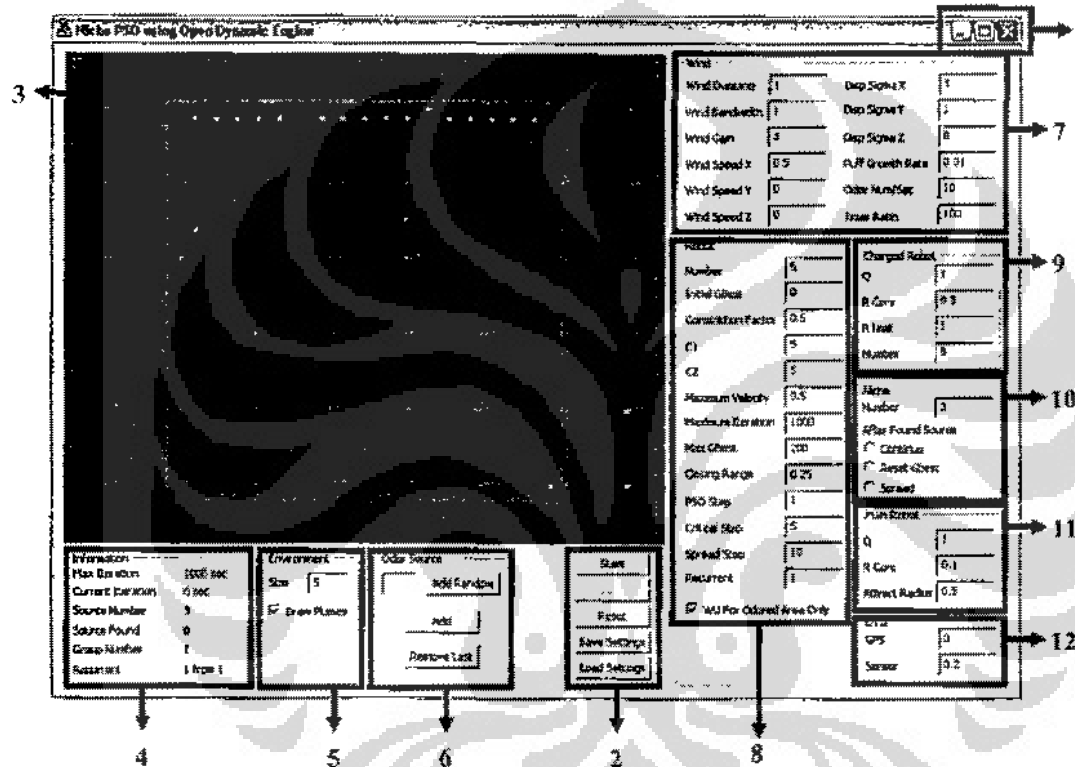


Gambar 2.4.2 Sumber Asap (Setelah ditemukan)

BAB 3 PETUNJUK PEMAKAIAN SIMULATOR

Bab ini akan menjelaskan tentang petunjuk pemakaian simulator. Yaitu tentang bagaimana pengaturan robot, medan, jumlah asap, atau iterasi.

3.1 Antar Muka



Gambar 3.1.1 GUI Simulator

Gambar 3.1.1 adalah layout Simulator Pencarian Sumber Asap, berikut adalah keterangan untuk nomor-nomor pada gambar:

1. Fungsi yang telah disediakan oleh Windows™ untuk mengecilkan dan memaksimalkan jendela Simulator. Untuk menutup Simulator dapat menggunakan tombol 'close', yaitu tombol yang paling kanan pada bagian ini.
2. Menu Simulator

Dijelaskan lebih lengkap pada subbab 3.3.

3. Tampilan animasi simulasi

Bagian yang menggambarkan animasi pencarian sumber asap. Penjelasan lebih lengkap dapat dibaca pada subbab 3.4.

4. Tampilan informasi Simulator

Memberikan informasi mengenai pengaturan simulasi dan hasil yang telah dicapai dalam simulasi sampai kurun waktu tertentu. Penjelasan lebih lengkap dapat dibaca pada subbab 3.5.

5. Environment Menu

Digunakan untuk mengatur kondisi lingkungan simulasi. Penjelasan lebih lengkap dapat dibaca pada subbab 3.7.

6. Odor Source Menu

Digunakan untuk mengatur jumlah dan letak sumber asap. Penjelasan lebih lengkap dapat dibaca pada subbab 3.8.

7. Wind Menu

Digunakan untuk mengatur perilaku angin. Penjelasan lebih lengkap dapat dibaca pada subbab 3.9.

8. Robot Menu

Digunakan untuk mengatur perilaku robot dan jumlah robot netral. Penjelasan lebih lengkap dapat dibaca pada subbab 3.10.

9. Charge Robot Menu

Digunakan untuk mengatur besar muatan dan jumlah robot bermuatan. Penjelasan lebih lengkap dapat dibaca pada subbab 3.11.

10. Niche Menu

Digunakan untuk mengatur jumlah dan perilaku Niche atau koloni robot. Penjelasan lebih lengkap dapat dibaca pada subbab 3.12.

11. Main Robot Menu

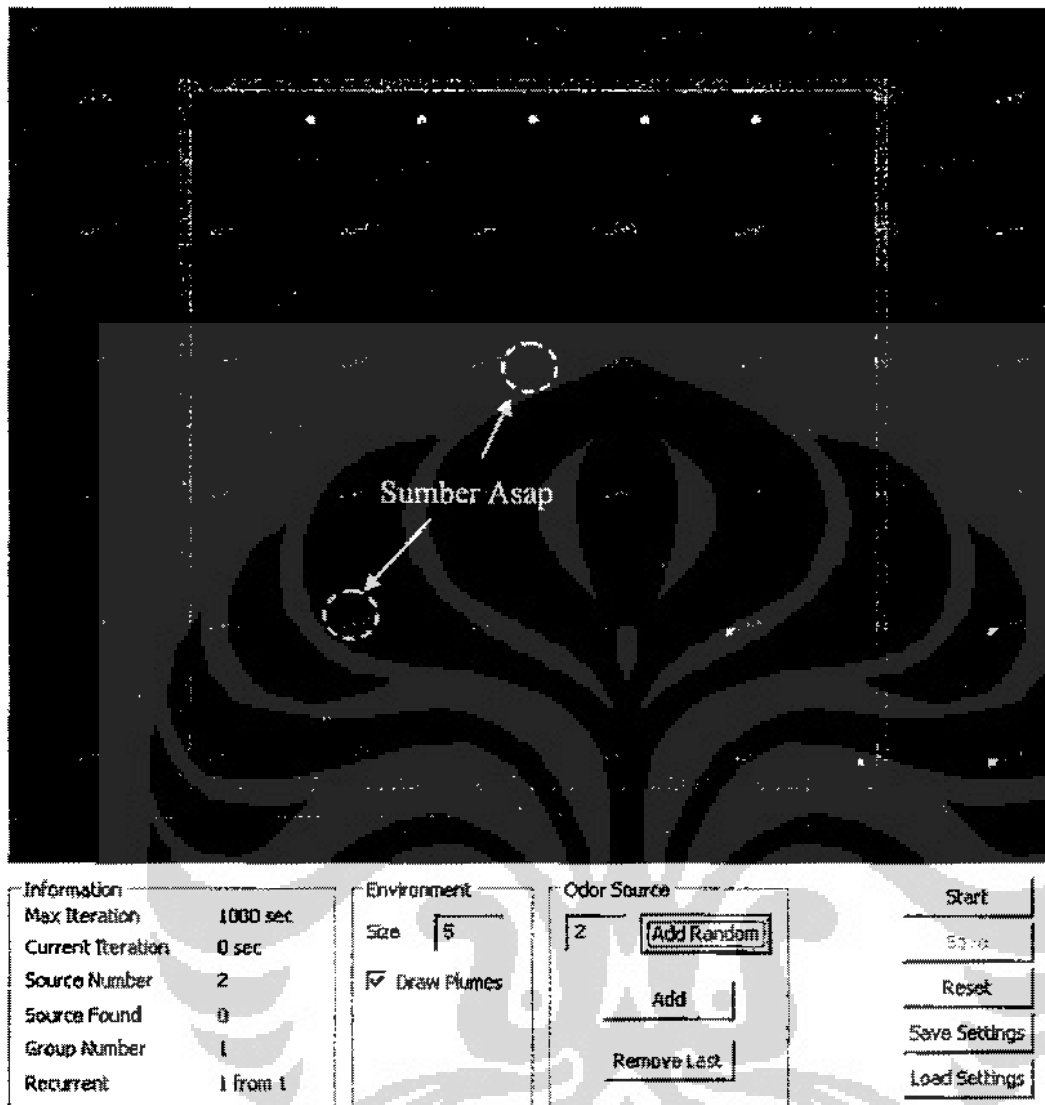
Mengatur atribut robot utama, yaitu muatan robot utama, radius penolakan robot utama, dan radius area ketertarikan robot utama. Penjelasan lebih lengkap dapat dibaca pada subbab 4.3.1

12. Error Menu

Mengatur faktor kesalahan dalam pembacaan sensor. Penjelasan lebih lengkap dapat dibaca pada subbab 3.13.

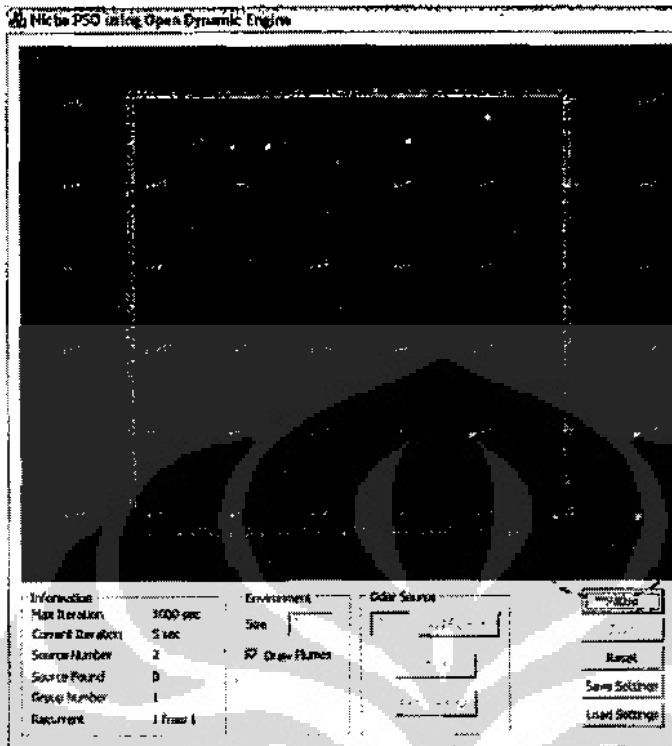
3.2 Menjalankan Simulator

Untuk menjalankan Simulator, anda terlebih dahulu harus menentukan jumlah dan letak sumber kebocoran. Hal ini dapat dilakukan pada bagian yang ditandai dengan angka 6 pada Gambar 3.1.1. Sebagai contoh, pada Gambar 3.2.1 telah ditambahkan 1 buah sumber asap yang secara acak. Untuk melakukan hal ini, isi jumlah sumber asap pada bagian yang ditandai dengan lingkaran merah pada Gambar 3.2.1, yaitu 2. Jika sudah tekan tombol "Add random" yang ada disebelahnya. Simulator akan menentukan secara acak posisi sumber asap dan akan ditampilkan pada bagian animasi. Pada Gambar 3.2.1, posisi sumber asap terletak dalam lingkaran yang berwarna biru muda. Sumber asap digambarkan dengan sebuah titik berwarna ungu.

