UNIVERSITAS INDONESIA

DYNAMIC MONTE CARLO LOCALIZATION UNTUK TRACKED MOBILE ROBOT

SKRIPSI

VEKTOR DEWANTO
0606029492

FAKULTAS TEKNIK
PROGRAM SARJANA
DEPOK
JUNI 2010

UNIVERSITAS INDONESIA

DYNAMIC MONTE CARLO LOCALIZATION UNTUK TRACKED MOBILE ROBOT

SKRIPSI

VEKTOR DEWANTO
0606029492

diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

FAKULTAS TEKNIK
PROGRAM SARJANA
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2010

HALAMAN PERNYATAAN ORISINALITAS

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

Nama : Vektor Dewanto
NPM : 0606029492
Tanda Tangan : .....
Tanggal : Juni 2010

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama : Vektor Dewanto
NPM : 0606029492
Program Studi : Teknik Elektro
Judul Skripsi : Dynamic Monte Carlo Localization untuk Tracked Mobile Robot

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro Fakultas Teknik Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Dr. Abdul Muis, S.T, M.Eng. (.....)

Penguji : Prof. Dr.Eng. Drs. Benyamin Kusumoputro M.Eng. (.....)

Penguji : Ir. Wahidin Wahab M.Sc., Ph.D. (.....)

Ditetapkan di : Depok

Tanggal : Juni 2010

KATA PENGANTAR

Alhamdulillah. Puji syukur penulis haturkan kehadirat Allah SWT atas limpahan karunia sehingga penyusunan skripsi ini dapat selesai dengan baik. Skripsi berjudul Dynamic Monte Carlo Localization untuk Tracked Mobile Robot diajukan untuk memenuhi salah satu persyaratan memperoleh gelar sarjana teknik. Selain itu, ini adalah bagian dari usaha penulis untuk dapat berkontribusi dalam bidang robotika.

Banyak pihak telah membantu. Penulis berterima kasih kepada Dr. Abdul Muis, S.T, M.Eng dan Dr. Wisnu Jatmiko atas bimbingan, akomodasi peralatan, dan kepercayaan untuk membangun robot Alfathvrss serta merawat hardware robot Alfathv1. Rasa terima kasih yang sebesar-besarnya juga penulis sampaikan kepada anggota Tim Robot Universitas Indonesia (TRUI) atas masukan teknis, akomodasi peralatan dan tempat, semangat kebersamaan, pengalaman akan budaya kerja keras, dan canda-tawa yang meredakan kepenatan. Perlu dikabarkan bahwa kemampuan robotika penulis dilahirkan oleh TRUI. Oleh karena itu, untuk anggota TRUI, penulis menyatakan diri sebagai saudara yang begitu dekat. Penulis menyampaikan apresiasi atas kesetiaan dalam perjuangan bersama untuk KRCI Expert Single 2008 dan 2009, dengan robot PreatenC128 dan Alfathv1, bagi anggota Tim Al Fath: Alvis, Syukron, Ifa, Rizky, Antoni, Fitra, Erik, dan Bogie. Ucapan terima kasih yang spesial untuk Rizky Prasetya atas bantuan yang begitu berharga semenjak membangun robot PreatenC128 sampai sekarang. Begitu juga untuk Erik Dominikus atas pondasi dalam software roboLab dan bimbingan dalam pemrograman JAVA, Teguh dan Hendra atas pinjaman peralatan mekanik, serta Zaky atas jasa penyolderan IC SMD. Selanjutnya, penulis mengucapkan terima kasih kepada teman-teman Departemen Teknik Elektro atas pertemanan yang sungguh berguna selama ini. Penulis selalu berharap untuk dapat membalas kebaikan kalian dan berdoa kepada Allah agar memberi pahala yang sedemikian pantas. Bagi penulis, ibu, bapak, adik-adik, dan keluarga yang lain adalah bagian tak terpisahkan dari diri penulis. Oleh karena itu, pada hakikatnya, karya ini dipersembahkan untuk mereka.

Akhirnya, penulis mengharapkan saran terhadap skripsi ini sehingga pada kesempatan mendatang akan lebih baik. Demikian, semoga karya ilmiah ini bermanfaat.

Depok, Juni 2010
Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Vektor Dewanto
NPM : 0606029492
Program Studi : Teknik Elektro
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

DYNAMIC MONTE CARLO LOCALIZATION UNTUK TRACKED MOBILE ROBOT

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan 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 : Juni 2010
Yang menyatakan

(Vektor Dewanto)

ABSTRAK

Nama : Vektor Dewanto
Program Studi : Teknik Elektro
Judul : Dynamic Monte Carlo Localization untuk Tracked Mobile Robot

Monte Carlo Localization (MCL) dikenal handal sebagai algoritma self-localization. Akan tetapi, implementasinya pada tracked mobile robot (TMR) masih jarang. Bisa jadi hal itu karena odometry TMR yang selalu berkonotasi buruk. Selain itu, kinerja MCL pada robot dengan jumlah sensor exteroceptive yang minim masih belum dibuktikan. Lagipula, isu sensitif pada MCL yaitu kebutuhan akan sumber daya komputasi masih menantang untuk dicari solusinya yang mudah diaplikasikan tetapi optimal. Pada riset ini, mula-mula dirumuskan model gerakan dan persepsi probabilistik TMR yang berperan dalam tahap sampling dan weighting pada algoritma MCL. Lebih lanjut, model gerakan probabilistik yang dipakai berjenis Odometry Motion Model dengan pendekatan distribusi Normal (Gausssian), dimana odometry berfungsi sebagai pendeskripsi informasi kontrol. Sementara itu, model persepsi probabilistik dirumuskan dengan asumsi bahwa semua exteroceptive sensor bersifat independent satu sama lain. Lebih spesifik, untuk sonar, ada tiga tipe error yang diperhitungkan secara eksplisit dalam model probabilistiknya, yaitu local noise, failures, dan random measurements. Akhirnya, berhasil dikembangkan suatu varian baru dari MCL yang dinamakan Dynamic-MCL. Karakteristik khasnya adalah adanya variasi jumlah particle yang dilibatkan berdasarkan Spread-factor (S), yaitu parameter yang mengindikasikan persebaran particles. Lebih lanjut, integrasinya dengan algoritma Plain-MCL dilakukan pada tahap resampling. Dengan eksperimen yang ekstensif, dibuktikan bahwa Dynamic-MCL dapat memecahkan tantangan lokalisasi pada TMR, mulai dari local-localization (pose tracking), global-localization, sampai kidnapped-robot.

Kata kunci:
Monte Carlo Localization (MCL), Tracked Mobile Robot (TMR), Plain-MCL, Dynamic-MCL, Spread-factor, Odometry Motion Model, model gerakan probabilistik, model persepsi probabilistik, local-localization (pose tracking), global-localization, kidnapped-robot

ABSTRACT

Name : Vektor Dewanto
Study Program : Electrical Engineering
Title : Dynamic Monte Carlo Localization for Tracked Mobile Robots

Monte Carlo Localization (MCL) is known as a robust self-localization algorithm. Nonetheless, its implementation on tracked mobile robots (TMRs) is rare. It is likely because odometry of a TMR always seems unworthy (erroneous). Besides, performance of MCL on robots lack of exteroceptive sensors has not been well proven yet. Moreover, a sensitive issue on MCL i.e. the need of computational resources still warrants further investigations looking for handy-yet-optimal solutions. In this research, a probabilistic motion model of a TMR is developed, as well as its probabilistic perception model. The former has a vital role in the sampling step, while the latter in the weighting step of MCL algorithm. Furthermore, the type of the probabilistic motion model used is Odometry Motion Model fitted by Normal (Gaussian) distribution, at which odometry is employed as a descriptor for control information. Meanwhile, the probabilistic perception model is formulated based on an assumption that is all sensors are mutually independent. Specifically, for sonars, there are three kinds of error that are explicitly reckoned in its probabilistic model, namely local noise, failures and random measurements. Finally, a new variant of MCL is introduced named Dynamic-MCL. Its unique characteristic is there is a variation on the number of particles involved based on Spread-factor (S) i.e. a parameter indicating the spread of particles. Furthermore, its integration to the Plain-MCL algorithm is carried out in the resampling step. Based on extensive experiments, it is explicable to claim that Dynamic-MCL is capable to solve localization challenges on TMRs including local-localization (pose tracking), global-localization and kidnapped-robot.

Key words:
Monte Carlo Localization (MCL), Tracked Mobile Robot (TMR), Plain-MCL, Dynamic-MCL, Spread-factor, Odometry Motion Model, probabilistic motion model, probabilistic perception model, local-localization (pose tracking), global-localization, kidnapped-robot

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PERNYATAAN ORISINALITAS ii
LEMBAR PENGESAHAN iii
KATA PENGANTAR iv
LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH v
ABSTRAK vi
DAFTAR ISI viii
DAFTAR GAMBAR xi

1. PENDAHULUAN
1.1 Latar Belakang 1
1.2 Perumusan dan Pembatasan Masalah 2
1.3 Tujuan dan Kontribusi Riset 3
1.4 Sistematika Penulisan 3

2. PERANGKAT KERAS TMR ALFATHVRSS DAN LINGKUNGAN KERJANYA
2.1 Pendahuluan 6
2.2 Sensors
2.2.1 Sonar SRF08 8
2.2.2 Kompas-digital CMPS03 8
2.2.3 Wheel-Encoders TraxsterII 11
2.3 Actuators
2.3.1 Brush DC Motor TraxsterII 11
2.3.2 LCD2x16 11
2.4 Microcontrollers 12
2.5 Media Komunikasi 13
2.6 Platform 13
2.7 Sistem Daya 13
2.8 Lingkungan Kerja Robot 14

3. PENJEJAK POSE-ABSOLUT
3.1 Pendahuluan 15
3.2 Prinsip Kerja 15
3.3 Implementasi 17
3.4 Pengujian Sistem
3.4.1 Prosedur Pengujian 21
3.4.2 Hasil Pengujian 22
3.5 Pengembangan Lebih Lanjut 23

4. ODOMETRY PADA TMR ALFATHVRSS
4.1 Pendahuluan 24
4.2 Model Kinematika 25
4.3 Kalibrasi Odometry 30
4.4 Pengujian Kinerja Odometry 35

5. MODEL GERAKAN PROBABILISTIK TMR ALFATHVRSS
5.1 Pendahuluan 37
5.2 Odometry Motion Model (OMM) 38
5.3 Eksperimen odometry-error 41
5.4 Probability Distribution untuk Model Gerakan
5.4.1 Model Gerakan dengan Parametric Probability Distribution . 43
5.4.2 Model Gerakan dengan Non-parametric Probability Distribution .. 50
5.5 Eksperimen Sampling
5.5.1 Algoritma Sampling 52
5.5.2 Hasil Eksperimen Sampling 55
5.5.3 Analisis Hasil Eksperimen Sampling 57

6. MODEL PERSEPSI PROBABILISTIK TMR ALFATHVRSS
6.1 Pendahuluan 62
6.2 Model Probabilistik Sonar SRF08
6.2.1 Masalah Dasar Sonar 63
6.2.2 Metode Ray-Casting 65
6.2.3 Hasil Eksperimen Pembacaan Sonar 66
6.2.4 Tipe-tipe Error Sonar-Reading dan Pemodelannya
6.2.4.1 Local Noise 67
6.2.4.2 Failures 70
6.2.4.3 Random Measurement 71
6.2.5 Pemodelan Probabilistik Sonar 73
6.3 Model Probabilistik Kompas-Digital CMPS03
6.3.1 Hasil Eksperimen Pembacaan Kompas-Digital 74
6.3.2 Pemodelan Probabilistik Kompas-Digital 76
6.4 Peranan Model Persepsi Probabilistik pada MCL 77

7. MCL PADA TMR ALFATHVRSS
7.1 Pendahuluan 79
7.2 Standard Kualitas Algoritma Lokalisasi 80
7.3 Plain-MCL
7.3.1 Algoritma Plain-MCL 82
7.3.2 Eksperimen Local-Localization dengan Plain-MCL
7.3.2.1 Tujuan Ekperimen 86
7.3.2.2 Detil Eksperimen 86
7.3.2.3 Hasil Eksperimen 87
7.3.2.4 Analisis Hasil Eksperimen 90
7.4 Dynamic-MCL
7.4.1 Ide Dasar 91
7.4.2 Proses Implementasi 93
7.4.3 Algoritma 98
7.5 Analisis Perbandingan Plain-MCL dengan Dynamic-MCL 99

8. DYNAMIC-MCL UNTUK LOKALISASI GLOBAL DAN KIDNAPPED-ROBOT
8.1 Pendahuluan 101
8.2 Global-Localization 102
8.3 Kidnapped-Robot
8.3.1 Ide Dasar 107
8.3.2 Hasil Eksperimen dan Analisis 108

9. PENUTUP
9.1 Kesimpulan 115
9.2 Diskusi tentang Implementasi Dynamic-MCL untuk Robot KRCI 116
9.3 Riset Lebih Lanjut 118

DAFTAR REFERENSI 119

DAFTAR GAMBAR

Gambar 1.1 Outline Skripsi 5

Gambar 2.1 TMR Alfathvrss 6
Gambar 2.2 Definisi Titik Referensi Pose R dan Koordinat Lokal Robot 7
Gambar 2.3 Spesifikasi Sonar SRF08 9
Gambar 2.4 Pembacaan Kompas-digital 10
Gambar 2.5 Hasil Kalibrasi Kompas-digital 10
Gambar 2.6 Spesifikasi Motor TraxsterII 11
Gambar 2.7 Diagram Koneksi Microcontrollers 12
Gambar 2.8 Spesifikasi Piranti Komunikasi YS1020U RF 13
Gambar 2.9 Lingkungan Kerja Robot 14