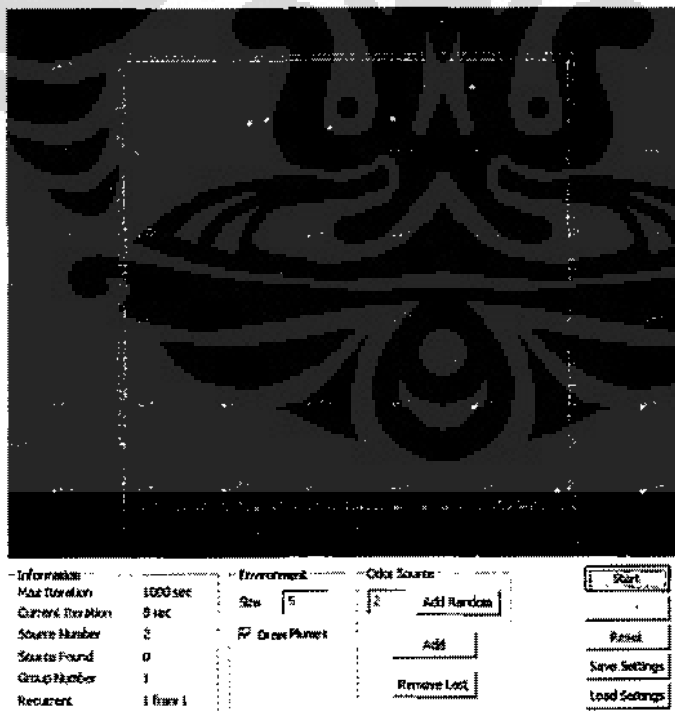


Gambar 3.2.1 Penambahan Sumber Asap

Setelah sumber asap ditentukan, Simulator dapat dijalankan dengan menekan tombol 'Start'. Gambar 3.2.2 menunjukkan tampilan Simulator pada saat dijalankan. Dapat dilihat bahwa pada saat Simulator dijalankan Anda tidak diperbolehkan melakukan pengubahan pada elemen-elemen simulasi. Selain itu, tombol 'Start' berubah menjadi tombol 'Pause', perhatikan tombol yang dilingkari dengan warna merah pada Gambar 3.2.2. Tombol 'Pause' dapat digunakan untuk menghentikan simulasi.



Gambar 3.2.2 Tombol "Pause"

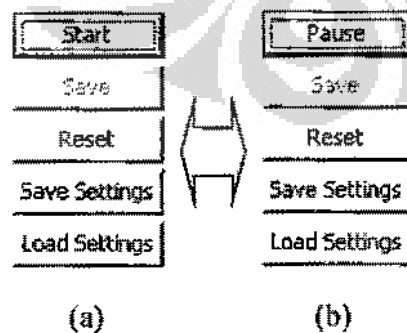


Gambar 3.2.3 Simulasi

Jika Anda menekan tombol 'Pause', maka Simulator akan berada dalam kondisi *pause*. Gambar 3.2.3 memperlihatkan Simulator pada kondisi *pause*. Pada kondisi ini Anda dapat melakukan pengaturan ulang terhadap elemen-elemen simulasi. Jika Anda menekan tombol 'Start', maka simulasi akan dilanjutkan. Jika Anda melakukan perubahan, data-data tersebut baru akan dibaca oleh Simulator setelah tombol 'Save' ditekan. Menekan tombol 'Start' setelah melakukan perubahan tanpa terlebih dahulu menekan tombol 'Save' akan membuat Simulator melanjutkan simulasi dengan parameter sebelum kondisi *pause*. Perlu diingat bahwa setiap perubahan terhadap elemen-elemen yang ada, kecuali untuk pilihan 'Draw Plums', akan membuat Simulator melakukan simulasi dari awal.

3.3 Menu Simulator

Menu Simulator ditunjukkan pada Gambar 3.3.1. Tombol-tombol pada bagian ini digunakan untuk mengatur perilaku Simulator. Gambar 3.3.1 (a) menunjukkan tombol-tombol pada saat Simulator tidak dalam keadaan menjalankan simulasi. Gambar 3.3.1 (b) adalah tampilan pada saat Simulator dalam kondisi menjalankan simulasi. Perbedaan antara (a) dan (b) adalah tombol 'Start' dan 'Pause'. Kedua tombol ini akan muncul bergantian, tergantung pada kondisi apakah simulasi sedang dijalankan atau tidak. Jika sedang dijalankan maka tombol 'Pause' yang akan muncul. Tombol 'Start' akan muncul jika simulasi tidak sedang dijalankan.



Gambar 3.3.1 Menu Simulator

Berikut penjelasan untuk masing-masing tombol:

- **Start**

Digunakan untuk menjalankan simulasi sesuai dengan parameter yang diberikan. Tombol ini baru dapat berfungsi jika sudah ada sumber asap yang akan dicari oleh robot. Jika sebelumnya Simulator dalam keadaan *pause* dan tidak ada elemen yang diubah, maka Simulator akan melanjutkan perhitungan simulasi sesuai kondisi sebelum keadaan *pause*.

- **Pause**

Hanya akan berlaku pada kondisi Simulator menjalankan simulasi. Menekan tombol ini akan membuat simulasi dihentikan sementara dan Simulator masuk dalam kondisi *pause*. Karena simulasi dihentikan maka animasi juga tidak akan berjalan.

- **Save**

Tombol ini akan membuat Simulator mengubah nilai tiap elemennya sesuai dengan nilai yang diberikan. Perubahan pada elemen-elemen simulasi tidak akan berlaku dalam simulasi sebelum tombol ini ditekan. Menekan tombol ini pada saat Simulator dalam kondisi *pause* selain membuat Simulator membaca nilai tiap elemen juga akan membuat Simulator melakukan simulasi dari awal.

- **Reset**

Mengembalikan nilai pada setiap elemen sesuai dengan nilai terakhir yang disimpan oleh Simulator. Hal ini berarti jika telah Anda melakukan perubahan dan juga sudah menekan tombol 'Save'. Anda melakukan perubahan lagi tanpa menekan tombol 'Save' dan menekan tombol 'Reset', maka Simulator akan mengembalikan nilai setiap elemen sesuai dengan perubahan yang terakhir Anda 'Save'. Hal ini akan terus berlaku sampai Anda menekan tombol 'Save' untuk kedua kalinya, sekarang tombol 'Reset' akan mengembalikan nilai setiap elemen sesuai perubahan terakhir yang Simulator simpan. Atau dengan kata lain kembali sesuai perubahan kedua yang Anda lakukan.

- Save Setting dan Load Setting

'Save Setting' digunakan untuk menyimpan nilai setiap elemen pada sebuah berkas. Tujuannya agar nilai-nilai tersebut dapat dibaca kembali dikemudian hari. Sangat berguna jika Anda sedang melakukan pengambilan data percobaan dan terpaksa berhenti ditengah-tengah. Dengan menyimpan nilai-nilai tersebut, Anda dapat melanjutkan percobaan tanpa perlu direpotkan dengan pengaturan elemen-elemen simulasi.

'Load Setting' digunakan untuk membuat Simulator mengubah nilai tiap elemen sesuai dengan data yang disimpan. Berkas untuk menyimpan semua informasi ini berada dalam folder hasil instalasi program Simulator. Berkas tersebut diberi nama Data.txt. Sebagian isi berkas tersebut dapat dilihat pada Gambar 3.3.2.

```

wind information
=====
100.00      Wind::wind_damping
100.00      Wind::wind_bandwidth
100.00      Wind::wind_gain
100.00      Wind::speedx
100.00      Wind::speedy
100.00      Wind::speedz
100.00      Wind::sigmax
100.00      Wind::sigmay
100.00      Wind::sigmaz
100.00      Wind::puffs_growth_rate
100         Wind::Odor_Num_Per_Second
0.00       Wind::timer

robot information
=====
30         RobotPSO::n_robot
100.00     RobotPSO::gbest_init
100.00     RobotPSO::cons_factor
100.00     RobotPSO::c1
100.00     RobotPSO::c2
10.00      RobotPSO::max_velocity

searching information
=====
100        RobotPSO::max_iteration

```

Gambar 3.3.2 Bagian dari Data.txt

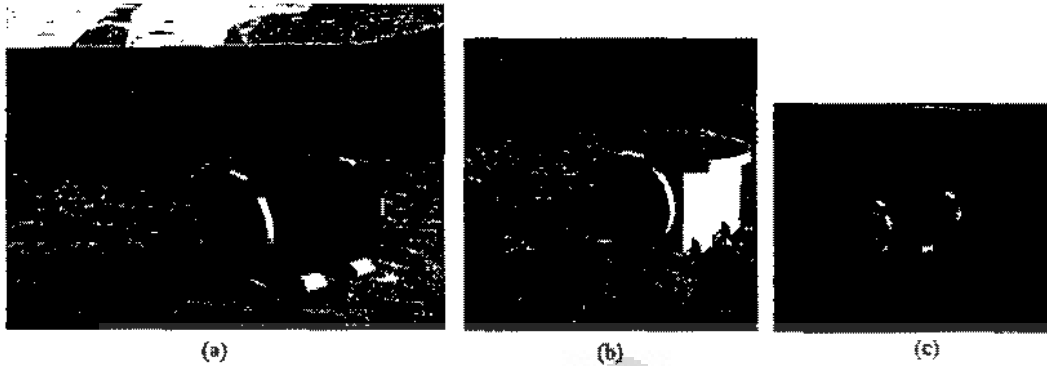
3.4 Tampilan Animasi Simulasi

Pada simulator, bagian yang terlihat seperti Gambar 3.4.1 adalah tempat dimana animasi pencarian sumber asap digambarkan. Pada Gambar 3.4.1, bagian yang dikelilingi oleh dinding dengan banyak titik didalamnya disebut ruang pencarian. Dinding membatasi pergerakan robot. Robot tidak dapat berjalan menembus dinding atau menghancurkan dinding. Jika ada sumber asap diluar dinding, maka robot-robot tidak akan pernah bisa menemukannya.

Berbeda dengan robot, asap dapat menembus dinding. Jika terdapat sumber asap diluar dinding dan asapnya masuk dalam ruang pencarian, maka kemampuan robot dalam mencari sumber asap dalam ruang pencarian akan terganggu. Sumber asap digambarkan dengan sebuah titik berwarna ungu. Asap-asap yang keluar dari sumber asap juga digambarkan dengan warna ungu. Bola yang diberikan tanda Global Best adalah posisi pergerakan terbaik hasil perhitungan dari seluruh robot yang diturunkan dalam pencarian.



Gambar 3.4.1 Ruang Pencarian



Gambar 3.4.2 Robot Netral dan Robot Bermuatan

Robot-robot dalam simulasi terbagi dalam tiga jenis, robot netral, robot bermuatan, dan robot utama. Titik-titik kuning adalah robot-robot bermuatan, titik-titik merah adalah robot-robot netral, dan titik ungu adalah robot utama. Gambar 3.4.2 memperlihatkan model dari ketiga jenis robot tersebut. Gambar 3.4.2 (a) adalah bentuk dari robot netral, robot ini diberi warna merah pada bagian penutupnya. Gambar 3.4.2 (b) adalah model dari robot bermuatan dengan warna kuning pada bagian penutupnya. Sedangkan Gambar 3.4.2 (c) dengan warna ungu pada bagian penutupnya. Lingkaran biru pada robot utama adalah area ketertarikan robot utama.

3.5 Tampilan Informasi Simulator

Pada bagian ini ditampilkan informasi mengenai simulasi yang ditunjukkan pada Gambar 3.5.1. Pada bagian ini dapat dilihat pengaturan dan status dari simulasi, yaitu:

1. Jumlah iterasi maksimum yang boleh dilakukan oleh Simulator.
Nilai ini dalam satuan detik. Dimana satu iterasi diasumsikan satu detik.
2. Iterasi yang sudah dilakukan sampai saat ini.
Atau dapat juga dibaca sebagai waktu yang telah dihabiskan dalam pencarian.

3. Total jumlah sumber asap yang ditentukan oleh pengguna.
4. Total jumlah sumber asap yang sudah ditemukan oleh robot pencari.
5. Banyaknya kelompok pencarian yang diturunkan.
6. Pengulangan simulasi yang telah dilakukan.

Digunakan untuk menunjukkan berapa kali pengambilan data dengan konfigurasi yang sama telah dilakukan.

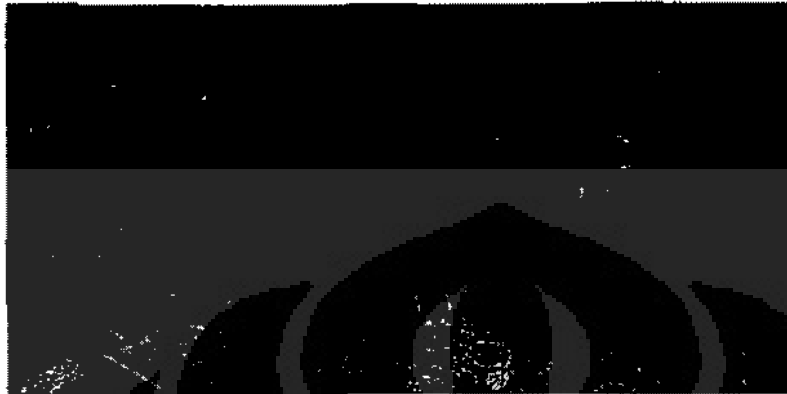
Nilai iterasi dinyatakan dalam detik dengan asumsi satu iterasi dalam simulasi adalah satu detik di dunia nyata. Nilai untuk data nomor 2, 4, dan 6 akan diperbaharui selama simulasi dijalankan.

Information	
Max Iteration	1000 sec
Current Iteration	40 sec
Source Number	3
Source Found	1
Group Number	1
Recurrent	1 from 1

Gambar 3.5.1 Informasi Simulasi

3.6 Pengaturan Kamera

Simulator dikembangkan dalam bentuk 3D. Anda dapat mengatur sudut pandang kamera dengan menggunakan tombol yang ada pada mouse. Fungsi ini baru dapat dijalankan apabila kursor berada pada bagian yang menampilkan animasi simulasi. Gambar 3.6.1 memperlihatkan tampilan simulasi setelah mengubah sudut pandang kamera.



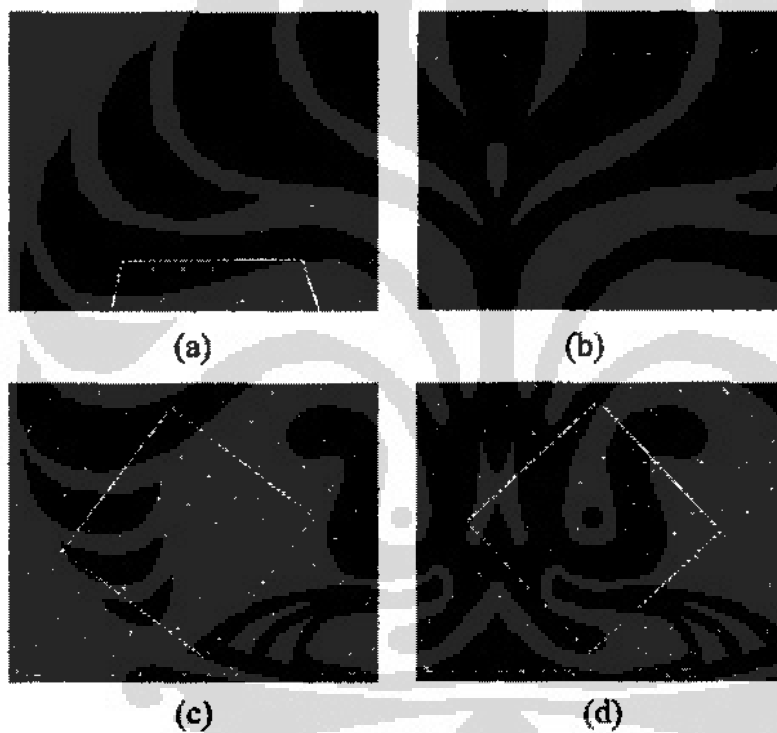
Gambar 3.6.1 Hasil Pengaturan Kamera

Untuk membantu pemahaman Anda, penjelasan berikut akan selalu merujuk pada Gambar 3.4.1. Untuk melakukan pengaturan kamera Anda dapat menggunakan tiga cara, yaitu:

- Menekan tombol sebelah kiri pada mouse.
Tekan dan tahan tombol sebelah kiri pada mouse dan gerakkan mouse Anda. Jika digerakkan ke atas maka akan menghasilkan seperti pada Gambar 3.6.3 (b), sedangkan jika digerakkan ke bawah akan seperti gambar Gambar 3.6.3 (a). Dengan menggerakkan ke atas dan ke bawah akan membuat kamera berputar yang pada akhirnya dapat menghasilkan sesuatu seperti pada Gambar 3.6.2. Gambar 3.6.3 (c) dan (d) akan didapatkan jika Anda menggerakkan mouse ke kiri dan kanan. Pergerakan ini akan memutar sudut peta.



Gambar 3.6.2 Rotasi Vertikal Pandangan

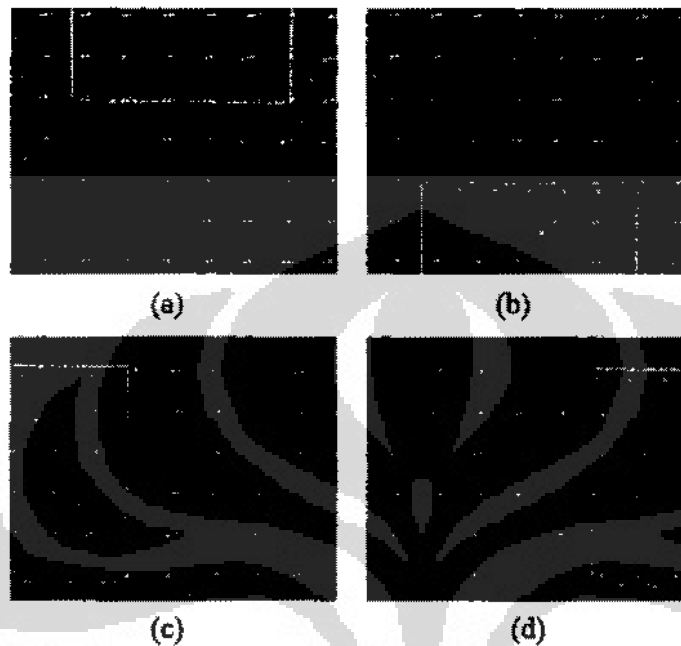


Gambar 3.6.3 Rotasi Pandangan

- Menekan tombol sebelah kanan pada mouse.

Tekan dan tahan tombol sebelah kanan pada mouse dan gerakkan mouse Anda. Secara sederhana hal ini akan membantu Anda untuk menggeser layar animasi sesuai dengan arah gerakkan mouse. Gambar 3.6.4 (a) dan (b) adalah hasil dari menggerakkan mouse ke atas dan ke bawah.

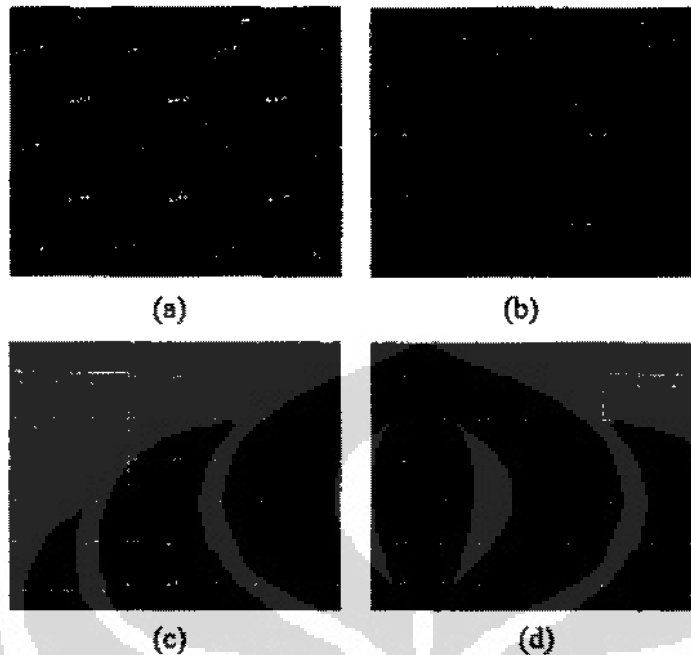
Sedangkan Gambar 3.6.4 (c) dan (d) adalah hasil menggerakkan mouse ke kiri dan ke kanan.



Gambar 3.6.4 Pergeseran Horizontal Pandangan

- Menekan tombol tengah pada mouse.

Tekan dan tahan tombol tengah pada mouse dan gerakkan mouse Anda. Hal ini akan memberikan fungsi zoom dan menggeser layar animasi. Untuk melakukan zoom-in atau memperbesar gambar, Anda cukup menggerakkan mouse ke atas, diperlihatkan pada Gambar 3.6.5 (a). Sedangkan untuk zoom-out atau memperkecil gambar, mouse digerakkan ke bawah yang hasilnya diperlihatkan pada Gambar 3.6.5 (b). Gambar 3.6.5 (c) dan (d) merupakan hasil menggerakkan mouse ke kiri dan kanan, hal ini mengakibatkan peta animasi bergeser sesuai arah pergerakan mouse.



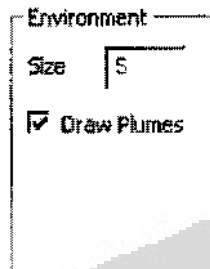
Gambar 3.6.5 Pergeseran Pandangan

3.7 Pengaturan Lingkungan

Gambar 3.7.1 menunjukkan menu untuk mengatur lingkungan simulasi. Anda dapat mengatur ukuran ruang pencarian dengan cara menuliskan angka tertentu pada bagian size. Nilai yang anda tuliskan akan mempengaruhi panjang dinding ruangan. Nilai panjang ini dalam satuan meter. Ruang pencarian akan selalu berbentuk persegi, jadi ukuran ruang tersebut akan selalu $n \times n$, atau dalam contoh Gambar 3.7.1 akan menjadi $25m^2$. Simulator tidak akan menyesuaikan ukuran ruang pencarian dalam simulasi sampai Anda menekan tombol 'Save'.

Selain ukuran ruang pencarian, pada bagian ini Anda juga dapat menentukan apakah asap yang keluar dari sumber asap dianimasikan. Jika kotak pada bagian Draw Plumes diberikan tanda \checkmark maka asap akan dianimasikan. Sebaliknya, maka asap tidak akan dimasukkan dalam animasi. Pengaturan penggambaran asap

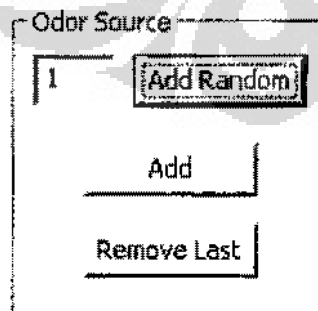
langsung berlaku tanpa perlu menekan tombol 'Save'. Bentuk animasi dari asap akan terlihat seperti pada Gambar 3.8.2.



Gambar 3.7.1 Pengaturan Lingkungan

3.8 Pengaturan Sumber Asap

Pengaturan sumber asap dapat dilakukan pada bagian yang tampak seperti pada Gambar 3.8.1. Anda hanya dapat mengatur jumlah dan posisi sumber asap pada peta animasi. Titik sumber asap ditunjukkan dengan sebuah titik berwarna ungu pada layar animasi, sedangkan bentuk animasi dari asap akan terlihat seperti pada Gambar 3.8.1. Harap diingat bahwa sumber asap yang terletak diluar dinding ruang pencarian tidak akan pernah ditemukan oleh robot. Banyaknya sumber asap yang sudah ada dalam simulator dapat dilihat pada bagian informasi, Gambar 3.5.1. Pengubahan terhadap sumber asap ini langsung akan dibaca dan diterapkan dalam simulasi. Jika perubahan dilakukan ditengah-tengah simulasi, maka simulasi akan kembali dijalankan dari awal.