Gambar 3.1 Citra Robot dan Lingkungannya 16
Gambar 3.2 Ilustrasi penentuan faktor konversi pixel-cm dan offset dari citra 18
Gambar 3.3 Flowchart Ekstraksi Citra Robot pada Penjejak Pose-Absolut 19
Gambar 3.4 Citra RGB dan BW-nya 20
Gambar 3.5 Hasil Akhir dari Penjejak Pose-Absolut 20
Gambar 3.6 Hasil Proses Filtering dan Clutering terhadap white-pixel 21
Gambar 3.7 Hasil Pengujian Kinerja Penjejak Pose Absolut 22
Gambar 3.8 Error Posisi Hasil Penjejak Pose Absolut 22

Gambar 4.1 Ekivalensi antara TMR dan DMR untuk Pergerakan Sesaat 25
Gambar 4.2 Kerangka Referensi Global dan Lokal Robot 26
Gambar 4.3 Lintasan roda-roda pada DMR 27
Gambar 4.4 Pendefinisian Dn pada TMR Alfathvrss 29
Gambar 4.5 Perbandingan Systematic-Error Odometri 35
Gambar 4.6 Perbandingan Pose-Odometry dengan Pose-Absolut: arah-gerak CCW 36
Gambar 4.7 Perbandingan Pose-Odometry dengan Pose-Absolut: arah-gerak CW .. 36

Gambar 5.1 Dekomposisi Gerakan Robot 39
Gambar 5.2 Grafik $error_{rot1}$ 42
Gambar 5.3 Grafik $error_{trans}$ 42
Gambar 5.4 Grafik $error_{rot2}$ 43
Gambar 5.5 Hasil Normal-Fitting untuk Data $error_{rot1}$ 44
Gambar 5.6 Detil Proses Normal-Fitting pada Data $error_{rot1}$ 45
Gambar 5.7 Hasil Normal-Fitting untuk Data $error_{trans}$ 45
Gambar 5.8 Detil Proses Normal-Fitting pada Data $error_{trans}$ 46
Gambar 5.9 Hasil Normal-Fitting untuk Data $error_{rot2}$ 46
Gambar 5.10 Detil Proses Normal-Fitting pada Data $error_{rot2}$ 47
Gambar 5.11 Normal Probability Plot untuk $error_{rot1}$ 48
Gambar 5.12 Normal Probability Plot untuk $error_{trans}$ 48
Gambar 5.13 Normal Probability Plot untuk $error_{rot2}$ 49
Gambar 5.14 Normal Probability Plot untuk Data Normal 49
Gambar 5.15 Hasil Non-parametric-Fitting untuk Data $error_{rot1}$ 51
Gambar 5.16 Hasil Non-parametric-Fitting untuk Data $error_{trans}$ 51

Gambar 5.17 Hasil Non-parametric-Fitting untuk Data $error_{rot2}$ 52
Gambar 5.18 Algoritma sampling pada MCL 53
Gambar 5.19 Ilustrasi Acceptance-rejection Method 54
Gambar 5.20 Hasil Sampling untuk Jenis Gerakan Forward 56
Gambar 5.21 Hasil Sampling untuk Jenis Gerakan Backward 57
Gambar 5.22 Hasil Sampling untuk Jenis Gerakan Rot-CW 58
Gambar 5.23 Hasil Sampling untuk Jenis Gerakan Rot-CCW 59
Gambar 5.24 Grafik Perbandingan Error X pada Eksperimen Sampling 60
Gambar 5.25 Grafik Perbandingan Error Y pada Eksperimen Sampling 60
Gambar 5.26 Grafik Perbandingan Error Theta pada Eksperimen Sampling ... 61
Gambar 5.27 Grafik Perbandingan Waktu Komputasi pada Eksperimen Sampling . 61

Gambar 6.1 Masalah Dasar Sonar 63
Gambar 6.2 Beam Pattern Sonar SRF08 64
Gambar 6.3 Visualisasi Ray-Casting pada TMR Alfathvrss 65
Gambar 6.4 Hasil Eksperimen Pembacaan Sonar 66
Gambar 6.5 Catatan Normal Fitting pada Local Noise dari Sonar 68
Gambar 6.6 Visualisasi Model Persepsi Probabilistik Sonar SRF08: Local Noise ... 70
Gambar 6.7 Visualisasi Model Persepsi Probabilistik Sonar SRF08: Failure Error .. 72
Gambar 6.8 Visualisasi Model Persepsi Probabilistik Sonar SRF08: Random Error 72
Gambar 6.9 Model Persepsi Probabilistik Sonar SRF08 73
Gambar 6.9 Error Pembacaan Kompas-digital 74
Gambar 6.10 Normal Fitting pada Error Pembacaan Kompas-Digital 75
Gambar 6.11 Catatan Normal Fitting pada Error Pembacaan Kompas-Digital 75
Gambar 6.12 Model Persepsi Probabilistik Kompas-Digital CMPS03 76
Gambar 6.13 Algoritma Weighting pada MCL 78

Gambar 7.1 Hierarki Algoritma Lokalisasi Bayer Filter 79
Gambar 7.2 Algoritma Bayes-Filter 83
Gambar 7.3 Algoritma Particle-Filter 83
Gambar 7.4 Algoritma Plain-MCL 83
Gambar 7.5 Algoritma Residual Systematic Resampling (RSR) 85
Gambar 7.6 Contoh Penerapan Algoritma RSR 85
Gambar 7.7 Beberapa titik-referensi-pose pada Lingkungan-A 87
Gambar 7.8 Grafik Hubungan Waktu Komputasi dengan Jumlah Particle 87
Gambar 7.9 Grafik Hubungan Err2 dengan Jumlah Particle 88
Gambar 7.10 Grafik Hubungan $Err1_{Theta}$ dengan Jumlah Particle 88
Gambar 7.11 Grafik Hubungan $Err1_{Y}$ dengan Jumlah Particle 89
Gambar 7.12 Grafik Hubungan $Err1_{X}$ dengan Jumlah Particle 89
Gambar 7.13 Ilustrasi Persebaran Particles 92
Gambar 7.14 Variasi S dalam Plain-MCL 94
Gambar 7.15 Hubungan $Err1_{X}$ dengan Spread-Factor pada Plain-MCL 95
Gambar 7.16 Hubungan $Err1_{Y}$ dengan Spread-Factor pada Plain-MCL 95
Gambar 7.17 Hubungan $Err1_{Theta}$ dengan Spread-Factor pada Plain-MCL 96
Gambar 7.18 Hubungan Err2 dengan Spread-Factor pada Plain-MCL 96
Gambar 7.19 Hubungan Spread-Factor dengan Jumlah Particle pada Dynamic-MCL 98
Gambar 7.20 Beberapa titik-referensi-pose pada Lingkungan-B 99

Gambar 7.21 Perbandingan Plain-MCL dengan Dynamic-MCL pada Lingkungan-A 100
Gambar 7.22 Perbandingan Plain-MCL dengan Dynamic-MCL pada Lingkungan-B 100

Gambar 8.1 Perbandingan Dynamic-MCL dengan variasi n_{init} 103
Gambar 8.2 Perbandingan Dynamic-MCL dengan variasi n_{init} pada parameter err_2 .. 104
Gambar 8.3 Perbandingan Dynamic-MCL dengan variasi n_{init} pada parameter $err1_X$ 104
Gambar 8.4 Perbandingan Dynamic-MCL dengan variasi n_{init} pada parameter $err1_Y$ 105
Gambar 8.5 Variasi n pada Dynamic-MCL untuk global-localization 105
Gambar 8.6 Visualisasi kerja Dynamic-MCL untuk global-localization 106
Gambar 8.4 Visualisasi Kerja Dynamic-MCL pada Kidnapped-robot t = 0 .. 109
Gambar 8.5 Visualisasi Kerja Dynamic-MCL pada Kidnapped-robot t = 2 . 110
Gambar 8.6 Visualisasi Kerja Dynamic-MCL pada Kidnapped-robot t = 8 .. 110
Gambar 8.7 Visualisasi Kerja Dynamic-MCL pada Kidnapped-robot t = 10 111
Gambar 8.8 Visualisasi Kerja Dynamic-MCL pada Kidnapped-robot t = 11 111
Gambar 8.9 Visualisasi Kerja Dynamic-MCL pada Kidnapped-robot t = 12 112
Gambar 8.10 Visualisasi Kerja Dynamic-MCL pada Kidnapped-robot t = 17 112
Gambar 8.11 Visualisasi Kerja Dynamic-MCL pada Kidnapped-robot t = 22 113
Gambar 8.12 Variasi nilai w_{max} (unnomarlized) 113
Gambar 8.13 Visualisasi tahap Weighting Dynamic-MCL pada lingkungan-C 114

BAB 1

PENDAHULUAN

1.1 Latar Belakang

"Where am I?" adalah pertanyaan paling mendasar bagi autonomous mobile robot (AMR) dan merupakan representasi lugas dari masalah lokalisasi. Klaim otomatis musti didasarkan pada kemampuan robot untuk tahu tentang pose-nya (yang mencakup posisi x, y, dan orientasi θ) relatif terhadap peta lingkungan yang diberikan dalam berbagai kemungkinan kondisi yaitu pose awal diketahui, pose awal tidak diketahui, dan kidnapped-robot (yaitu robot yang sedang yakin terhadap pose-nya diteleportasi ke pose lain tanpa sepengetahuan robot). Kondisi-kondisi tersebut, secara berurutan, biasa diasosiasikan dengan local localization atau pose tracking, global localization, dan kidnapped-robot localization. Solusi terhadap lokalisasi yang terbaik adalah AMR dapat menyimpulkan sendiri pose-nya secepat mungkin dengan mengandalkan informasi dari sensor-sensornya dan dengan tidak bergantung pada atau membutuhkan peralatan luar, seperti global positioning system (GPS) atau landmark khusus. Dengan demikian, dibutuhkan suatu algoritma atau kecerdasan buatan yang mengolah informasi sensor sehingga diperoleh pose secara mandiri; melakukan self-localization.

Lebih lanjut, Monte Carlo Localization (MCL) dikenal handal sebagai algoritma self-localization [3], [25], [26], [27], [28], [35]. Akan tetapi, implementasinya pada tracked mobile robot (TMR) masih jarang (jika terlalu berlebihan untuk dikatakan belum ada sama sekali); tidak sebanyak pada wheeled maupun legged mobile robot. Bisa jadi hal itu karena odometry TMR yang selalu berkonotasi buruk. Selain itu, kinerja MCL pada robot dengan jumlah sensor exteroceptive yang minim, di mana hanya ada empat sonar dan sebuah kompas-digital, masih belum dibuktikan. Dengan demikian, menerapkan MCL pada TMR dengan sumber daya sensor sebagaimana disebutkan tadi adalah suatu tantangan.

Isu lain pada self-localization adalah kebutuhan akan sumber daya komputasi yang meliputi kecepatan komputasi. Oleh karena itu, walaupun prinsip dasar pada MCL adalah semakin banyak particle yang dilibatkan, semakin bagus kualitas hasil lokalisasi, pada prakteknya jumlah particle tersebut dibatasi oleh sumber daya komputasi. Dengan demikian, terbesit keinginan untuk membuat jumlah particle tersebut dinamis bergantung pada kualitas hasil lokalisasi pada satu waktu sebelumnya.

1.2 Perumusan dan Pembatasan Masalah

Masalah pokok dalam riset ini adalah merumuskan suatu varian MCL yang mudah diaplikasikan dan optimal dalam memecahkan tantangan lokalisasi pada TMR mulai dari local localization, global localization, sampai kidnapped-robot localization. Lebih lanjut, dari pokok masalah itu, timbul beberapa sub-masalah yang menuntut untuk diselesaikan terlebih dahulu, yaitu pengembangan suatu sistem penjejak pose-absolut yang otomatis dan handal sebagai media validasi kinerja kecerdasan self-localization, optimasi dan pengujian kinerja odometry TMR, perumusan model gerakan dan persepsi probabilistik TMR, serta penerapan dan pengujian kinerja Plain-MCL pada TMR.

Lebih spesifik, objek riset ini adalah sebuah TMR Alfathvrss sehingga segala masalah dan solusinya akan diterapkan padanya. Selain itu, lingkungan robot bersifat indoor dan statis yang berupa area persegi panjang berukuran 160 cm x 120 cm, beralaskan karpet, dan dibatasi oleh empat dinding lurus dengan tinggi 30 cm berwarna hitam, terbuat dari plastik. Perlu disampaikan juga bahwa modul kecerdasan lokalisasi yang akan dikembangkan bersifat independent terhadap modul kecerdasan lain, misal path-planning. Dengan kata lain, robot dikontrol oleh suatu modul kecerdasan lain yang tidak ditujukan untuk membantu proses lokalisasi; algoritma atau modul lokalisasi dapat dikatakan hanya mengamati dan selanjutnya menyimpulkan pose robot, tanpa ada kemampuan untuk mengontrol robot.

Jadi tantangan lokalisasi yang ingin dijawab dalam riset ini dapat digolongkan ke dalam kasus atau tingkatan: single-robot, static-environment, dan passive-localization.

1.3 Tujuan dan Kontribusi Riset

Tujuan dan kontribusi utama riset ini adalah mengembangkan suatu varian MCL yang dapat memecahkan masalah lokalisasi pada TMR mulai dari local localization, global localization, sampai kidnapped-robot localization. Dalam rangka mewujudkan tujuan itu dilakukan beberapa langkah kerja sebagai berikut:

- Pengembangan suatu sistem penjejak pose-absolut yang otomatis dan handal sebagai media uji atau validasi kecerdasan lokalisasi robot.
- Kalibrasi odometry dan kalkulasi error odometry-terkalibrasi pada TMR.
- Perumusan model gerakan dan persepsi probabilistik TMR.
- Pengujian dan analisis kinerja Plain-MCL pada TMR.