Gambar 3.8.1 Pengaturan Sumber Asap

Pengaturan dapat dilakukan dengan menggunakan tiga buah tombol, yaitu:

- **Add Random**

Tombol ini akan membuat simulator meletakkan sebanyak n sumber asap dalam ruang pencarian secara acak. Letak sumber asap akan ditentukan secara acak oleh Simulator. Nilai yang terdapat disamping tombol ini adalah banyaknya sumber asap yang akan diletakkan dalam simulasi.

- **Add**

Tombol ini dapat digunakan untuk menambahkan satu sumber asap secara manual. Setelah tombol ditekan, pengguna akan diminta untuk memasukkan posisi sumber tersebut dalam koordinat X dan Y . Gambar 3.8.1 memperlihatkan form yang ditampilkan simulator agar pengguna dapat menentukan posisi sumber asap.

- **Remove Last**

Menghapus satu sumber asap dari dalam ruang pencarian. Sumber asap yang dihapus adalah sumber asap yang terakhir kali dimasukkan. Tombol ini adalah tombol untuk pasangan tombol "Add", bukan untuk pasangan tombol "Add Random". Jadi tidak bisa dipakai untuk mencabut kembali sumber asap yang ditaruh dengan menggunakan fitur "Add Random".



Gambar 3.8.2 Animasi Asap

Gambar 3.8.3 Form Penambahan Asap

3.9 Pengaturan Angin

Pengaturan angin dapat dilakukan pada bagian yang terlihat seperti pada Gambar 3.9.1. Pada bagian ini dapat diatur elemen-elemen angin dan animasi. Perubahan yang dilakukan pada pengaturan ini akan dibaca oleh Simulator setelah tombol 'Save' ditekan. Untuk lebih lengkapnya mengenai elemen-elemen pengaturan angin adalah:

Wind			
Wind Dumping	1	Disp Sigma X	1
Wind Bandwidth	1	Disp Sigma Y	1
Wind Gain	3	Disp Sigma Z	0
Wind Speed X	0.5	Puff Growth Rate	0.01
Wind Speed Y	0	Odor Num/Sec	10
Wind Speed Z	0	Timer Ratio	100

Gambar 3.9.1 Pengaturan Angin

- Wind Dumping
Kekuatan dorong angin.
- Wind Bandwidth
Besarnya ukuran lubang penangkap angin pada sensor.
- Wind Gain

Menentukan seberapa besar daya tahan dari sensor terhadap angin yang melewati sensor. Semakin besar daya tahannya maka akan semakin keras usaha yang dibutuhkan untuk membuat nilai sensor maksimum. Dengan kata lain, parameter ini menentukan sensitifitas sensor angin.

- Wind Speed X
Kecepatan angin pada sumbu X atau horizontal.
- Wind Speed Y
Kecepatan angin pada sumbu Y atau horizontal yang tegak lurus dengan sumbu X.
- Wind Speed Z
Kecepatan angin pada sumbu Z atau vertikal.
- Disp Sigma X
Tingkat ketebalan dan penampakan pada penggambaran asap untuk sumbu X atau horizontal.
- Disp Sigma Y
Tingkat ketebalan dan penampakan pada penggambaran asap untuk sumbu Y atau horizontal yang tegak lurus sumbu X.
- Disp Sigma Z
Tingkat ketebalan dan penampakan pada penggambaran asap untuk sumbu Z atau vertikal.
- Puff Growth Rate
Mengatur seberapa besar pembesaran asap yang keluar dari sumber asap. Perambahan waktu akan menyebabkan asap membesar.
- Odor Num/Sec
Berapa banyak asap yang dikeluarkan dari sumber asap.
- Timer Ratio

3.10 Pengaturan Robot Netral

Untuk mengatur robot pencarian, khususnya robot yang tidak bermuatan, dapat dilakukan pada bagian yang tampak seperti Gambar 3.10.1. Pada bagian ini dapat

diatur jumlah robot, faktor-faktor perhitungan robot, dan elemen-elemen yang berkaitan dengan PSO. Perubahan pada pengaturan ini baru akan dibaca oleh Simulator setelah tombol 'Save' ditekan. Lebih lengkapnya mengenai elemen-elemen dalam pengaturan ini adalah:

Robot	
Number	5
Initial GBest	0
Constriction Factor	0.5
C1	5
C2	5
Maximum Velocity	0.5
Maximum Iteration	1000
Max GBest	200
Closing Range	0.25
PSO Step	1
Critical Step	5
Spread Step	10
Recurrent	1
<input checked="" type="checkbox"/> WU For Ordered Area Only	

Gambar 3.10.1 Pengaturan Robot Netral

- **Number**
Elemen ini menentukan banyaknya jumlah robot netral yang digunakan dalam pencarian. Pada Gambar 3.10.1, terlihat bahwa robot netral yang digunakan sebanyak lima buah robot. Jumlah maksimal robot yang dapat diturunkan adalah 30 robot.
- **Initial Gbest**
Nilai Global Best awal (ppm). Objek setengah bola warna hijau transparan di dalam jendela simulator adalah hanya sebagai penunjuk letak Global Best, bukan nilai dari Global Best. Nilai Global Best dinyatakan dalam ppm, dan nilai itu akan terus terlihat menempel di objek setengah bola berwarna hijau transparan ini.
- **Constriction Factor**

Nilai ini digunakan pada perhitungan kecepatan PSO.

- **C1 & C2**

Dalam PSO, robot akan bergerak menurut perkalian vektor dia ke Local Best dan vektor ke Global Best. Jika nilai C1 dan C2 sama, maka hasil perkalian vektor tidak akan berat ke salah satu vektor. Tetapi jika nilai C1 yang adalah untuk Local Best lebih besar, maka arah agen akan cenderung lebih ke Local Best, dan sebaliknya jika nilai C2 yang adalah untuk Global Best lebih besar, maka arah agen akan cenderung lebih ke Global Best.

- **Maximum Felocity**

Nilai yang diberikan akan menentukan kecepatan maksimum dari robot. Nilai kecepatan ini dihitung dalam satuan m/s.

- **Maximum Iteration**

Nilai yang diberikan akan menentukan maksimal iterasi yang dapat dilakukan Simulator.

- **Max Gbest**

Ambang batas konsentrasi asap ditemukan, dalam satuan ppm.

- **Closing Range**

Dalam simulasi ini robot dapat menutup sumber asap yang telah ditemukan sehingga asap dari sumber tersebut tidak mengganggu pada pencarian sumber asap yang lain. Nilai ini akan menentukan luas daerah yang akan ditutup oleh robot agar asap tidak keluar.

- **PSO Step**

Anda dapat menentukan apakah seberapa sering perhitungan PSO dilakukan. Jika nilai yang diberikan 1 seperti pada Gambar 3.10.1, maka perhitungan PSO akan dilakukan setiap iterasinya. Simulator mengasumsikan setiap perhitungan PSO membutuhkan waktu 1 detik.

- **Critical Step**

Berapa lama waktu yang diperlukan robot untuk berpindah dari fase normal ke fase critical. Ukuran waktu dihitung dalam satuan detik.

- **Spread Step**

Berapa lama waktu yang diperlukan robot untuk berpindah dari fase critical ke fase spread. Ukuran waktu dihitung dalam satuan detik.

- **Recurrent**
Fungsi ini dapat digunakan untuk pengambilan data. Jika nilai yang diberikan pada bagian ini adalah n , maka simulasi akan dijalankan dengan konfigurasi yang sama sebanyak n kali.
- **WU for Odored Area Only**
WU adalah kepanjangan dari Wind Utilization (utilisasi angin). Jika fungsi ini diaktifkan maka utilisasi angin hanya akan dilakukan jika robot sudah menemukan zat yang dicari dalam udara.

3.11 Pengaturan Robot Bermuatan

Pengaturan robot bermuatan dapat dilakukan pada bagian yang tampak seperti pada Gambar 3.11.1. Perubahan pada pengaturan ini baru dibaca setelah tombol 'Save' ditekan. Elemen-elemen pada pengaturan ini adalah:

Charged Robot	
Q	1
R Core	0.5
R Limit	1
Number	5

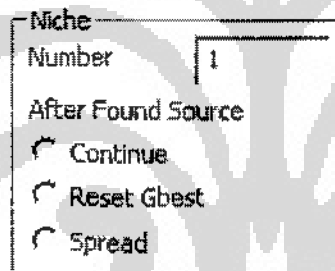
Gambar 3.11.1 Pengaturan Robot Bermuatan

- **Q**
Menyatakan besarnya muatan pada robot bermuatan. Nilai muatan dalam satuan coulomb.
- **R Core**
Menyatakan nilai radius inti pada robot bermuatan. Radius inti ini yang akan mempengaruhi kapan gaya tolak-menolak atau tarik-menarik dari muatan robot berlaku. Radius dinyatakan dalam satuan meter.

- R Limit
Menyatakan nilai radius luar pada robot bermuatan. Nilai radius luar ini dinyatakan dalam satuan meter.
- Number
Mengatur jumlah robot bermuatan yang digunakan dalam pencarian.

3.12 Pengaturan Kelompok Pencarian

Untuk melakukan pengaturan kelompok pencarian dapat dilakukan pada bagian yang terlihat seperti pada Gambar 3.12.1. Perubahan bagian ini baru akan dibaca oleh Simulator setelah tombol 'Save' ditekan. Elemen-elemen dalam pengaturan ini adalah:



Niche
Number 1
After Found Source
 Continue
 Reset Gbest
 Spread

Gambar 3.12.1 Pengaturan Kelompok Pencarian

- Number
Nilai yang diberikan akan menentukan banyaknya jumlah kelompok robot pencarian. Setiap kelompok pencarian akan memiliki kombinasi robot netral dan robot bermuatan yang sama. Misalnya jika jumlah robot netral ditentukan 5 buah seperti **Error! Reference source not found.** dan jumlah robot bermuatan 1 seperti pada Gambar 3.11.1. Jika dengan susunan tersebut dibuat dua kelompok pencarian, maka akan terdapat 2 buah kelompok dengan 5 buah robot netral dan 1 buah robot bermuatan. Maka total akan terdapat 10 robot netral dan 2 robot bermuatan yang diikutsertakan dalam pencarian.
- After Found Source

Mengatur tindakan yang akan dilakukan kelompok pencarian setiap kali menemukan sumber asap. Ada tiga hal yang dapat dipilih, yaitu Continue, Reset Gbest, dan Spread. Jika dipilih Continue, maka kelompok pencarian akan melakukan pencarian seperti biasa. Jika yang dipilih Reset Gbest, maka nilai dan posisi Global Best kelompok pencarian yang menemukan sumber akan di reset. Jika yang dipilih Spread, maka robot-robot pada niche yang menemukan sumber tersebut akan masuk pada fase Spread.

3.13 Pengaturan Robot Utama

Pengaturan robot utama dapat dilakukan pada bagian yang tampak seperti pada Gambar 3.13.1. Perubahan pada pengaturan ini baru dibaca setelah tombol 'Save' ditekan. Elemen-elemen pada pengaturan ini adalah:


Main Robot	
Q	1
R Core	0.1
Attract Radius	0.5

Gambar 3.13.1

- Q
Menyatakan besarnya muatan pada robot utama. Nilai muatan dalam satuan coulomb.
- R Core
Menyatakan nilai radius inti pada robot utama. Radius inti ini yang akan mempengaruhi kapan gaya tolak-menolak atau tarik-menarik dari muatan robot berlaku. Radius dinyatakan dalam satuan meter.
- Attract Radius
Menyatakan nilai radius area ketertarikan pada robot utama. Nilai radius luar ini dinyatakan dalam satuan meter.

3.14 Pengaturan Rasio Kesalahan Sensor

Pengaturan rasio kesalahan sensor dapat dilakukan pada bagian seperti pada Gambar 3.14.1. Tujuan rasio kesalahan sensor adalah membuat simulasi semakin nyata. Dengan turut diperhitungkannya kesalahan dalam pembacaan sensor, Anda dapat memperkirakan perbaikan yang perlu dilakukan dan perangkat keras yang harus digunakan untuk mendapatkan hasil pencarian terbaik. Hal-hal yang dapat diatur dalam rasio kesalahan adalah:



Error	
GPS	0
Sensor	0.2

Gambar 3.14.1 Pengaturan Rasio Kesalahan Sensor

- **GPS**
Nilai kesalahan ketika membaca posisi robot melalui GPS.
- **Sensor**
Nilai kesalahan yang mungkin terjadi pada sensor, dalam satuan ppm.