1.4 Sistematika Penulisan

Outline skripsi ini ditunjukkan pada gambar 1.1, tampak bahwa blok tentang penjejak pose-absolut terhubung dengan blok-blok lain melalui garis putus-putus. Hal tersebut menegaskan bahwa output sistem itu, yaitu pose absolut, berperan sebagai pendukung dalam perumusan model gerakan dan persepsi probabilistik serta sebagai media validasi bagi kinerja MCL. Lebih lanjut, gambaran tentang isi bab 2, 3, dan seterusnya adalah sebagai berikut:

Bab 2 Perangkat Keras TMR Alfathvrss dan Lingkungan Kerjanya: berisi detil piranti yang ada di TMR Alfatvrss, yaitu spesifikasi, konfigurasi, dan kalibrasi dari sensors, actuators, microcontrollers, media komunikasi, platform, dan sistem daya. Selain itu, juga dibahas tentang lingkungan kerja robot yang digunakan dalam setiap eksperimen.

Bab 3 Penjejak Pose-Absolut: berisi ide dasar, algoritma, dan implementasi dari sistem penjejak pose-absolut yang dibangun. Terdapat pula, hasil pengujian kinerjanya.

Bab 4 Odometry pada TMR Alfathvrss: berisi model kinematika TMR sebagai rumusan inti untuk melakukan odometry. Selain itu, dipaparkan pula proses dan hasil kalibrasi odometry dengan metode UMBmark. Selanjutnya, pengujian kinerja odometry-terkalibrasi menjadi penutup bab ini.

Bab 5 Model Gerakan Probabilistik TMR Alfathvrss: berisi perumusan odometry motion model (OMM), kemudian terdapat pembahasan mengenai model parametrik dan non-parametrik untuk gerakan TMR. Selain itu, algoritma sampling pada MCL dan visualisasi kerjanya juga dipaparkan di sini.

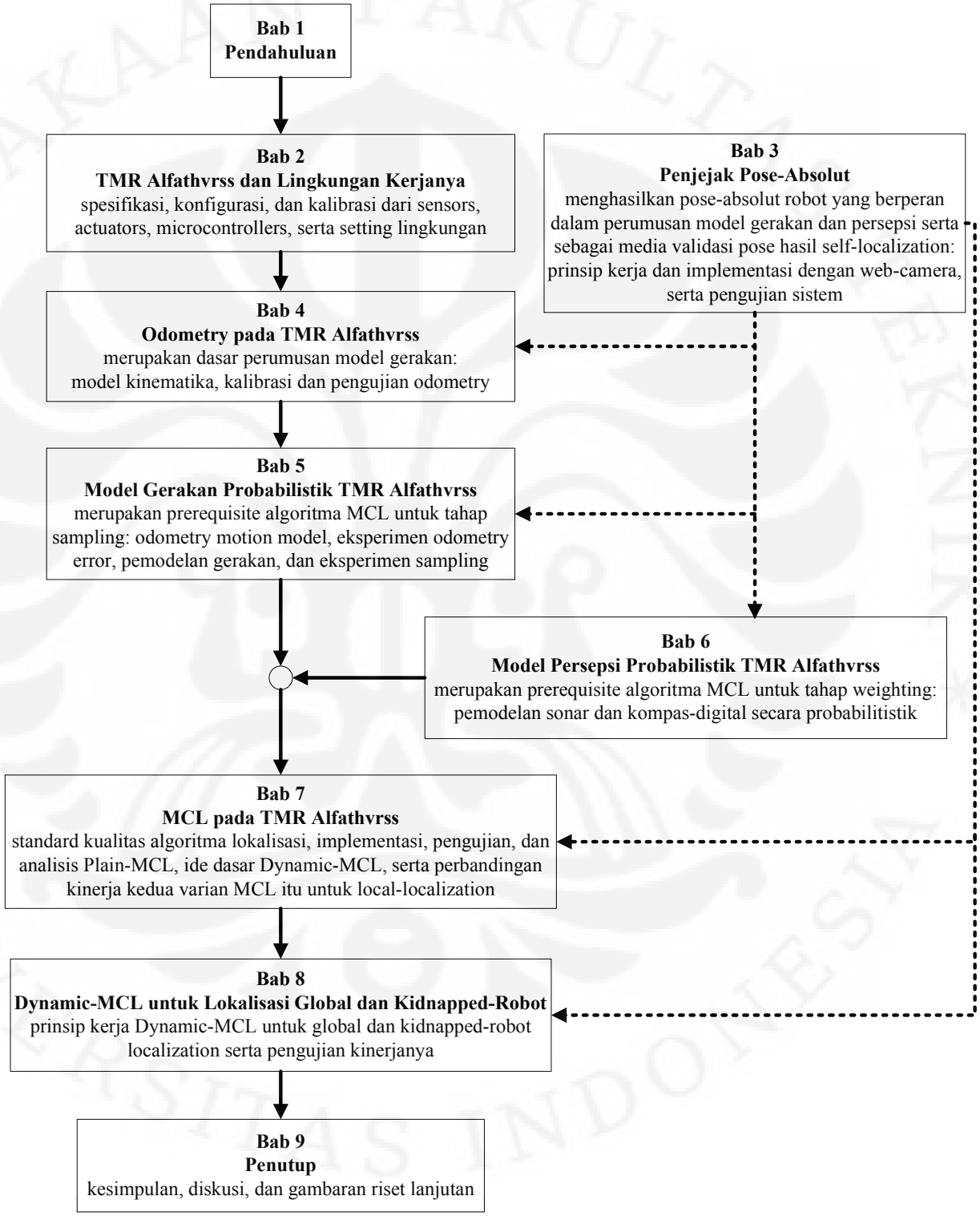
Bab 6 Model Persepsi Probabilistik TMR Alfathvrss: berisi model probabilistik untuk sonar SRF08 dan kompas-digital CMPS03. Jenis-jenis error pada pembacaan tiap sensor diidentifikasi dan selanjutnya digunakan untuk membentuk modelnya secara probabilistik. Dibahas pula peranan model persepsi dalam algoritma MCL yaitu pada tahap weighting.

Bab 7 MCL pada TMR Alfathvrss: mula-mula diuraikan tentang standard kualitas algoritma lokalisasi yang ingin dicapai. Kemudian dilanjutkan dengan penjelasan tentang algoritma Plain-MCL secara praktis dan lugas, termasuk metode resampling. Hasil eksperimen untuk menguji Plain-MCL juga dipaparkan mendetil sekaligus analisisnya. Pada bab ini dimunculkan gagasan untuk varian baru dari Plain-MCL, yaitu Dynamic-MCL, yang meliputi implementasi dan pengujian kinerjanya. Akhirnya, uji perbandingan kinerja keduanya menjadi penutup bab ini.

Bab 8 Dynamic-MCL untuk Lokalisasi Global dan Kidnapped-Robot: berisi analisis kinerja Dynamic-MCL dalam global-localization. Selain itu, juga dibahas solusi untuk kidnapped-robot dan penerapannya dengan Dynamic-MCL.

Bab 9 Penutup: berisi kesimpulan yang komprehensif mengenai temuan-temuan selama riset. Ada juga diskusi tentang penerapan algoritma self-localization

Dynamic-MCL untuk robot yang berlaga di Kontes Robot Cerdas Indonesia (KRCI). Bab ini ditutup dengan bahasan mengenai riset lebih lanjut dan peluang-peluang riset lain untuk terus dapat berkontribusi dalam bidang robotika.



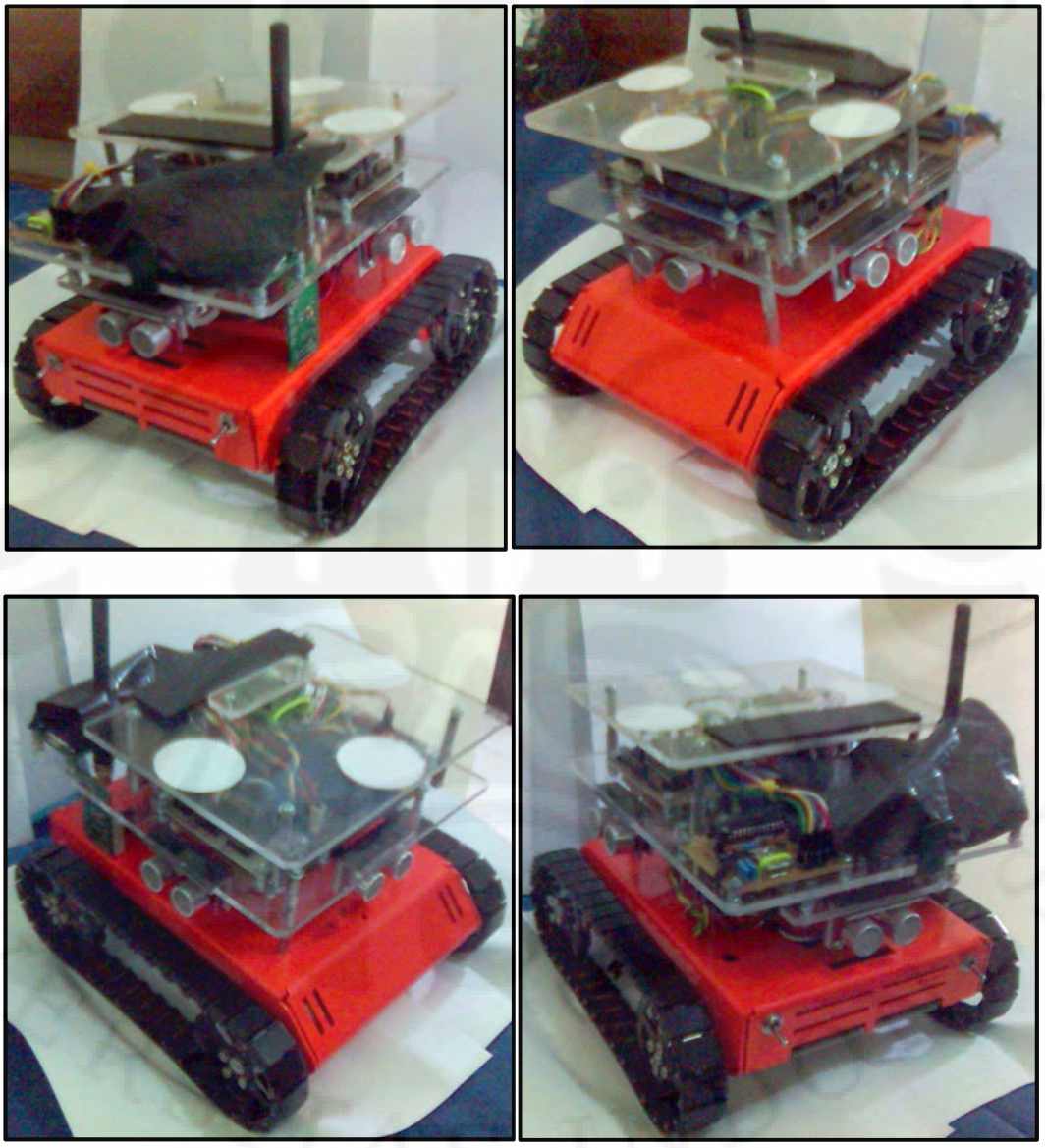
Gambar 1.1 Outline Skripsi

BAB 2

PERANGKAT KERAS TMR ALFATHVRSS DAN LINGKUNGAN KERJANYA

2.1 Pendahuluan

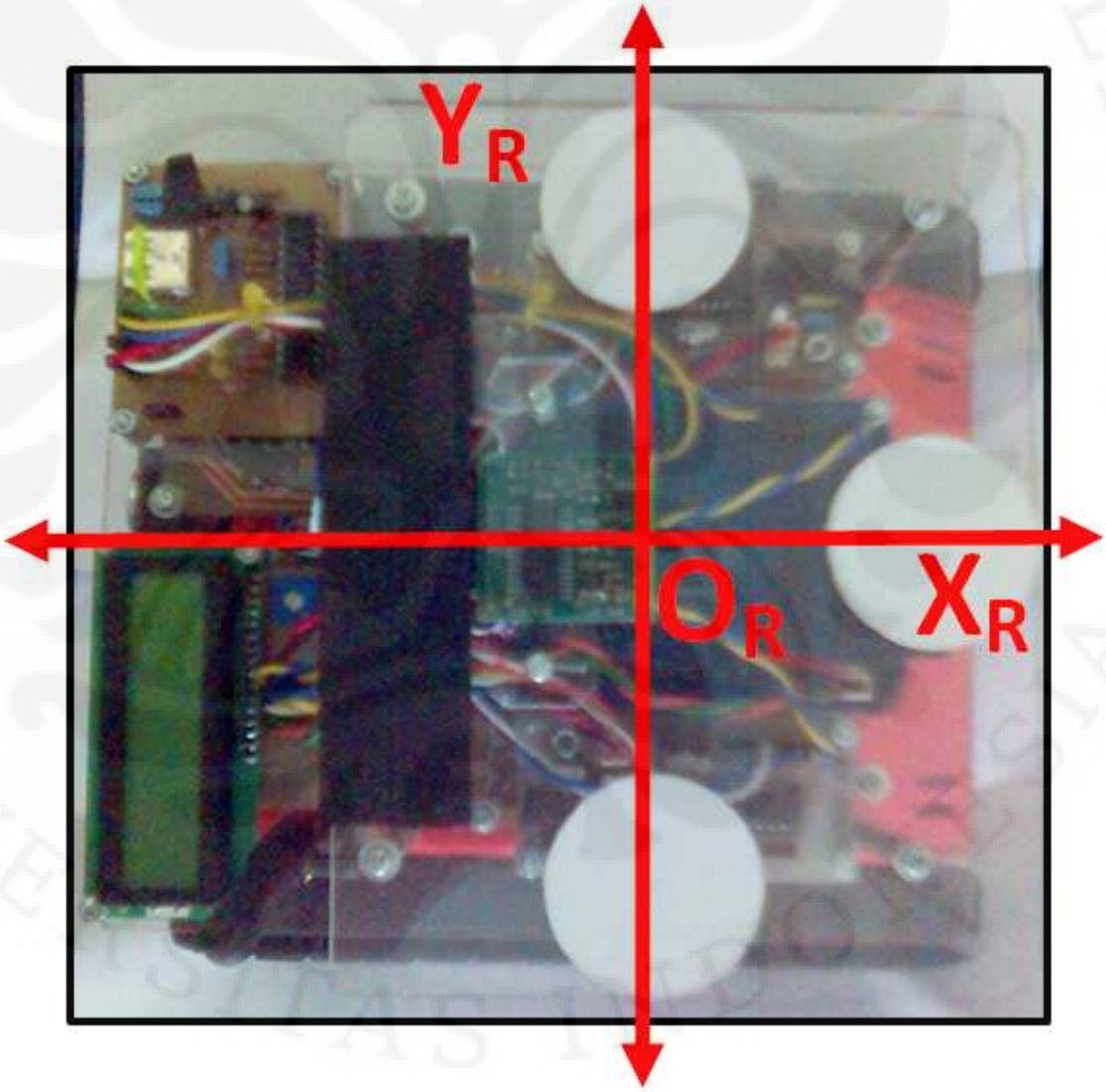
TMR Alfathvrss adalah robot yang dibangun untuk tujuan riset dalam ruang (indoor). Penampakannya dari berbagai sudut ditunjukkan pada gambar (2.1).