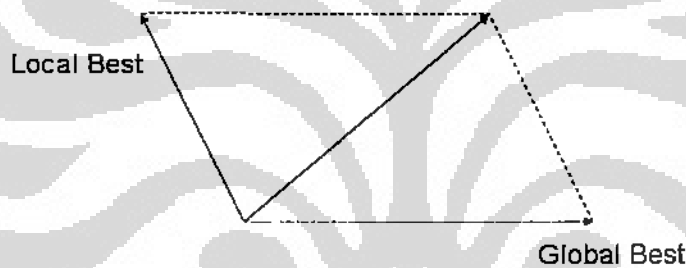
Kedua Nilai tersebut akan selalu mempengaruhi setiap kali robot membaca informasi dari GPS atau Sensor.

BAB 4 DYNAMIC NICHE-PSO

Bab ini akan menjelaskan tentang algoritma Dynamic Niche-PSO. Dimulai dari kekurangan dari algoritma sebelumnya. Lalu pengenalan robot utama, dan diakhiri dengan algoritma Dynamic Niche-PSO itu sendiri.

4.1 Algoritma Original

Dalam PSO, agen atau robot akan bergerak menurut penjumlahan vektor dari posisi pada saat itu menuju ke arah local best dan vektor yang menuju ke arah global best seperti terlihat pada Gambar 4.1.1. Masing-masing dari vektor ini mempunyai koefisien sosial. c_1 untuk local best dan c_2 untuk global best seperti terlihat pada Rumus 4.1.1 dan Rumus 4.1.2.



Gambar 4.1.1 Pergerakan dalam PSO

Pada algoritma sebelumnya niche terdiri dari 2 macam robot, yaitu robot netral dan robot bermuatan. Rumus pergerakan dari kedua jenis robot ini pada dasarnya sama. Hal yang sama adalah keduanya bergerak berdasar informasi local best dan global best. Yang berbeda hanya pada robot bermuatan ada gaya saling tolak antar sesama robot bermuatan. Gaya tolak ini bertujuan untuk meningkatkan sifat divergen dari algoritma.

Robot Netral

$$\mathbf{V}_i^{n+1} = \chi \left(\mathbf{V}_i^n + \underbrace{c_1 \cdot \text{rand}() \cdot (\mathbf{p}_i^n - \mathbf{x}_i^n)}_{\text{Individual}} + \underbrace{c_2 \cdot \text{Rand}() \cdot (\mathbf{p}_g^n - \mathbf{x}_i^n)}_{\text{Sosial}} \right)$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \mathbf{V}_i^{n+1}$$

Rumus 4.1.1 Pergerakan Robot Netral

Robot Bermuatan

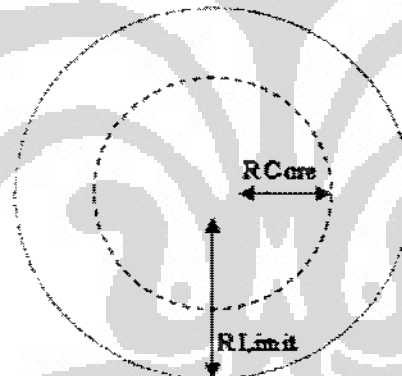
$$\mathbf{V}_i^{n+1} = \chi \left(\mathbf{V}_i^n + c_1 \cdot \text{rand}() \cdot (\mathbf{p}_i^n - \mathbf{x}_i^n) + c_2 \cdot \text{Rand}() \cdot (\mathbf{p}_g^n - \mathbf{x}_i^n) \right) + \mathbf{a}_i$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \mathbf{V}_i^{n+1}$$

Individual
Sosial
Gaya Tolak

Rumus 4.1.2 Pergerakan Robot Bermuatan

Gaya tolak antar robot bermuatan akan menguat apabila semakin dekat seperti ditunjukkan oleh Gambar 4.1.2 di bawah. Pada bab BAB 3 subbab 3.11 ditunjukkan R Core dan R Limit. R Core adalah ditunjukkan oleh lingkaran bagian dalam, dan R Limit oleh yang bagian luar.



Gambar 4.1.2 Robot Bermuatan

$$a_{ij}(t) = \begin{cases} \frac{Q_i Q_j (x_i(t) - x_j(t))}{r_{core}^2 |x_i(t) - x_j(t)|} & \text{jika } |x_i(t) - x_j(t)| < r_{core} \\ \frac{Q_i Q_j (x_i(t) - x_j(t))}{|x_i(t) - x_j(t)|^3} & \text{jika } r_{core} < |x_i(t) - x_j(t)| < r_{limit} \\ 0 & \text{jika } r_{limit} < |x_i(t) - x_j(t)| \end{cases}$$

Rumus 4.1.3 Gaya Tolak Robot Bermuatan

Satu lagi metode dari algoritma original ini yang meningkatkan sifat divergen adalah spread. Spread adalah sesuai dengan artinya, adalah penyebaran partikel-partikel niche. Spread akan terjadi apabila :

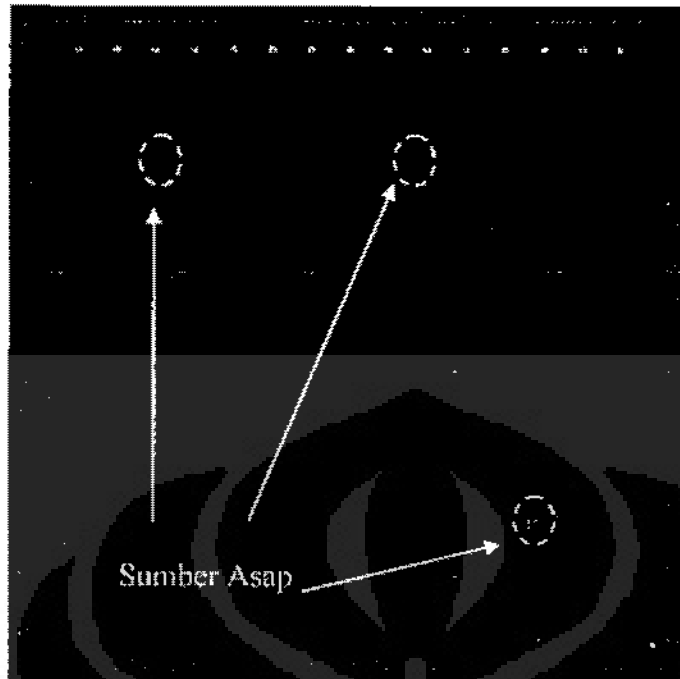
1. Global best niche bertubrukan dengan global best niche lain.
2. Nilai global best niche tidak berubah selama iterasi tertentu.

Apabila global best 2 niche bertubrukan, maka niche dengan nilai global best lebih rendah akan mengalah, dan melakukan fase spread. Spread dilakukan dengan masing-masing robot bergerak ke arah luar dari titik tengah niche. Titik tengah niche adalah titik tengah segi banyak dimana titik sudut dari segi banyak ini adalah robot itu sendiri.

Niche mempunyai 3 fase yaitu fase PSO, fase kritikal, dan fase spread. Tiap fase mempunyai batas iterasi tertentu. Apabila nilai global best tidak berubah selama batas iterasi ini, maka misalkan sebuah niche yang tadinya sedang dalam fase PSO, dia akan berpindah ke fase kritikal. Lalu di sini, jika nilai global best masih tidak berubah, maka niche akan berpindah ke fase spread.

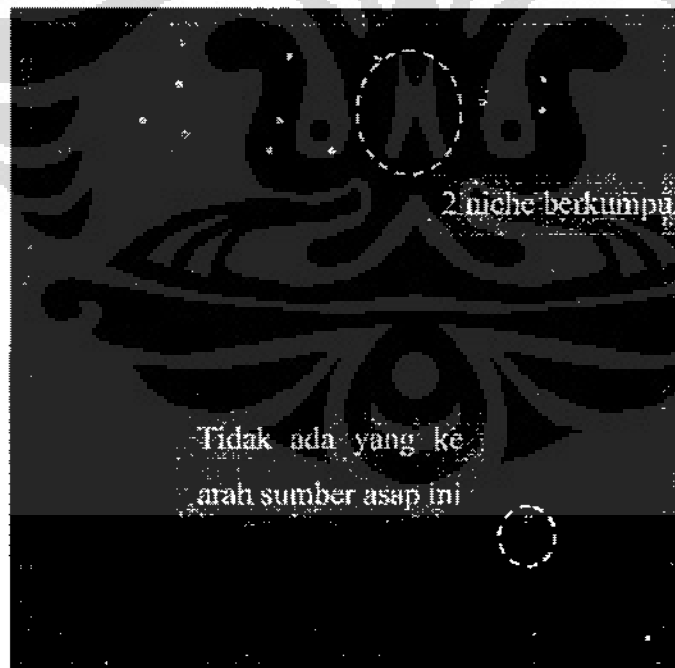
4.2 Kelemahan Algoritma Original

Untuk menjelaskan kelemahan dari algoritma originan akan dipakai contoh di bawah. Gambar 4.2.1 menunjukkan ada 3 niche, dan 3 sumber asap. 1 sumber asap letaknya jauh dari 2 yang lain yang berdekatan. 2 sumber asap yang berdekatan ini letaknya dekat dengan posisi awal robot. Letak global best ditunjukkan oleh bola hijau transparan. Karena ada 3 niche maka ada 3 global best, karena masing-masing niche mempunyai satu global best..



Gambar 4.2.1 PSO Algoritma Original (Keadaan Awal)

Setelah beberapa waktu, keadaan ditunjukkan oleh Gambar 4.2.2 di bawah.



Gambar 4.2.2 PSO Algoritma Original (Hasil)

Kita lihat bahwa 2 niche yang tengah dan yang kanan sama-sama menuju ke arah sumber asap yang di tengah. Ini dikarenakan niche mengandalkan global best untuk bertubrukan dengan global best niche lain untuk memasuki fase spread. Bilamana global best tidak bertubrukan maka tidak memasuki fase spread.

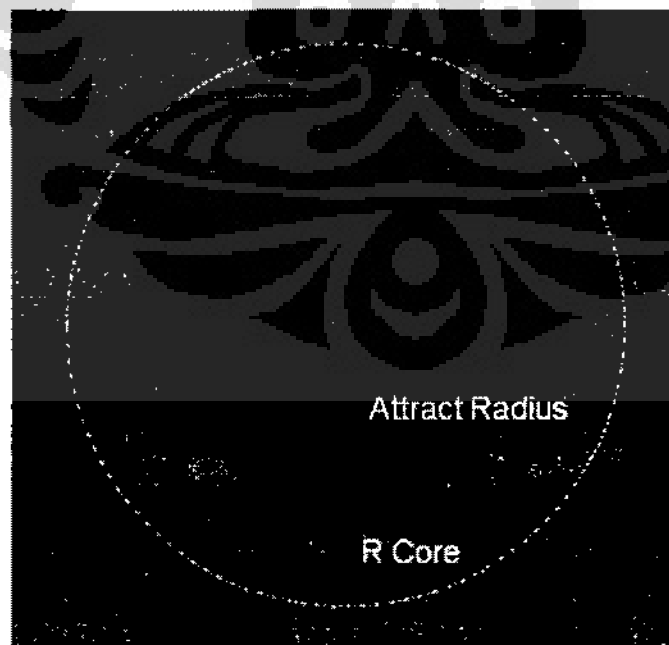
4.3 Dynamic Niche-PSO

Algoritma baru yang diajukan pada tesis ini adalah Dynamic Niche-PSO. Update dari algoritma sebelumnya adalah adanya perpindahan keanggotaan robot dari satu niche ke niche yang lain dan adanya robot baru yaitu robot utama.