Gambar 2.1 TMR Alfathvrss

Lebih lanjut, perlu didefinisikan satu titik pada badan robot sebagai titik acuan posenya, dilambangkan dengan R. Titik tersebut juga merupakan titik Origin O_R dari koordinat lokal robot. Letak R (= O_R) ditunjukkan pada gambar (2.2).

Bab ini membahas perangkat keras TMR Alfathvrss yang meliputi sensors (subbab 2.2), actuators (subbab 2.3), microcontrollers (subbab 2.4), media komunikasi (subbab 2.5), platform (subbab 2.6), dan sistem daya (subbab 2.7). Selain itu, terdapat deskripsi mengenai lingkungan kerja robot yang menjadi lingkungan eksperimen dalam riset ini.



Gambar 2.2 Definisi Titik Referensi Pose R dan Koordinat Lokal Robot

2.2 Sensors

2.2.1 Sonar SRF08

Spesifikasi tiap sonar SRF08 ditunjukkan pada gambar (2.3). Untuk pemakaian pada riset ini, jarak maksimum diset pada nilai 200 cm. Lebih detil, TMR Alfathvrss mempunyai empat buah sonar SRF08 yang dipasang pada empat arah utama robot, yaitu arah depan (X+), arah kanan (Y-), arah belakang (X-), dan arah kiri (Y+). Koordinat tiap sonar dalam kerangka lokal robot ditunjukkan pada persamaan (2.1).

$$sonar_{front} = \begin{bmatrix} 7.5 \\ 0 \end{bmatrix} \quad (2.1a)$$

$$sonar_{right} = \begin{bmatrix} 0 \\ -8.5 \end{bmatrix} \quad (2.1b)$$

$$sonar_{rear} = \begin{bmatrix} -12 \\ 0 \end{bmatrix} \quad (2.1c)$$

$$sonar_{left} = \begin{bmatrix} 0 \\ 8.5 \end{bmatrix} \quad (2.1d)$$

Perlu disampaikan bahwa pada algoritma kecerdasan robot, sonar-sonar tersebut dianggap menempel di titik R. Oleh karena itu, pembacaan mentah dari tiap sensor perlu disesuaikan dengan persamaan (2.1) untuk mengakomodasi anggapan tersebut.

2.2.2 Kompas-digital CMPS03

TMR Alfathvrss mempunyai sebuah kompas digital dengan resolusi 1˚. Pemasangannya ditunjukkan pada gambar (2.2) yaitu di sekitar titik O_R, tepatnya di koordinat (-1, 0). Perlu disampaikan bahwa resolusi tersebut tidak sepenuhnya stabil karena sensor ini cukup sensitif terhadap medan magnet luar. Oleh karena itu, pada tiap pemakaiannya di suatu lingkungan dibutuhkan kalibrasi terhadap keluarannya.

$$cmps_{cal} = -0.0031 cmps_{raw}{}^2 + 2.4629\, cmps_{raw} - 226.5120 \quad (2.2)$$

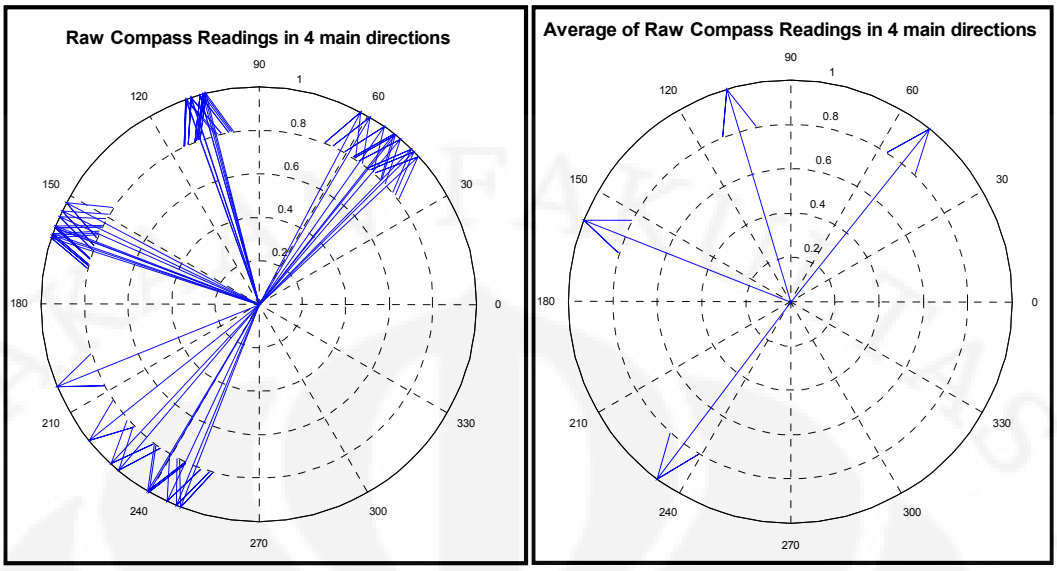
Specifications

Beam Pattern	see graph
Voltage	5v
Current	15mA Typ. 3mA Standby
Frequency	40KHz
Maximum Range	6 m
Minimum Range	3 cm
Max Analogue Gain	Variable to 1025 in 32 steps
Connection	Standard IIC Bus
Light Sensor	Front facing light sensor
Timing	Fully timed echo, freeing host computer of task
Echo	Multiple echo - keeps looking after first echo
Units	Range reported n uS, mm or inches
Weight	0.4 oz.
Size	43mm w x 20mm d x 17mm h

Gambar 2.3 Spesifikasi Sonar SRF08 [66]

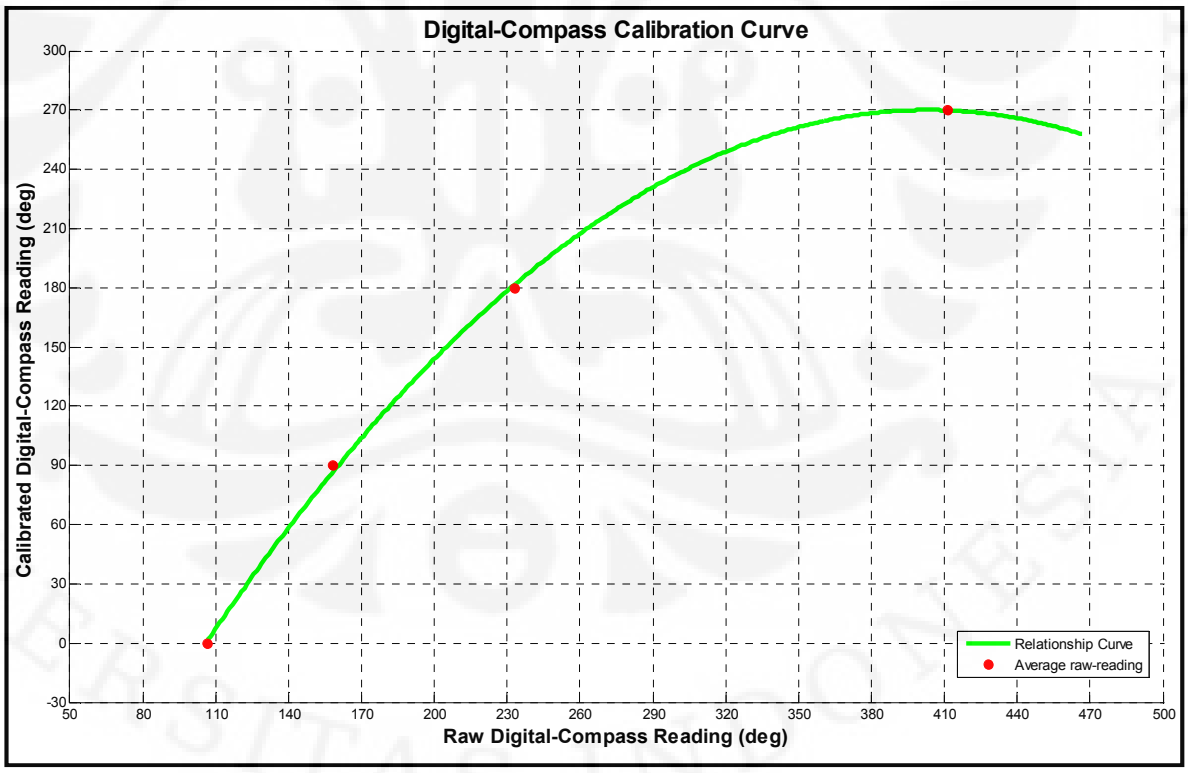
Prosedur kalibrasi kompas-digital yang dilakukan dalam riset ini adalah sebagai berikut.

- Mengambil data pembacaan kompas-digital pada empat arah utama world-frame pada 9 titik yang berbeda di lingkungan sehingga total ada 36 pembacaan kompas-digital. Arah utama (X+, X-, Y+, dan Y-) dipilih karena arah absolutnya bernilai pasti, secara berurutan, yaitu 0˚, 90˚, 180˚, dan 270˚. Penentuan kesembilan titik dibuat sedemikian rupa sehingga diperoleh jarak yang sejauh mungkin antara satu titik dengan yang lain, dengan tujuan untuk mendapatkan gambaran variasi pembacaan kompas-digital yang representatif pada lingkungan. Data tersebut ditunjukkan pada gambar (2.4a).
- Mencari nilai rata-rata pembacaan mentah kompas-digital pada tiap arah utama sehingga total ada empat nilai rata-rata, gambar (2.4b).
- Mencari persamaan yang menghubungkan hasil pembacaan mentah rata-rata dengan arah yang seharusnya (arah-absolut). Plotnya ditunjukkan pada gambar (2.5), sedangkan persamaan kurva dinyatakan di persamaan (2.2).



(a) (b)

Gambar 2.4 Pembacaan Kompas-digital, (a) data mentah (b) rata-rata data mentah



Gambar 2.5 Hasil Kalibrasi Kompas-digital

2.2.3 Wheel-Encoders TraxsterII

TMR Alfathvrss dilengkapi dengan dua Quadrature Wheel Encoders yang telah dibangun secara tetap (embedded) di motor-motor. Spesifikasi utamanya adalah 624 pulses per output shaft revolution.

2.3 Actuators

2.3.1 Brush DC Motor TraxsterII

TMR Alfathvrss mempunyai sepasang brush DC Motor TraxsterII yang merupakan actuator utama. Spesifikasi detil motor ditunjukkan pada gambar (2.6). Lebih detil, motor digerakkan melalui rangkaian H-bridge berbasis IRF540 dan IRF9540.

Specifications:

- Voltage: 7.2 VDC
- No Load Current: 300mA
- Max Current Draw: 4.0A
- No Load Speed: 160 RPM
- Gear Reduction: 1:52
- Stall Torque: 100 oz-in (7.2 kg-cm)
- Weight: 4.7oz
- Motor Leads: 12" Long 18AWG

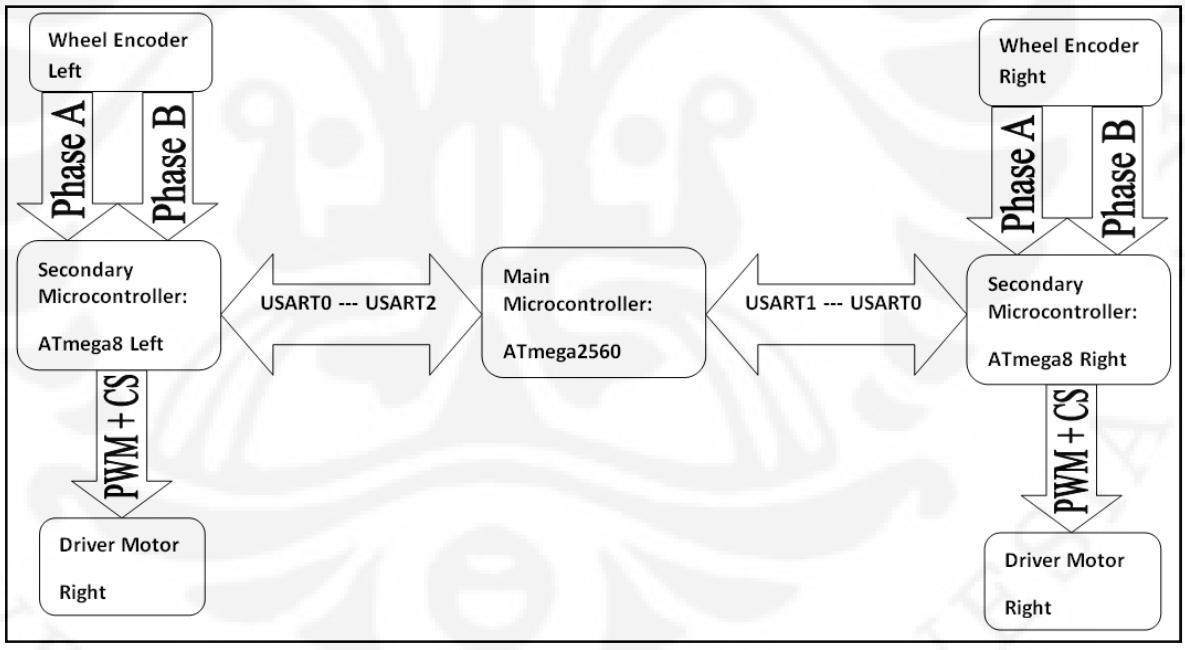
Gambar 2.6 Spesifikasi Motor TraxsterII [69]

2.3.2 LCD2x16

TMR Alfathvrss dilengkapi dengan sebuah LCD2x16 sebagai piranti debugging. Sinyal kendalinya dipegang oleh pengontrol utama.

2.4 Microcontrollers

Pada TMR Alfathvrss, terdapat tiga buah microcontrollers, di mana satu di antaranya berfungsi sebagai pengontrol utama, yaitu Atmega2560; sedangkan dua yang lain sebagai pengontrol sekunder, yaitu Atmega8. Kecepatan komputasi ketiganya dapat diasumsikan sama dengan kecepatan crystal-clock-nya yaitu 11.0592 MHz. Lebih lanjut, pengontrol sekunder digunakan untuk mengontrol motor secara mandiri, satu microcontroller untuk tiap motor. Selain itu, pembacaan wheel-encoder juga ditangani oleh pengontrol sekunder. Komunikasinya dengan pengontrol utama dijalankan melalui protokol USART. Diagram koneksi ketiganya ditunjukkan pada gambar (2.7). Sebagai tambahan, fungsi pengontrol sekunder dibatasi pada dua fungsi di atas, pekerjaan lain ditangani oleh pengontrol utama.



Gambar 2.7 Diagram Koneksi Microcontrollers

2.5 Media Komunikasi

Terdapat piranti komunikasi dengan sistem di luar robot yaitu wireless USART YS1020U RF Transceiver produksi Hong Kong HuaWei International Electronic Limited, spesifikasinya ditunjukkan pada gambar (2.8). Kegunaan utamanya adalah mengakomodasi aktivitas akuisisi data dan remote-controlling.

Power supply: DC 5v or 3.3V;
RF power: ≤10mw;
Receiving current: <25mA;
Transmitting current: <40mA;
Sleep current: <20uA;
Transmission distance can reach 500m (BER=10-3@9600bps);
Size: 47mm×26mm×10mm (without antenna port).

Gambar 2.8 Spesifikasi Piranti Komunikasi YS1020U RF [70]

2.6 Platform

TMR Alfathvrss dibangun di atas platform (locomotion) produksi Robotics-Connection yang bermerek TraxsterII. Spesifikasi dimensinya adalah sebagai berikut:

- Panjang (Length): 9 inches = 229 mm
- Lebar (Width): 8 inches = 203 mm
- Tinggi (Height): 3 inches = 76 mm
- Berat (Weight): 2.0 lbs =32 ounces = 907.1680 gram
- Ground Clearance: 0.75 inches = 19 mm

2.7 Sistem Daya

Ada dua level tegangan di TMR Alfathvrss yaitu 5VDC and 7.4VDC. Yang pertama digunakan untuk mencatu sensors, microcontrollers, LCD, wireless-USART, and piranti-piranti TTL yang lain, sedangkan yang kedua digunakan untuk menyuplai daya ke motor-motor.

2.8 Lingkungan Kerja Robot

Dalam riset ini lingkungan kerja robot bersifat statis (tidak berubah-ubah selama robot bekerja) dan berupa area persegi-panjang berukuran 160 cm x 120 cm, beralaskan karpet, serta dibatasi oleh empat dinding lurus dengan tinggi 30 cm terbuat dari plastik hitam, gambar (2.9).

Konfigurasi lingkungan tersebut memang bisa dikategorikan sederhana, tetapi di situ justru timbul tantangan bagi pembacaan sonar. Hal tersebut karena jarak robot dengan dinding-dinding, sebagian besar, relatif lebih jauh daripada jika konfigurasi berupa ruang-ruang (labirin) dengan banyak dinding. Perlu disampaikan juga bahwa berdasarkan pengalaman, ada kecenderungan pembacaan sonar memburuk dengan bertambahnya jarak yang dibacanya. Dengan demikian, dapat diklaim bahwa pembacaan sonar dalam riset ini relatif lebih noisy sehingga menuntut pemodelan yang handal.

Gambar 2.9 Lingkungan Kerja Robot

BAB 3

PENJEJAK POSE-ABSOLUT

3.1 Pendahuluan

Untuk validasi atau pengecekan pose robot hasil algoritma self-localization, perlu dibangun sistem lokalisasi yang independent dari robot dan menghasilkan pose yang selalu mendekati pose robot yang sesungguhnya (nyata atau aktual); pose yang demikian selanjutnya disebut sebagai pose-absolut. Dengan kata lain, sistem itu dapat menggantikan aktivitas pengukuran pose robot secara manual dan fisik dengan menggunakan alat ukur, seperti penggaris, meteran, busur derajat, dan sebagainya. Dengan demikian, kerja sistem otomatis itu adalah menjejak pose pada titik-titik referensi-pose secara aktual sehingga layak untuk dinamakan sebagai penjejak pose-absolut.

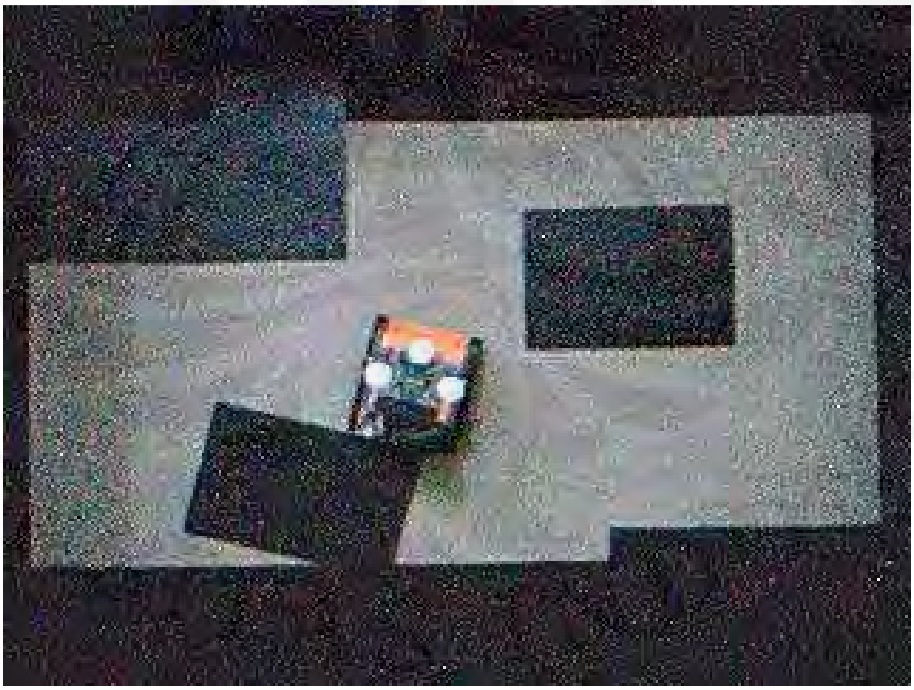
Dalam riset ini dibangun penjejak pose-absolut yang mengandalkan citra 2D hasil jepretan webcam. Tantangan yang dihadapi adalah bagaimana mengekstrak citra robot dalam lingkungan kerjanya menjadi tiga komponen pose-absolut, yaitu x, y, dan θ. Untuk menguraikan solusi atas tantangan itu, pada subbab 3.2 dibahas mengenai ide dasar dari penjejak pose-absolut yang dibangun. Kemudian, implementasi sistem diuraikan di subbab 3.3, yang dilanjutkan dengan hasil pengujian kinerjanya pada subbab 3.4. Bab ini diakhiri dengan pembahasan tentang kemungkinan pengembangan sistem lebih lanjut pada subbab 3.5.

3.2 Prinsip Kerja

Seperti disebutkan barusan, inti sistem penjejak pose-absolut adalah pemrosesan citra (image processing). Oleh karena itu, kualitas dan konfigurasi informasi dalam citra menjadi faktor penentu keberhasilan. Berdasarkan hasil observasi mengenai piranti penghasil citra yang tersedia di pasaran, diputuskan untuk

menggunakan webcam bermerek Logitech QuickCam berspesifikasi: true VGA with resolution 320x160 pixels, CMOS webcam, dan frame-rate 30 fps.

Selanjutnya, konfigurasi informasi pose robot dalam citra dibuat sedemikian rupa sehingga mempermudah ekstraksinya. Untuk itu, citra tersebut berisi penampakan dari atas (bird's view) dari robot dan lingkungannya. Lebih lanjut, di bagian atas robot dipasang tiga buah lingkaran berwarna putih berjari-jari 2 cm yang membentuk segitiga sama-kaki, contoh citra tersebut ditunjukkan pada gambar (3.1). Pemilihan bentuk lingkaran didasarkan pada sifat lingkaran yang mempunyai tingkat kesimetrisan tak berhingga sehingga, jika dilihat dari atas, bentuk tidak akan berubah-ubah walaupun pose robot berubah. Selain itu, warna putih ditujukan untuk mempermudah membedakan lingkaran dengan objek lain di lingkungan yang sebagian besar cenderung berwarna hitam. Jadi, sesungguhnya hanya ada dua warna yang berperan yaitu hitam dan putih sehingga yang dibutuhkan adalah citra hitam-putih.



Gambar 3.1 Citra Robot dan Lingkungannya

Secara garis besar, ide untuk mendapatkan pose adalah dengan mencari posisi tiga buah lingkaran putih. Berbekal posisi-posisi tersebut maka dapat dibentuk segitiga sama-kaki di mana sisi terpanjang (alas segitiga sama-kaki itu) adalah sisi yang berhimpit dengan sumbu Y pada kerangka-lokal robot, sedangkan titik puncak segitiga sama kaki (titik di depan alas) adalah suatu titik di sumbu X+ pada kerangka-lokal robot, yang menandakan bagian depan robot. Lebih lanjut, posisi-absolut adalah letak titik tengah alas segitiga sama-kaki itu dalam kerangka global (world-frame), sedangkan arah-hadap-absolut (orientasi atau bearing) ditentukan sebagai besar sudut yang dibentuk oleh garis tinggi segitiga sama-kaki dengan sumbu X+ pada kerangka global.

3.3 Implementasi

Gagasan untuk mengekstrak posisi tiga lingkaran putih pada citra diimplementasikan dengan suatu software aplikasi yang mempunyai fungsi untuk menkonversi citra RGB hasil jepretan webcam menjadi citra hitam-putih; fungsi tersebut dapat dinotasikan: bw-image = im2bw(rgb-image).

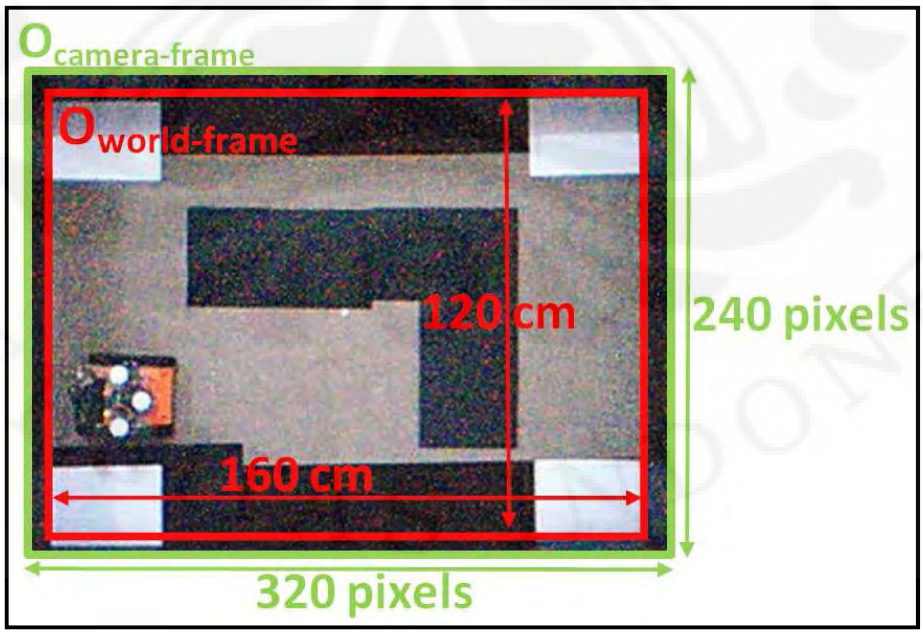
Pada citra hitam-putih, pixel yang berwarna putih bernilai 1 sedangkan yang hitam bernilai 0. Oleh karena itu, langkah berikutnya adalah mencari pixel yang bernilai 1. Setelah itu, diperlukan penyaringan pixel-pixel yang bernilai 1 sehingga tersisa pixel yang benar-benar membentuk suatu lingkaran. Proses penyaringan dilakukan dengan pengecekan terhadap nilai keempat pixel tetangga (atas, bawah, kiri, dan kanan) dari suatu pixel yang bernilai 1. Pixel yang lolos saringan adalah yang mempunyai minimal 3 buah tetangga bernilai 1 dalam satu arah atau 12 tetangga bernilai 1 secara total.