4.3.1 Robot Utama

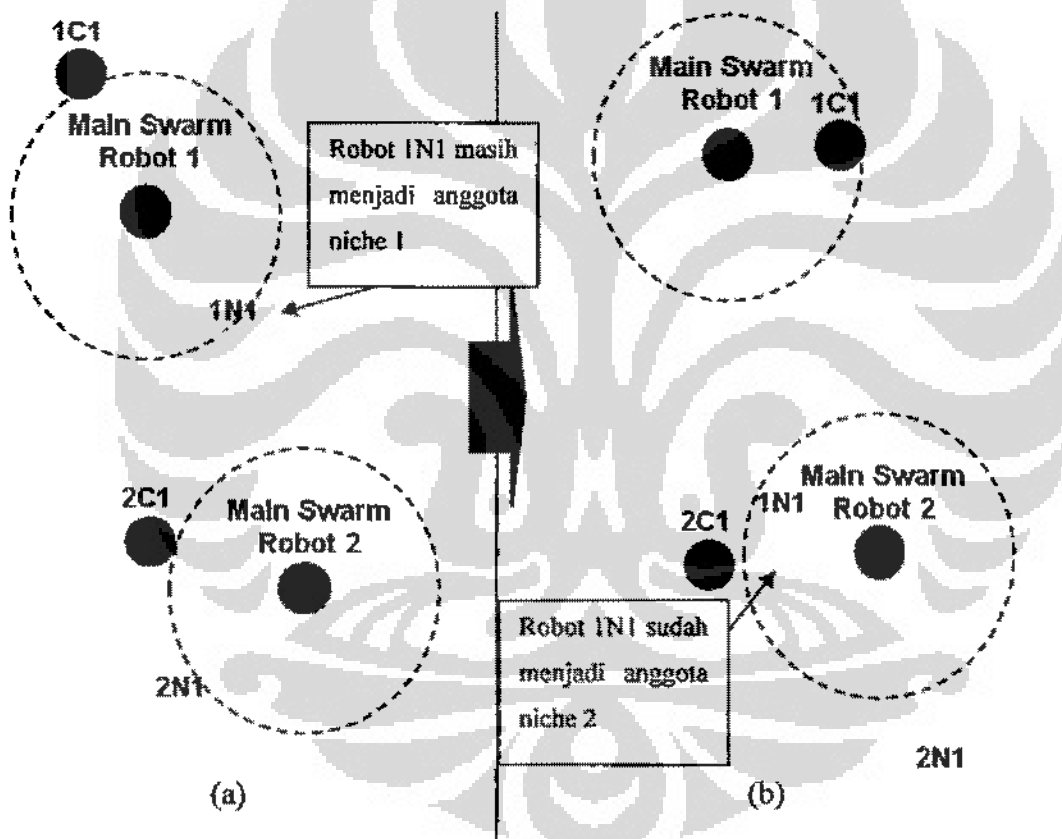
Robot Utama memiliki ciri-ciri sebagai berikut :

1. Mempunyai area ketertarikan
2. Mempunyai gaya tolak-menolak antar robot utama
3. Hanya memiliki informasi local best



Gambar 4.3.1 Robot Utama

Robot utama mempunyai area ketertarikan dimana robot netral atau bermuatan dari niche lain yang memasuki area ini akan menjadi anggota niche robot utama ini. Dengan kata lain, robot netral atau bermuatan tersebut berpindah keanggotaan. Robot utama tidak dapat berpindah keanggotaan, dan jumlah robot utama untuk setiap niche adalah 1. Radius area ketertarikan dapat diubah-ubah melalui GUI. Tetapi radius area ketertarikan tidak dapat diubah di tengah-tengah simulasi sedang berlangsung. Gambar penjelasan mengenai perpindahan keanggotaan robot ditunjukkan oleh Gambar 4.3.2.



Gambar 4.3.2 Perpindahan Keanggotaan

Area ketertarikan robot utama ditunjukkan dengan lingkaran dengan garis terputus-putus. Robot utama ditunjukkan oleh lingkaran berwarna ungu, robot netral oleh yang berwarna merah, dan robot bermuatan oleh yang berwarna kuning seperti pada simulator. Gambar (a) adalah keadaan sebelum dan (b) adalah

gambar sesudah. Pada Gambar 4.3.2 (a) tabel keanggotaan niche ditunjukkan oleh Tabel 4.3.1 di bawah.

Tabel 4.3.1 Keanggotaan Niche (Sebelum Berpindah)

Niche I	Niche II
Main Swarm Robot 1	Main Swarm Robot 2
1C1	2C1
1N1	2N1

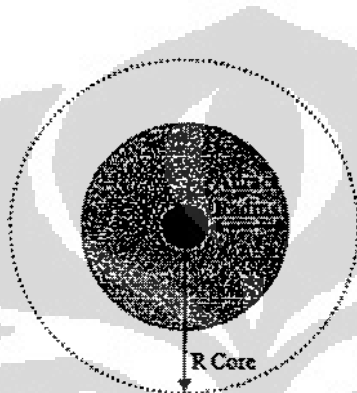
Robot-robot bergerak sedemikian sehingga menjadi keadaan pada Gambar 4.3.2 (b) dimana robot 1N1 masuk ke dalam area ketertarikan robot utama niche II (Main Swarm Robot 2). Maka robot 1N1 berubah keanggotaan dan menjadi anggota niche II sehingga update tabel keanggotaan niche menjadi seperti yang ditunjukkan oleh di bawah.

Tabel 4.3.2 Keanggotan Niche (Sesudah Berpindah)

Niche I	Niche II
Main Swarm Robot 1	Main Swarm Robot 2
1C1	2C1
	2N1
	1N1

Robot utama juga memiliki gaya tolak menolak antar sesama robot utama. Tujuannya adalah untuk meningkatkan sifat divergen. Karena robot utama adalah penentu arah pergerakan niche, diharapkan tiap niche menuju sumber asap yang berbeda. Sebagai batas penolakan, robot utama mempunyai radius penolakan seperti pada robot bermuatan. Tetapi pada robot utama hanya terdapat R Core.

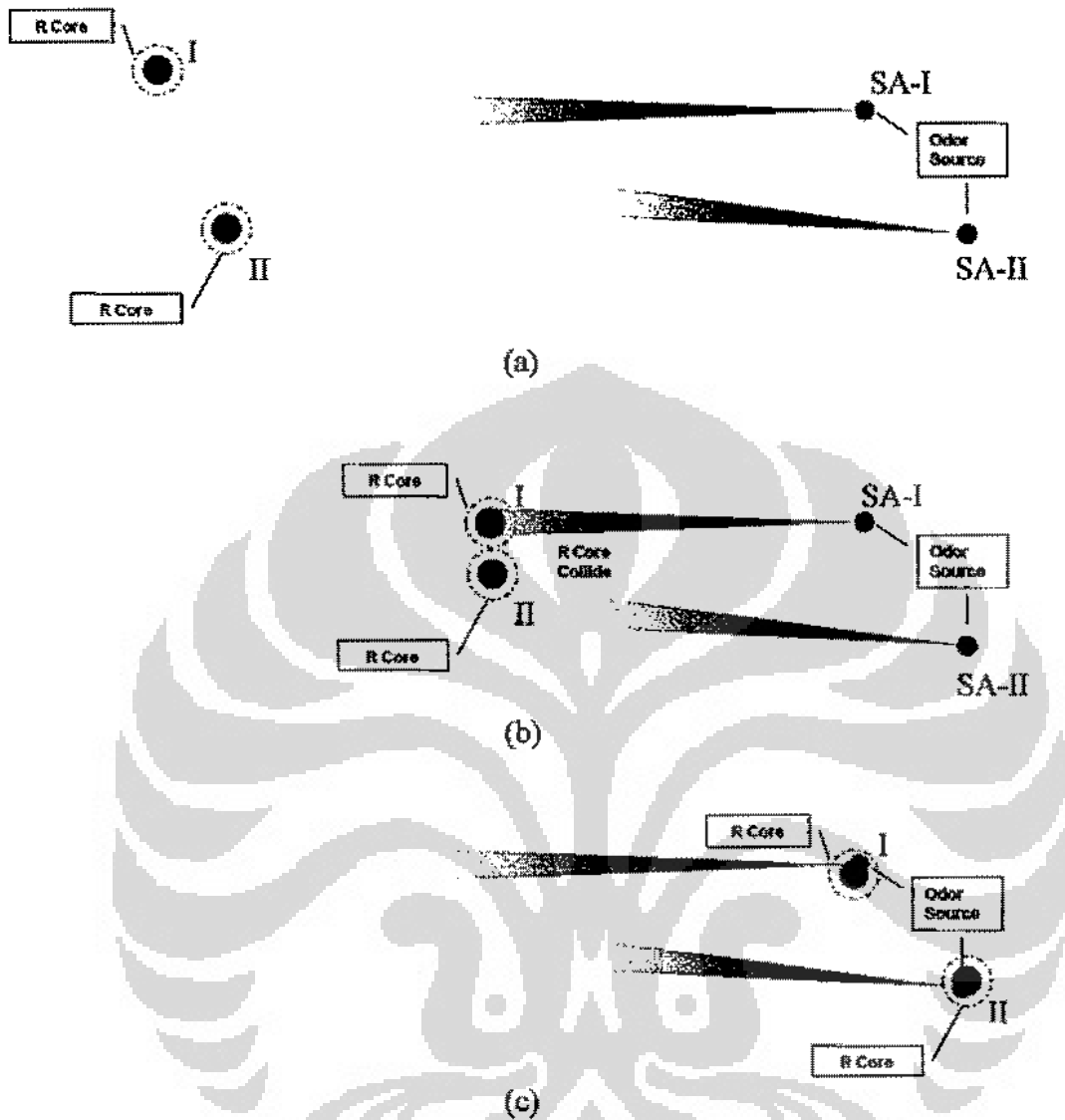
Pada robot bermuatan ada penolakan yang kuat dan yang lemah seperti ditunjukkan oleh Rumus 4.1.3. Untuk robot utama hanya ada antara adanya penolakan dan tidak adanya penolakan. Gaya tolak robot utama mengambil dari gaya tolak robot beruatan yang kuat seperti ditunjukkan oleh Rumus 4.3.1. Jarak penolakan adalah jarak ketika R Core mulai beririsan dengan area ketertarikan seperti terlihat pada Rumus 4.3.1 dimana jaraknya adalah $2 \times AttractRadius + RCore$.



Gambar 4.3.3 Robot Utama - R Core

$$u_{Rk}(t) = \begin{cases} \frac{Q_j Q_k (x_j(t) - x_k(t))}{(AttractRadius + RCore)^2 \|x_j(t) - x_k(t)\|} & \text{jika } \|x_j(t) - x_k(t)\| < 2 \times AttractRadius + RCore \\ 0 & \text{jika } \|x_j(t) - x_k(t)\| > 2 \times AttractRadius + RCore \end{cases}$$

Rumus 4.3.1 Gaya Tolak Robot Utama



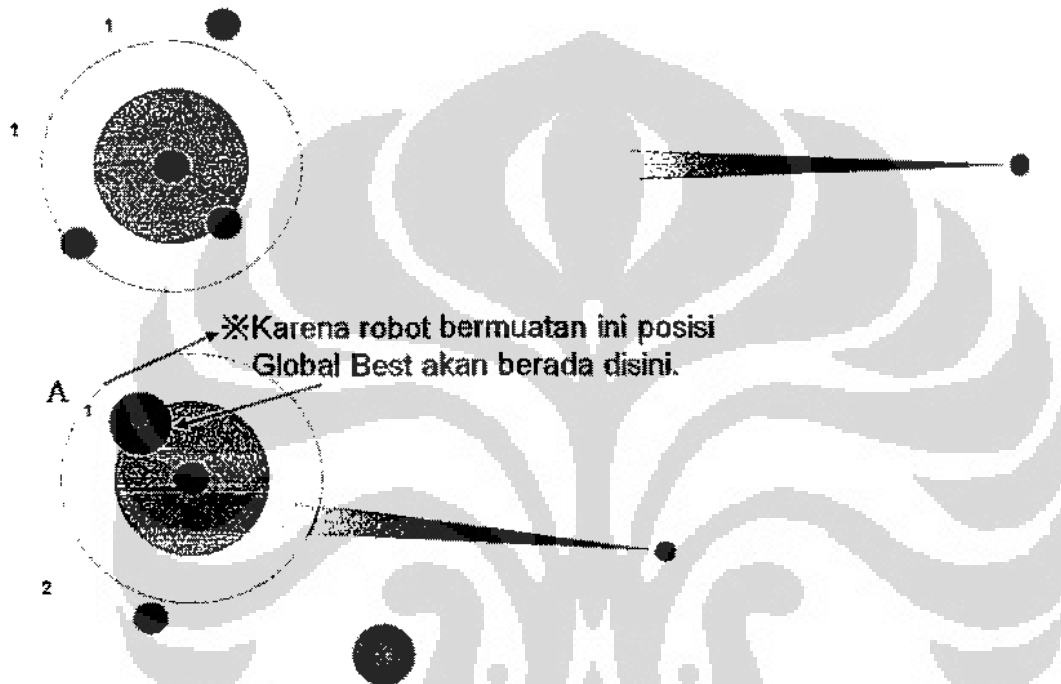
Gambar 4.3.4 Gaya Tolak Robot Utama

Pada Gambar 4.3.4 lingkaran berwarna ungu adalah robot utama, dan lingkaran dengan garis terputus-putus adalah R Core dari robot utama. Lingkaran berwarna merah muda gelap di sebelah kanan adalah sumber asap (SA-I dan SA-II), dan segitiga yang warnanya degradasi dari arah sumber asap ini adalah asap.