Kemudian, dilakukan clustering yaitu aktivitas mengelompokkan pixel-pixel berwarna putih menjadi tiga kelompok. Akhirnya, pada tiap kelompok dihitung titik pusatnya yang merepresentasikan sebuah posisi lingkaran. Selanjutnya, dicari

dua titik yang menghubungkan alas segitiga sama kaki dan sebuah titik yang menjadi titik puncaknya. Dengan demikian, sampai pada tahap ini, telah ditemukan posisi tiga buah lingkaran putih (pada badan robot bagian atas) yang dideskripsikan dalam koordinat kamera.

Langkah berikutnya adalah melakukan transformasi koordinat dari camera-frame menjadi world-frame. Lebih detil, camera-frame bersatuan pixel dan titik originnya berada tepat di pojok kiri atas citra, sedangkan world-frame bersatuan cm dan titik originnya tidak tepat berhimpit dengan titik origin camera-frame, ditunjukkan pada gambar (3.2). Dengan demikian, dalam melakukan transformasi, diperlukan dua nilai yaitu faktor konversi pixel-cm dan faktor yang harus ditambahkan (offset) karena perbedaan letak kedua titik origin kedua frame tersebut. Untuk sistem penjejak pose-absolut ini, berdasarkan pengukuran, didapatkan nilai faktor konversi pixel-cm dan offset dari citra sebagai berikut:

- OFFSET_X = -7 pixel
- OFFSET_Y = 16 pixel
- PIXEL_PER_CM = 1.8677
- CM_PER_PIXEL = 0.5354

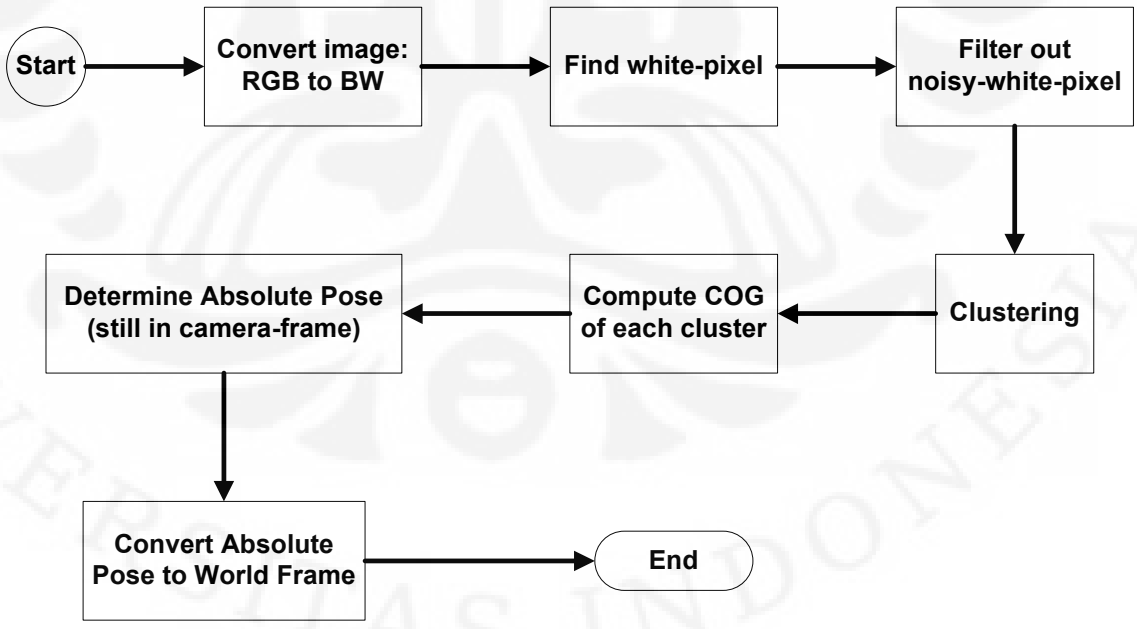


Gambar 3.2 Ilustrasi penentuan faktor konversi pixel-cm dan offset dari citra

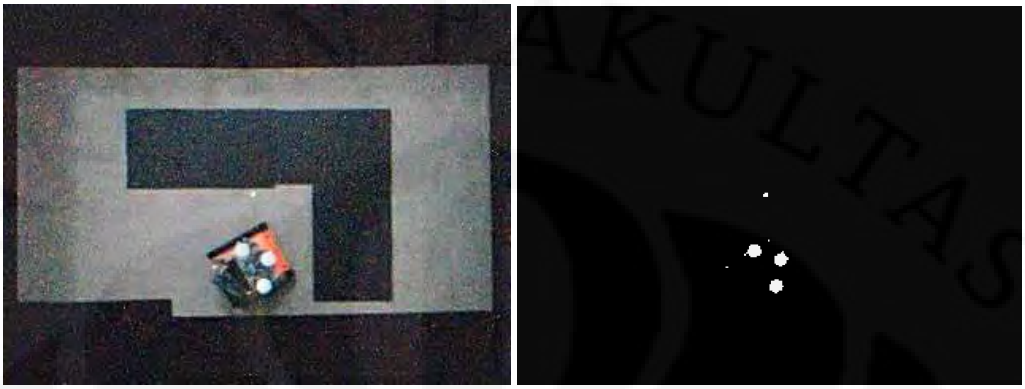
Perlu disampaikan bahwa terdapat distorsi citra hasil jepretan webcam sehingga titik-titik yang berada di pinggir tampak lebih jauh daripada aslinya. Oleh karena itu, dua parameter di atas belum cukup untuk mengkonversi pose robot dari camera-frame ke world-frame, artinya dibutuhkan suatu kalibrasi kamera. Kalibrasi kamera dilakukan dengan 13 titik referensi pada citra dengan cara mencari besar error antara inferensi posisi sebelum dikalibrasi (hanya menggunakan dua faktor di atas) dengan posisi sesungguhnya yang diukur secara nyata atau fisik menggunakan meteran. Hasil kalibrasi berupa persamaan nilai yang perlu ditambahkan (disebut bias) pada tiap titik berupa $bias_X$ dan $bias_Y$, sebagaimana ditunjukkan pada persamaan (3.1) dan (3.2). Akhirnya, secara lugas, alur ekstraksi citra dapat disajikan dalam flowchart pada gambar (3.3) dan beberapa visualisasi kerja sistem pada sebuah citra ditunjukkan pada gambar (3.4), (3.5), dan (3.6).

$$bias_X = -0.071\, position_X + 6.828 \quad (3.1)$$

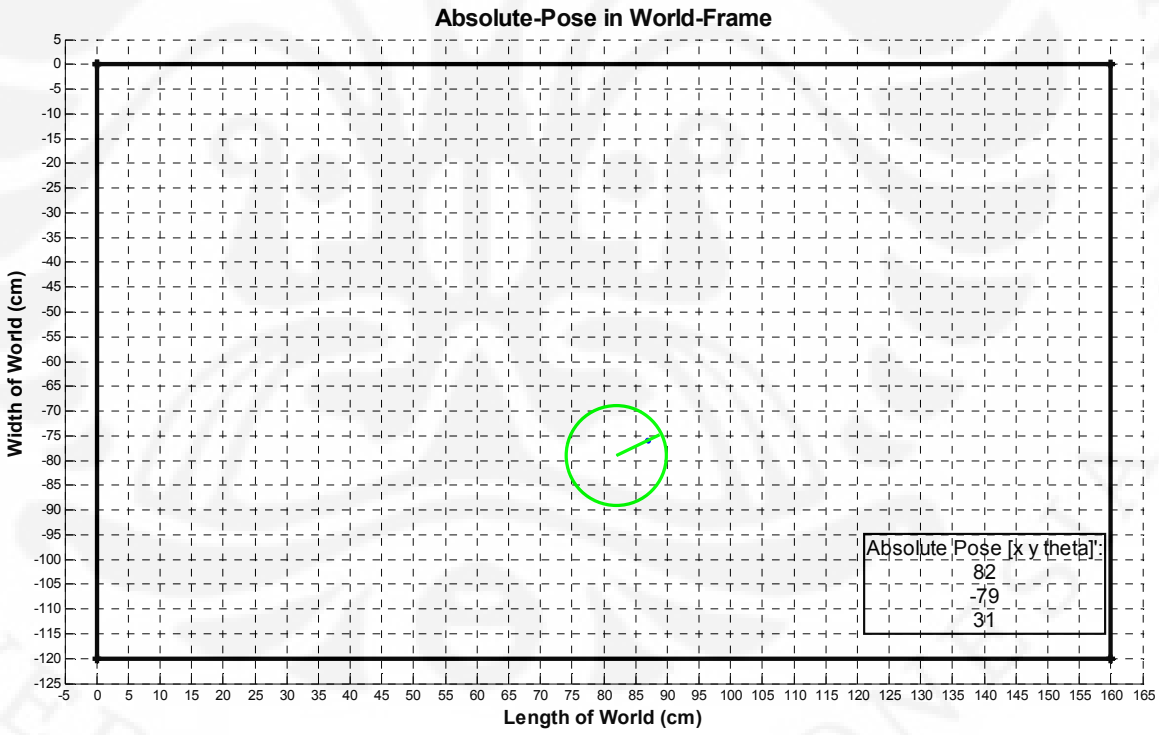
$$bias_Y = -0.080\, position_Y + 4.514 \quad (3.2)$$



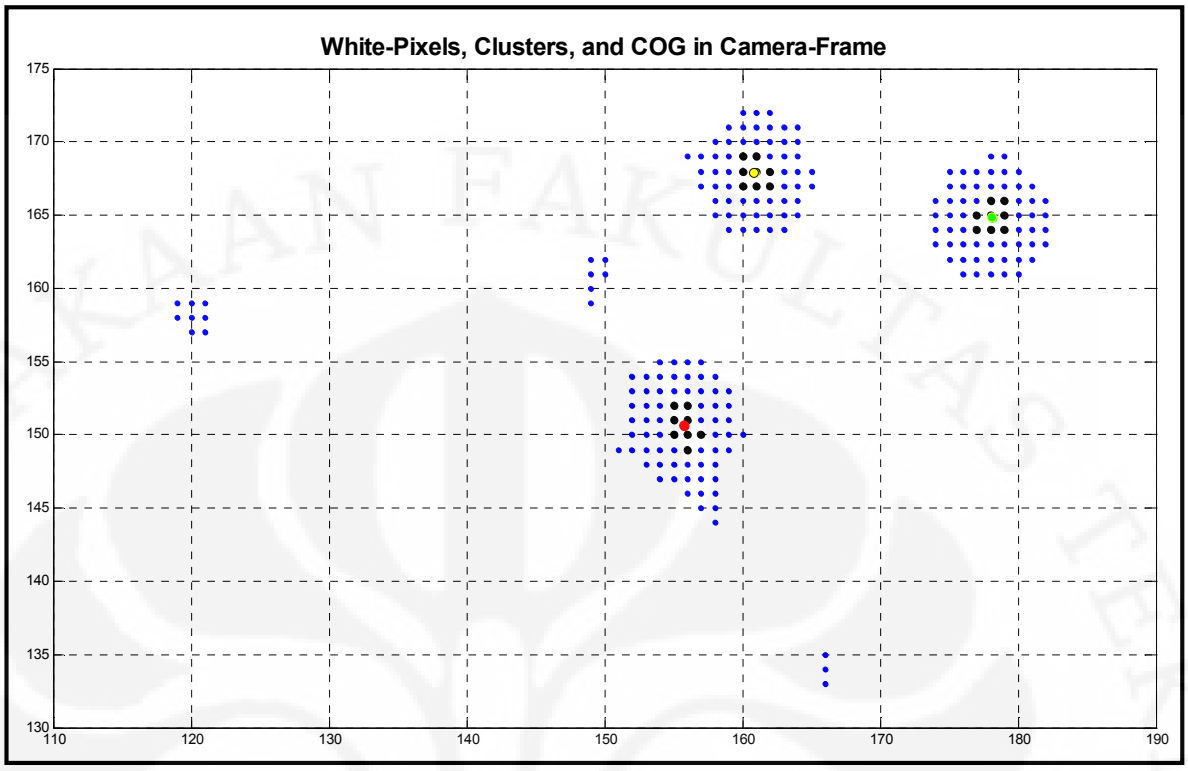
Gambar 3.3 Flowchart Ekstraksi Citra Robot pada Penjejak Pose-Absolut