Posisi awal 2 robot utama ditunjukkan oleh Gambar 4.3.4 (a), dimana mereka akan bergerak sedemikian sehingga saling mendekati seperti ditunjukkan oleh Gambar 4.3.4 (b). Karena robot utama saling tolak menolak, maka robot utama I akan

menolak robot utama II. Sehingga robot utama II tidak akan bergerak menuju sumber asap SA-I, dan diharapkan bertemu dengan sumber asap lain seperti ditunjukkan oleh Gambar 4.3.4 (c).

4.3.2 Pengaruh Perpindahan Keanggotaan

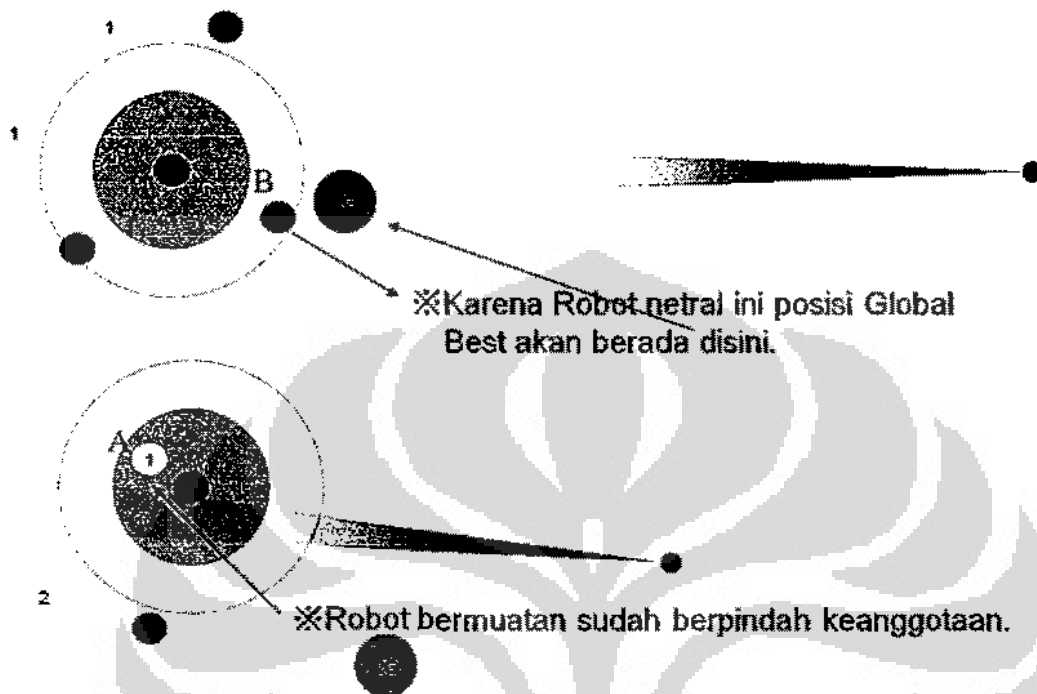


Gambar 4.3.5 Pengaruh Perpindahan Keanggotaan (Keadaan awal)

Pada Gambar 4.3.5, penomoran robot adalah nomor niche, jadi hanya ada 2 niche di sini. Robot berwarna ungu adalah robot utama, kuning adalah robot bermuatan, dan merah adalah robot netral. Sedangkan hijau adalah posisi global best dimana 1G adalah posisi global best untuk niche 1 dan 2G adalah posisi global best untuk niche 2.

Pada Gambar 4.3.5 posisi global best niche 1 akan pada posisi yang ditunjukkan oleh 1G didapat dari robot bermuatan niche 1 (robot berwarna kuning) yang ada pada posisi yang ditunjukkan oleh A. Karenanya, robot-robot lain anggota niche 1 akan mempunyai elemen pergerakan global best ke arah global best 1G ini.

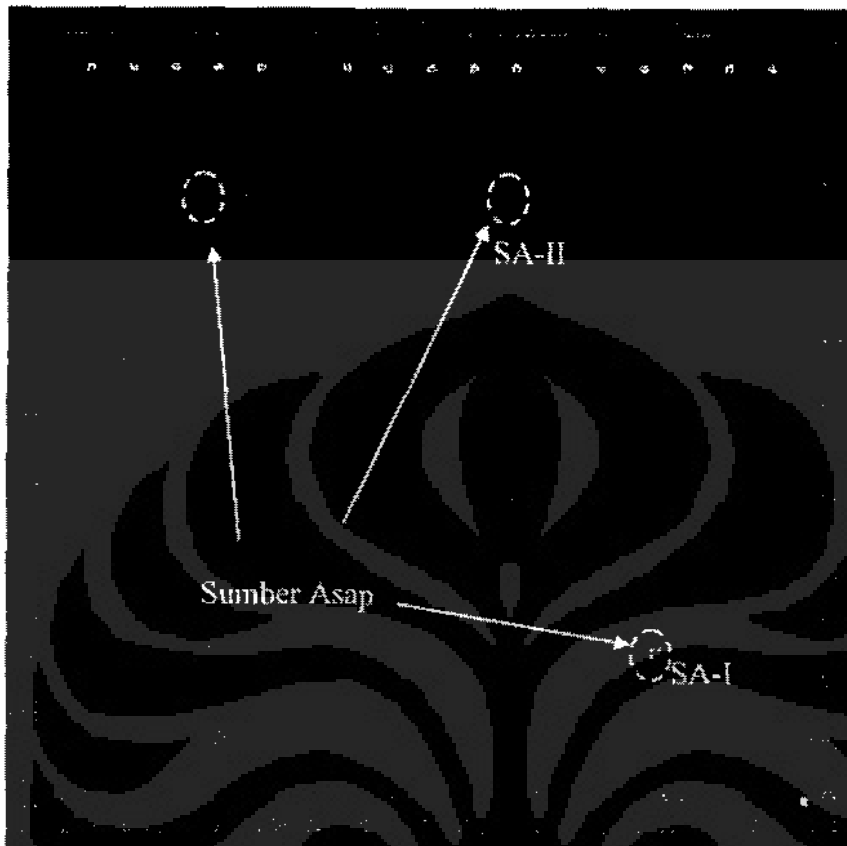
Padahal di sini seharusnya niche 1 mengarah ke sumber asap yang diatas, bukan yang di bawah.



Gambar 4.3.6 Pengaruh Perpindahan Keanggotaan (Hasil)

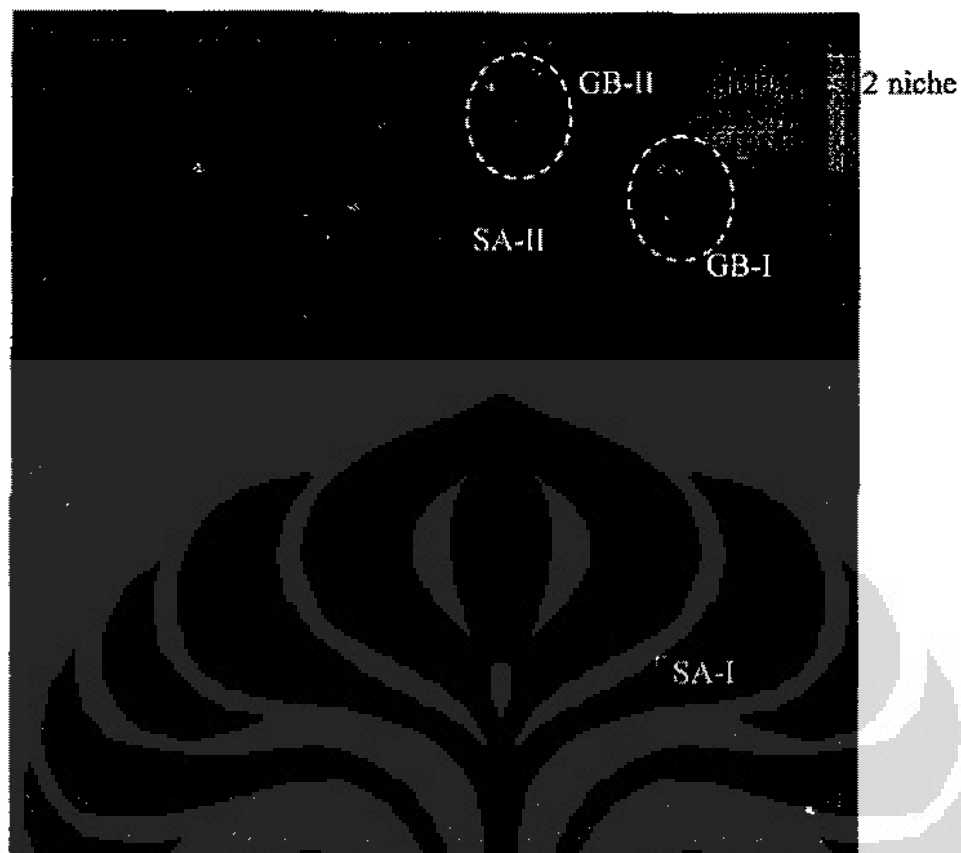
Pada Gambar 4.3.6, robot bermuatan yang ditunjukkan oleh A, sudah bukan lagi anggota niche 1 karena telah masuk ke dalam area ketertarikan robot utama niche 2, dan robot bermuatan ini menjadi anggota niche 2. Maka dari itu nilai local best nya sudah tidak dipedulikan lagi oleh niche 1. Sekarang niche 1 mendapat nilai global best yang baru oleh robot yang ditunjukkan oleh B, yaitu global best IG. Dengan nilai global best yang baru ini niche 1 akan mengarah ke sumber asap yang ada di atas.

4.4 Hasil



Gambar 4.4.1 Dynamic Niche-PSO (Keadaan Awal)

Gambar 4.4.1 adalah gambar keadaan awal dari percobaan Dynamic Niche-PSO. Keadaan awal ini disamakan dengan keadaan awal percobaan algoritma original pada subbab 4.2 untuk perbandingan. Hasil dari percobaan Dynamic Niche-PSO ditunjukkan oleh di bawah.



Gambar 4.4.2 Dynamic Niche-PSO (Hasil)

Dalam Gambar 4.4.2, yang menjadi perhatian adalah 2 global best yang dilingkari dengan lingkaran dengan garis terputus-putus (GB-I dan GB-II). 2 global best ini terpisah seperti yang diharapkan. Lalu GB-I bergerak menuju SA-I, dan GB-II menuju SA-II. Pada percobaan ini rata-rata total waktu pencarian adalah 272 detik sedangkan dengan algoritma MPSO yang mencapai 327 detik dari masing-masing 3 percobaan.