Gambar 3.4 Citra RGB dan BW-nya



Gambar 3.5 Hasil Akhir dari Penjejak Pose-Absolut



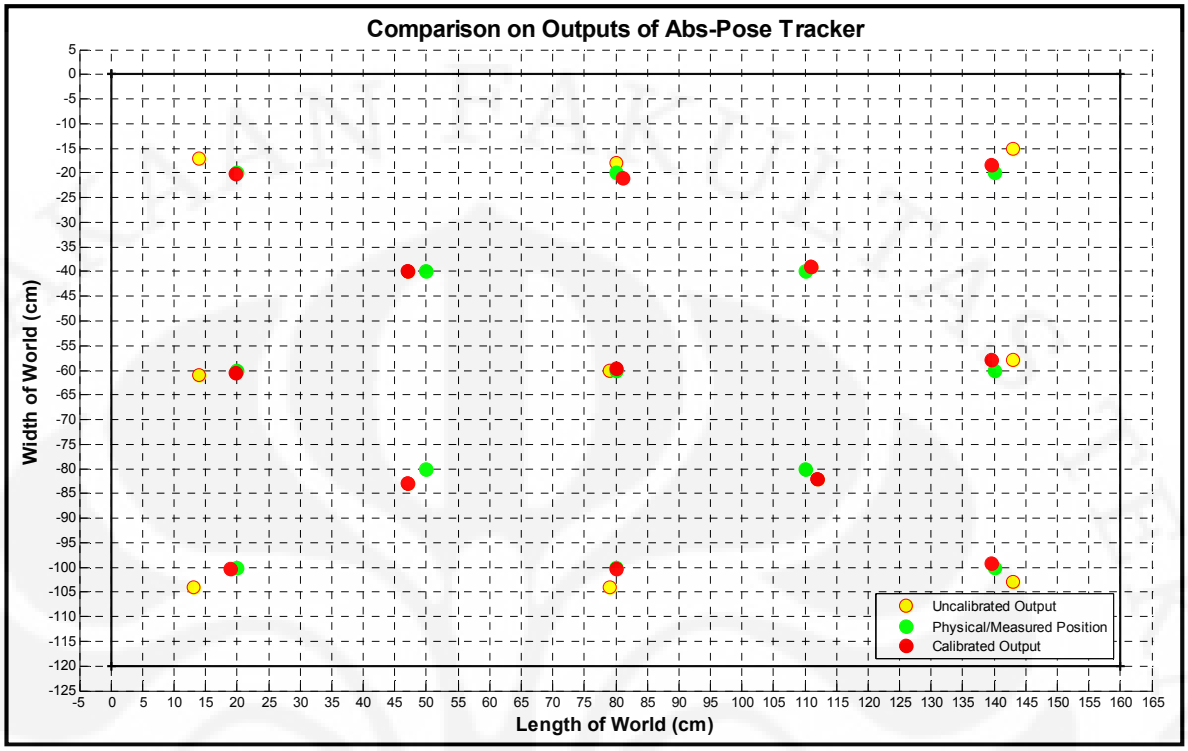
Gambar 3.6 Hasil Proses Filtering dan Clutering terhadap white-pixel;
blue points = white-pixels; black points = filtered white-pixels;
red,green,yellow points = COG of each cluster

3.4 Pengujian Sistem

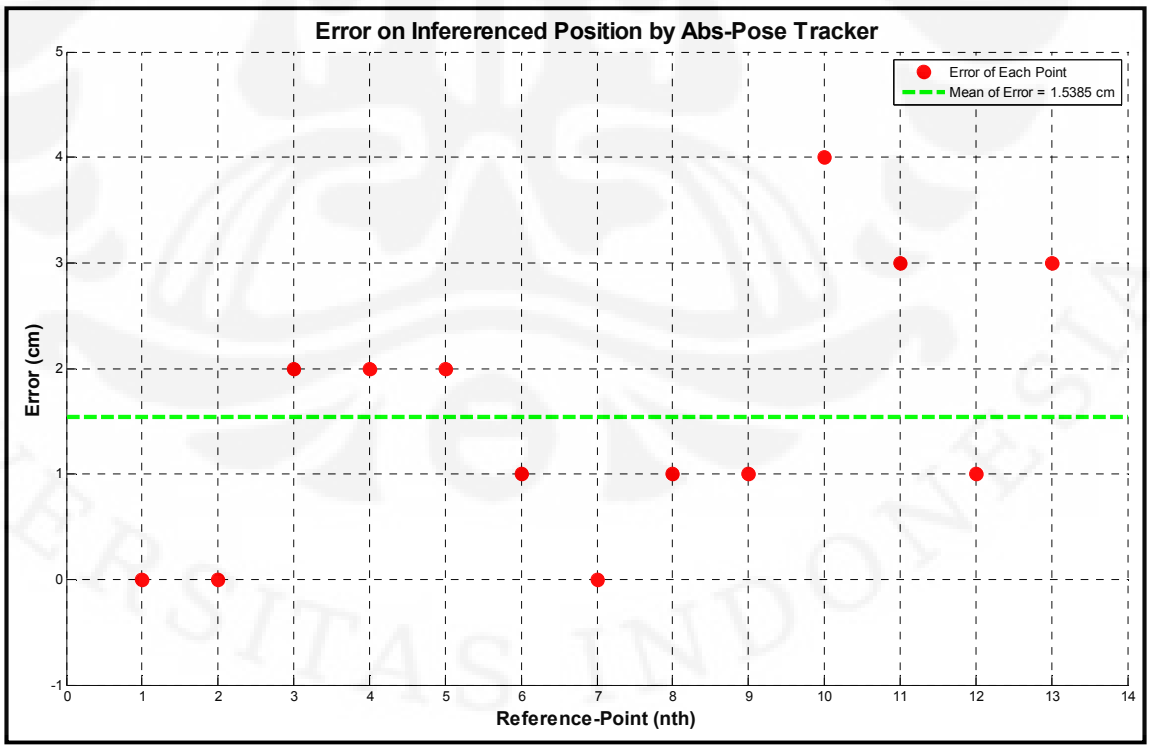
3.4.1 Prosedur Pengujian

Sistem penjejak pose-absolut ini diuji dengan 13 titik referensi. Pada titik-titik tersebut dilakukan pengukuran posisi (nilai x dan y) secara fisik menggunakan meteran. Perlu disampaikan bahwa perlakuan serupa tidak dapat dilakukan untuk arah-hadap robot θ karena tidak ada alat ukur yang sesuai. Oleh karena itu, error besaran itu tidak dapat diidentifikasi secara presisi, hanya berdasarkan observasi dengan toleransi $\pm$15˚. Kemudian, sistem dijalankan untuk menghasilkan pose dari ketigabelas titik referensi tadi. Dengan demikian, posisi-posisi dari 13 titik dapat dibandingkan; antara hasil pengukuran fisik dengan hasil penjejak pose-absolut.

3.2.2 Hasil Pengujian



Gambar 3.7 Hasil Pengujian Kinerja Penjejak Pose Absolut



Gambar 3.8 Error Posisi Hasil Sistem Penjejak Pose Absolut (terkalibrasi)

Berdasarkan hasil pengujian sistem yang ditunjukkan pada gambar (8.7) dan (8.8), didapatkan bahwa rata-rata error posisi adalah 1.5385 cm. Selain itu, berdasarkan observasi, error θ juga masih dalam batas toleransi yang ditetapkan yaitu $\pm 15^{\circ}$. Akhirnya, dinyatakan bahwa sistem tersebut layak untuk dapat digunakan sebagai media validasi pose hasil algoritma self-localization.

3.5 Pengembangan Lebih Lanjut

Sistem penjejak pose-absolut ini menggunakan citra BW sebagai bahan dasar yang merupakan hasil konversi dari citra RGB dengan menggunakan fungsi: bwImage = im2bw(rgbImage, Level). Tampak bahwa parameter kedua fungsi itu adalah batasan yang akan menentukan hasil konversi citra. Pada prakteknya, nilai argumen Level dipengaruhi oleh pencahayaan saat pengambilan gambar, berkisar dari 0.80 sampai dengan 0.90. Oleh karena itu, diakui bahwa sistem ini bersifat sensitif terhadap pencahayaan (lighting). Jika pencahayaan sulit dikontrol maka penentuan level harus dilakukan secara manual. Dengan demikian, di masa datang, akan diadakan perbaikan terhadap hal itu sehingga sistem handal terhadap berbagai kondisi pencahayaan. Alternatif perbaikan yang terpikir saat ini adalah mencoba teknik pemrosesan citra pada berbagai color-space seperti HSV (Hue Saturation Value), HIS (Hue Saturation Intensity), dan CMY (Cyan Magenta Yellow).

Selain itu, tahapan clustering terkadang gagal membagi white-pixels tersaring menjadi tiga kelompok yang unik satu sama lain; satu cluster per lingkaran putih. Kelak, hal tersebut akan diatasi dengan teknik statistik pengelompokan himpunan data yang biasa disebut cluster-analysis.

BAB 4

ODOMETRY PADA TMR ALFATHVRSS

4.1 Pendahuluan

Odometry adalah metode prediksi pose robot dengan memanfaatkan informasi dari wheel-encoder (proprioceptive sensor) yang diolah dengan rumusan matematik yang terhimpun dalam model kinematika robot. Metode tersebut berbeda dengan dead-reckoning, di mana pose diprediksi berdasarkan informasi dari dua jenis sensor yaitu wheel-encoder dan sensor orientasi. Lebih lanjut, odometry adalah hal yang fundamental dalam lokalisasi. Dengan kata lain, odometri yang akurat akan mendukung keberhasilan lokalisasi. Salah satu buktinya adalah peranan odometry dalam metode Probabilistic-Localization Particle-filter di mana ia diandalkan dalam tahap sampling untuk memprediksi pose robot. Selain itu, beberapa fakta penting lain tentang odometry adalah sebagai berikut:

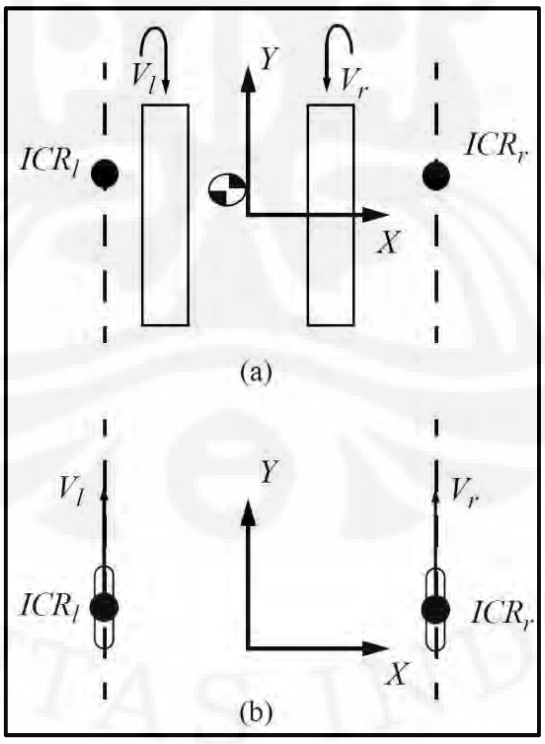
- odometry adalah hal yang mudah dan pertama dapat dilakukan untuk lokalisasi karena hampir tiap mobile robot dilengkapi dengan wheel encoder,
- informasi dari wheel encoder bersifat sederhana dalam pengolahannya,
- odometri yang akurat akan menekan biaya untuk sistem lokalisasi, baik yang menyangkut penyediaan berbagai macam sensor exteroceptive yang dipasang di robot maupun piranti lokalisasi yang dipasang di lingkungan, misal beacon dan landmark khusus.

Bab ini diawali dengan pembahasan tentang model kinematika TMR Alfathvrss; subbab 4.2. Kemudian, proses kalibrasi odometry dengan metode UMBmark diuraikan pada subbab 4.3. Pengujian kinerja lokalisasi dengan odometry terkalibrasi menjadi bagian akhir bab ini.

4.2 Model Kinematika

Dalam [67] disebutkan bahwa TMR merupakan differential-drive mobile robot (DMR) yang menggunakan prinsip skid-steering, di mana variasi kecepatan relatif dari dua track menghasilkan slippage, soil-shearing, dan akhirnya steering. Selain itu, perbandingan antara karakteristik TMR dan DMR adalah sebagai berikut.

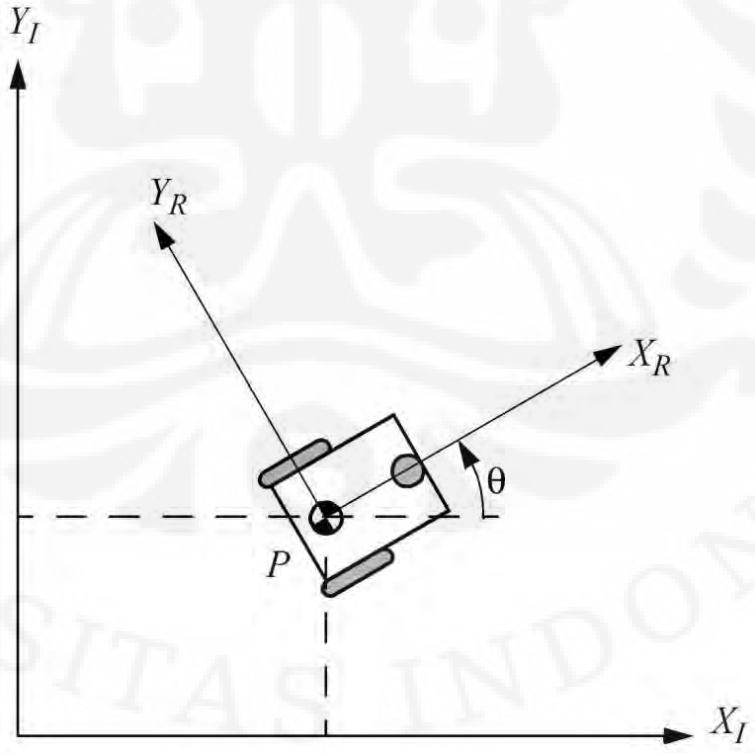
- Berkaitan dengan ICR (Instantaneous Center of Rotation), perbedaan keduanya adalah ICR pada ideal DMR (tanpa lateral-sliding) bersifat konstan dan terletak pada poin kontak roda dengan lantai, sedangkan pada TMR bersifat dynamics-dependent dan selalu berada di luar track karena slippage. Dengan kata lain, semakin sedikit slippage, semakin dekat letak ICR dengan badan robot. Ilustrasi mengenai hal ini ditunjukkan pada gambar (4.1).
- Steering efficiency index, c, pada DMR ideal adalah c = 1, sedangkan TMR c > 1;
- Eccentricity index, e, pada DMR adalah e = 0.