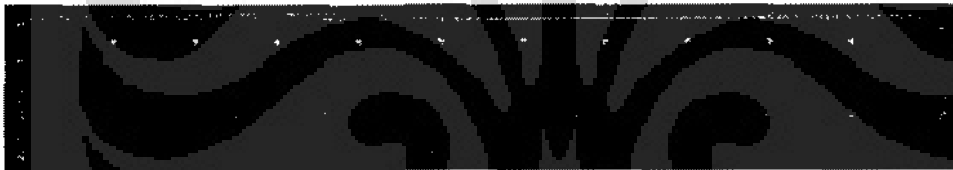
4.5 Analisis

Dari percobaan dijelaskan diatas ditambahkan percobaan untuk menambah pembuktian bahwa algoritma *Dynamic Niche-PSO* lebih baik dari algoritma MPSO.

Percobaan akan dilakukan untuk luas medan $10 \times 10 m^2$ dan $12 \times 12 m^2$ dan masing-masing dicobakan dengan 2 *niche* dan 5 *niche*. Jumlah robot netral dan robot bermuatan adalah 20. 10 robot netral dan 10 robot bermuatan. Jadi untuk percobaan dengan 2 *niche* pada masing-masing *niche* terdapat 5 robot netral dan 5 robot utama seperti ditunjukkan oleh Gambar 4.5.1 dan Gambar 4.5.2, dan untuk percobaan dengan 5 *niche* pada masing-masing *niche* terdapat 2 robot netral dan 2 robot bermuatan seperti ditunjukkan oleh Gambar 4.5.1 dan Gambar 4.5.3.

Tabel 4.5.1 Jumlah Robot/Niche

	MPSO		<i>Dynamic Niche-PSO</i>	
	2 <i>niche</i>	5 <i>niche</i>	2 <i>niche</i>	5 <i>niche</i>
Robot Bermuatan	5	2	5	2
Robot netral	5	2	5	2



Gambar 4.5.1 Susunan Robot (MPSO)



Gambar 4.5.2 Susunan Robot (*Dynamic Niche-PSO*, 2 *niche*)



Gambar 4.5.3 Susunan Robot (*Dynamic Niche-PSO*, 5 *niche*)

Lokasi sumber-sumber asap ditunjukkan oleh Tabel 4.5.2.

Tabel 4.5.2 Lokasi Sumber

Sumber Gas	Koordinat (x,y)
Sumber 1	(6,5)
Sumber 2	(5,7)
Sumber 3	(5,3)
Sumber 4	(4,9)
Sumber 5	(4,1)
Sumber 6	(3,5)

Banyaknya kombinasi yang dijalankan adalah $\sum_{\text{var isi luas area}} x \sum_{\text{var isi jumlah niche}}$

Jadi banyaknya kombinasi adalah $2 \times 2 = 4$ kombinasi.

Hasil dari percobaan ditunjukkan oleh Tabel 4.5.3 dan Tabel 4.5.4.

Tabel 4.5.3 Ujicoba Algoritma MPSO

Ujicoba ke-	Ujicoba Algoritma MPSO			
	$10 \times 10 m^2$		$12 \times 12 m^2$	
	2 Niche	5 Niche	2 Niche	5 Niche
1	549	516	1167	855
2	586	534	1097	843
3	-	-	1231	-
Rata-rata	567.5	525	1265.33	849

Tabel 4.5.4 Ujicoba Algoritma *Dynamic Niche-PSO*

Ujicoba ke-	Ujicoba Algoritma <i>Dynamic Niche-PSO</i>			
	$10 \times 10 m^2$		$12 \times 12 m^2$	
	<i>2 Niche</i>	<i>5 Niche</i>	<i>2 Niche</i>	<i>5 Niche</i>
1	475	316	1423	849
2	346	421	1283	831
3	-	-	948	-
Rata-rata	410.5	368.5	1231	840

Dari hasil dapat disimpulkan bahwa algoritma *Dynamic Niche-PSO* lebih cepat dari algoritma MPSO. Pada percobaan dengan luas medan $12 \times 12 m^2$ didapat hasil dimana algoritma *Dynamic Niche-PSO* lebih lambat. Setelah dilakukan analisis, ketika sumber tinggal 1, dan 2 robot utama atau lebih berada pada jarak yang sama dengan sumber, maka robot-robot utama ini akan bertabrakan terus yang akan menghalangi jalu robot-robot yang lain untuk mendekati sumber asap. Jika tidak menemui keadaan seperti ini, hasil dari algoritma *Dynamic Niche-PSO* lebih baik dari algoritma MPSO secara signifikan seperti ditunjukkan oleh percobaan ke-3.

BAB 5 PENUTUP

Bab ini akan menjelaskan tentang kesimpulan dan saran yang diperoleh melalui kegiatan penelitian tugas akhir ini. Pada bagian kesimpulan, penulis akan menyampaikan rangkuman hasil yang telah dicapai pada kegiatan penelitian yang dilakukan. Pada bagian saran, penulis menyampaikan usulan penelitian yang dapat dilaksanakan pada tahap pengembangan yang selanjutnya.

5.1 Kesimpulan

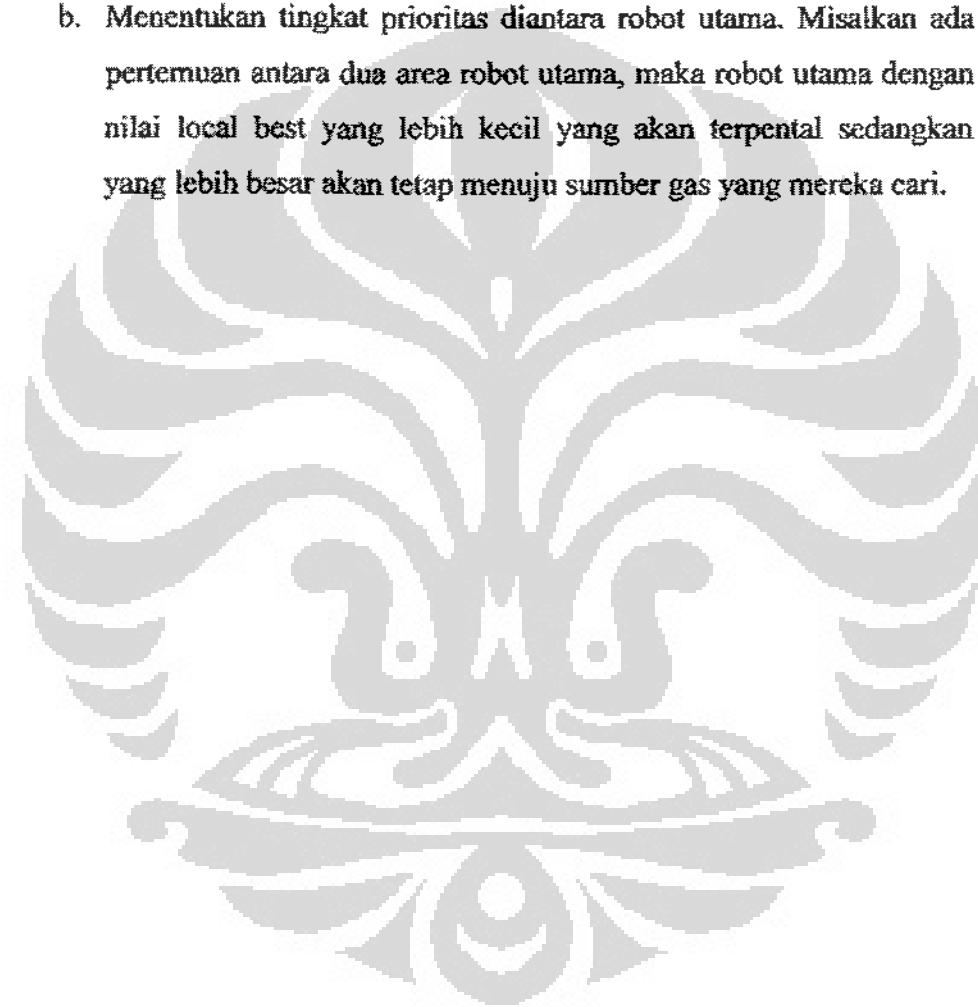
1. Setelah mengimplementasikan robot dan lingkungannya dalam simulasi 3D termasuk model fisik, diaplikasikan algoritma PSO untuk mencari banyak sumber asap dalam lingkungan yang dinamis. Hasilnya menunjukkan bahwa PSO bisa diaplikasikan di perangkat keras yang sesungguhnya. Dimana perangkat keras di sini adalah robot.
2. Dynamic Niche-PSO yang diajukan pada penelitian kali ini menunjukkan hasil yang lebih bagus dari algoritma sebelumnya. Robot-robot lebih tersebar dalam mencari sumber asap (peningkatan sifat divergen) seperti dijelaskan pada subbab 4.4. Waktu pencarian juga lebih cepat seperti dijelaskan pada subbab 4.5.

5.2 Saran

1. Perlu dilakukan pengecekan performa yang lebih lanjut. Yaitu pengecekan dengan berbagai macam faktor diantaranya jumlah robot netral dan robot bermuatan, variasi letak sumber asap yang lain dari yang ditunjukkan oleh Gambar 4.4.1 pada subbab 4.4, variasi radius area ketertarikan dan besar muatan robot utama.
2. Salah satu cara yang terpikirkan untuk mengatasi masalah saling bertabrakannya dua kelompok niche atau lebih pada algoritma *Dynamic Niche-PSO* adalah perlu ditambahkan metode adaptif untuk menentukan

area ketertarikan yang dimiliki sebuah robot utama. Syarat adaptif yang terpikirkan ada dua, yaitu:

- a. Menentukan jeda waktu dimana dua area robot utama dianggap tidak wajar untuk bertemu kembali, apabila dalam jeda waktu yang diberikan ternyata terjadi pengulangan maka area ketertarikan dari dua robot utama tersebut akan dikecilkan.
- b. Menentukan tingkat prioritas diantara robot utama. Misalkan ada pertemuan antara dua area robot utama, maka robot utama dengan nilai local best yang lebih kecil yang akan terpental sedangkan yang lebih besar akan tetap menuju sumber gas yang mereka cari.



DAFTAR PUSTAKA

- [1] Nugraha, Aditya, Pencarian Banyak Sumber Asap dengan Modifikasi Particle Swarm Optimization Berbasis Graphical User Interface, Skripsi FASILKOM, Universitas Indonesia, 2008
- [2] Kousei Demura, *Easy! Practical! Robot Simulation*, Morikita Press, 2007
- [3] Kousei Demura, RoboCup ODE Reference, <http://demura.net>
- [4] Russel Smith, *Open Dynamics Engine V.05 User Guide*, 2004
- [5] Wisnu Jatmiko, Kosuke Sekiyama, Toshio Fukuda, A PSO-Based Mobile Robot for Odor Source Localization in Dynamic Advection-Diffusion with Obstacles Environment: Theory, Simulation, and Measurement, IEEE Computational Intelligence Magazine, page 37-51, May 2007
- [6] W. Jatmiko, A. Nugraha, W. Pambuko, B. Kusumoputro, K. Sekiyama, T. Fukuda, Localizing Multiple Odor Sources in Dynamic Environment using Niche PSO with Flow of Wind Based on Open Dynamics Engine Library, paper presented in The Second International Conference on IT Application and Management, University of Indonesia, 2009
- [7] Engelbrecht, Andries P., "Fundamentals of Computational Swarm Intelligence", Wiley, 2005
- [8] Aditya Nugraha, W. Jatmiko, "Ranged Global Best in Parallel PSO Niching for Multiple Odor Sources Localization Problem", 2008
- [9] Aditya Nugraha, W. Jatmiko, J. Perkasa, "Modifikasi *Particle Swarm Optimization* untuk Pencarian Banyak Sumber Gas", Jurnal Ilmu Komputer dan Informasi, July 2008