Gambar 4.1 Ekivalensi antara (a) TMR dan (b) DMR untuk Pergerakan Sesaat [67]

Selanjutnya, merujuk pada [63], untuk TMR Alfathvrss $c_{Alfathvrss} = 1.0034 \cong 1$ dan $e_{Alfathvrss} = 0.0015 \cong 0$; nilai kedua parameter tersebut mendekati kepunyaan DMR ideal. Dengan demikian, diputuskan bahwa model kinematika TMR Alfathvrss dapat didekati dengan model kinematika DMR.

Sebagaimana dijelaskan dalam [2], untuk menspesifikasikan posisi robot pada bidang, dibutuhkan suatu hubungan antara kerangka referensi global dan kerangka referensi lokal robot, ditunjukkan pada gambar 4.2. Pada gambar itu, sumbu X_I dan Y_I mendefinisikan sembarang basis inersia pada bidang sebagai kerangka referensi global dari titik origin O; nama lain untuk kerangka ini adalah world-frame (W). Sementara itu, masih pada gambar yang sama, basis $\{X_R, Y_R\}$ mendefinisikan kerangka referensi lokal robot dan merupakan dua sumbu relatif terhadap P yaitu titik referensi posisi; selanjutnya titik P disebut juga sebagai titik R (kependekan dari kata Robot sehingga mudah diingat dan demi konsistensi dengan subscript pada nama sumbu lokal robot).



Gambar 4.2 Kerangka Referensi Global dan Lokal Robot [2]

Posisi R (= P) dalam kerangka referensi global dideskripsikan dengan koordinat x dan y, dan perbedaan sudut antara kerangka referensi global dan lokal dinyatakan dengan θ. Jadi, pose dari robot dalam kerangka referensi global dapat dinyatakan dengan suatu vektor yang memiliki tiga elemen, persamaan (4.1).

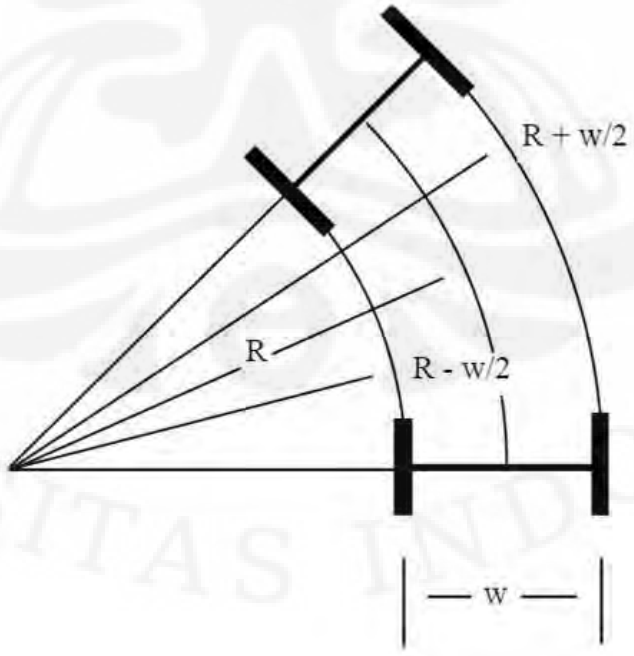
Peranan model kinematika adalah memberikan hubungan antara $pose_t$ dan $pose_{t+1}$ yang ditunjukkan oleh persamaan (4.2).

$$pose_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} \tag{4.1}$$

$$pose_{t+1} = \begin{bmatrix} x_t + \Delta s \cos\theta_{t+1} \\ y_t + \Delta s \sin\theta_{t+1} \\ \theta_t + \Delta\theta \end{bmatrix} \tag{4.2}$$

di mana besaran Δs dan Δθ dapat dijelaskan sebagai berikut.

Pada gambar (4.3), sepasang roda berjalan pada lintasan lingkaran berjari-jari R dengan perubahan sudut Δθ. Untuk wheel base b (= w), yaitu jarak antara dua titik kontak roda dengan lantai, roda kiri berjalan dengan lintasan lingkaran berjari-jari R – b/2 dengan jarak tempuh Δs_l, sedangkan roda kanan berjalan dengan lintasan lingkaran berjari-jari R + b/2 dengan jarak tempuh Δs_r.



Gambar 4.3 Lintasan roda-roda pada DMR

Panjang dari busur Δs dihubungkan dengan sudut Δθ dan jari-jari R melalui persamaan:

$$\Delta s = \Delta\theta\, R \tag{4.3}$$

Dengan demikian,

$$\Delta s_l = \Delta\theta \left(R - \frac{b}{2}\right) \tag{4.4}$$

dan

$$\Delta s_r = \Delta\theta \left(R + \frac{b}{2}\right) \tag{4.5}$$

Hasil pengurangan persamaan (4.5) oleh persamaan (4.4) adalah

$$\Delta s_r - \Delta s_l = \Delta\theta\, b \tag{4.6}$$

Dengan demikian,

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \tag{4.7}$$

Sementara itu, panjang busur dari titik tengah antara roda Δs diberikan dengan menjumlahkan dua persamaan (4.4) dan (4.5) di atas, yaitu:

$$\Delta s_r - \Delta s_l = 2\,\Delta\theta\, R = 2\,\Delta s \tag{4.8}$$

Akhirnya,

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{4.9}$$

Lebih lanjut, hubungan antara perpindahan linear roda Δs dengan keluaran wheel-encoder berupa jumlah pulsa N ditentukan oleh persamaan (4.10).

$$\Delta s = c_m N \tag{4.10}$$

$$c_m = \frac{\pi D_n}{n C_e} \tag{4.11}$$

di mana D_n adalah diameter roda nominal (mm); nominal berarti nilai yang tertera pada form spesifikasi pabrik, dan C_e adalah resolusi encoder (ppr). Dengan demikian, c_m merupakan faktor konversi untuk mendapatkan perpindahan linear roda dari perpindahan sudut roda dalam N.

Untuk TMR Alfathvrss, merujuk pada [63], diameter roda Dn didefinisikan sebagai hasil pengukuran diameter sprocket ditambah dengan tebal track yang melilitnya, gambar (4.4), yaitu Dn = 7.2 cm. Dengan demikian, nilai parameter c_m pada persamaan (4.11) adalah $c_m = 0.0362$.

Gambar 4.4 Pendefinisian Dn pada TMR Alfathvrss [63]

4.3 Kalibrasi Odometry

Dapat diklaim bahwa keuntungan odometry terletak pada sifatnya yang sederhana, tidak mahal, dan mudah dalam implementasi. Akan tetapi, error padanya akan terakumulasi secara tidak terbatas. Selain itu, mengacu pada persamaan (4.10), akurasi odometri bergantung pada asumsi bahwa revolusi roda (perpindahan sudut) dapat dikonversi menjadi perpindahan linear roda relatif terhadap lantai.

Namun demikian, asumsi tersebut memiliki validitas yang terbatas, salah satu contohnya adalah fenomena wheel-slippage. Ilustrasinya (yang sengaja dilebih-lebihkan) adalah sebagai berikut: misal roda berputar pada lantai dengan tumpahan oli, maka perputaran roda akan mengakibatkan perputaran wheel-encoder tetapi tidak menyebabkan perpindahan linear roda karena permukaan lantai yang sangat licin berlumuran oli.

Dalam [3] dirinci faktor-faktor yang menyebabkan penurunan akurasi odometri berkaitan dengan konversi perpindahan sudut ke perpindahan linear. Mereka dapat digolongkan menjadi dua tipe, yaitu non-systematic error dan systematic error. Penjelasan keduanya adalah sebagai berikut:

- Non-systematic-error disebabkan oleh interaksi robot dengan fitur tak terduga di lingkungan. Sebagai contoh: ketidateraturan permukaan lantai, seperti tonjolan, retakan, sampah, yang menyebabkan sebuah roda berotasi lebih dari yang diprediksi oleh persamaan (4.10). Dengan demikian, non-systematic-error adalah fungsi dari karakteristik lantai di mana robot bekerja.

- Systematic-error disebabkan oleh ketidaksempurnaan desain dan implementasi mekanik dari mobile robot; dua penyebabnya yang terpenting adalah diameter roda yang tidak identik E_d, (persamaan 4.12), dan ketidakpastian terhadap wheel-base yang efektif E_b ,(persamaan 4.13). Lebih lanjut, keduanya memiliki efek yang spesifik dan berbeda yaitu E_b pada saat perputaran (*turning*) dan E_d pada pergerakan lintasan lurus. Keduanya adalah besaran yang tidak bersatuan.

$$E_d = \frac{D_r}{D_l} \tag{4.12}$$

$$E_b = \frac{b_{actual}}{b_{nominal}} \tag{4.13}$$

di mana: D_r adalah diameter roda kanan aktual, D_l adalah diameter roda kiri aktual, dan b adalah wheel-base dari mobile robot yaitu jarak antara titik kontak dari dua roda dan lantai.

Akhirnya, untuk mengatasi kekurangan tersebut sekaligus mengoptimalkan akurasinya, maka dilakukan kalibrasi odometri dengan metode UMBmark [68]. Secara lebih spesifik, kalibrasi tersebut ditujukan hanya untuk menanggulangi systematic-error dengan alasan-alasan sebagai berikut:

- Systematic-error bersifat khas pada suatu mobile robot dan biasanya tidak berubah selama periode tertentu. Dengan demikian, akurasi odometri dapat ditingkatkan dengan pengukuran terhadap penyebabnya yang paling dominan dan kemudian melawan atau menghilangkan efeknya melalui pendekatan artificial intelligent dalam bentuk software atau kode program.
- Systematic-error adalah hal yang penting karena bersifat terakumulasi secara konstan. Pada pemukaan indoor yang halus dan atau teratur (sebagimana lingkungan yang dipakai di riset ini), systematic-error lebih dominan daripada nonsystematic-error.

Keunggulan metode kalibrasi odometri UMBmark terletak pada prosedurnya yang sistematik dan menjanjikan hasil yang optimal. Hal tersebut bertolak belakang dengan metode lain yang biasanya bersifat trial-and-error, time-consuming, dan bahkan seringkali hanya mengandalkan intuisi. Selain itu, metode UMBmark siap untuk diadaptasi menjadi prosedur kalibrasi otomatis asalkan terdapat sensor dengan tingkat akurasi dan presisi yang tinggi.

Lebih lanjut, akurasi odometry dinyatakan secara kuantitatif dengan nilai systematic-error odometry yaitu $E_{max,syst}$. Sementara itu, parameter odometry belum terkalibrasi untuk TMR Alfathvrss adalah nilai rasio diameter roda yaitu E_d = 1 dan nilai rasio wheelbase Eb = 1 yang menyebabkan b = 203 mm. Berdasarkan [63], nilai systematic-error odometri TMR Alfathvrss belum terkalibrasi secara kuantitatif adalah:

$$E_{max,syst} = \max(r_{c.g,cw}, r_{c.g,ccw}) = 87.3332 \quad (4.14a)$$

$$E_{cw,syst} = r_{c.g,cw} = 62.0854 \quad (4.14b)$$

$$E_{ccw,syst} = r_{c.g,ccw} = 87.3332 \quad (4.14c)$$

Eksekusi kalibrasi odometry UMBmark menghasilkan Ed = 1.1589 yang memiliki arti fisis bahwa diameter roda kanan dan kiri pada TMR Alfathvrss berbeda; tidak lagi Ed = 1 sebagaimana nilai belum terkalibrasi. Alasan logis mengenai hal tersebut adalah fakta bahwa definisi roda pada TMR Alfathvrss berupa sprocket yang dililiti track, lagipula, titik-titik lilit bersifat dinamis atau tidak tetap sehingga memungkinkan terjadi penambahan atau pengurangan dimensi roda pada saat berjalan.

Untuk memanfaatkan pengetahuan tentang parameter systematic-error odometry Ed maka dimunculkan faktor konversi c yang ditanamkan dalam persamaan (4.10) dengan cara dijadikan pengali pada ruas kanannya; c_L untuk roda kiri dan c_R untuk roda kanan. Faktor konversi tersebut adalah sebagai berikut:

$$c_L = \frac{2}{Ed+1} = 0.9264 \quad (4.15a)$$

$$c_R = \frac{2}{(1/Ed)+1} = 1.0736 \quad (4.15b)$$

Selain Ed, kalibrasi odomery menghasilkan Eb = 1.0977, yang dimanfaatkan untuk menghitung nilai wheelbase yang aktual b_{actual}, yaitu

$$b_{actual} = Eb \times b_{nominal} = 22.2827 \cong 22.3 \quad (4.16)$$

Selanjutnya nilai b dalam program odometri yang semula bernilai 203 mm diganti dengan nilai b_{actual} di atas yaitu 223 mm.

Hal yang menarik adalah kenyataan bahwa nilai wheelbase b terkalibrasi lebih besar daripada lebar TMR Alfathvrss; b_{actual} = 223 mm > 203 mm. Jika mengingat kembali definisi wheelbase b sebagai jarak titik kontak dua roda dengan lantai maka kenyataan itu tampak tidak masuk akal. Namun demikian, perlu diingat kembali bahwa robot Alfathvrss berjenis TMR yang memang dalam eksperimen ini model kinematikanya didekati dengan model kinematika DMR (di mana definisi wheelbase b yang barusan ditujukan). Selain itu, karakteristik titik kontak roda (track) dengan lantai pada TMR adalah multikontak. Hal tersebut juga menjadi suatu kendala untuk menerima secara mentah hasil kalibrasi wheelbase b kalaupun nilainya lebih kecil atau sama dengan lebar TMR Alfathvrss.

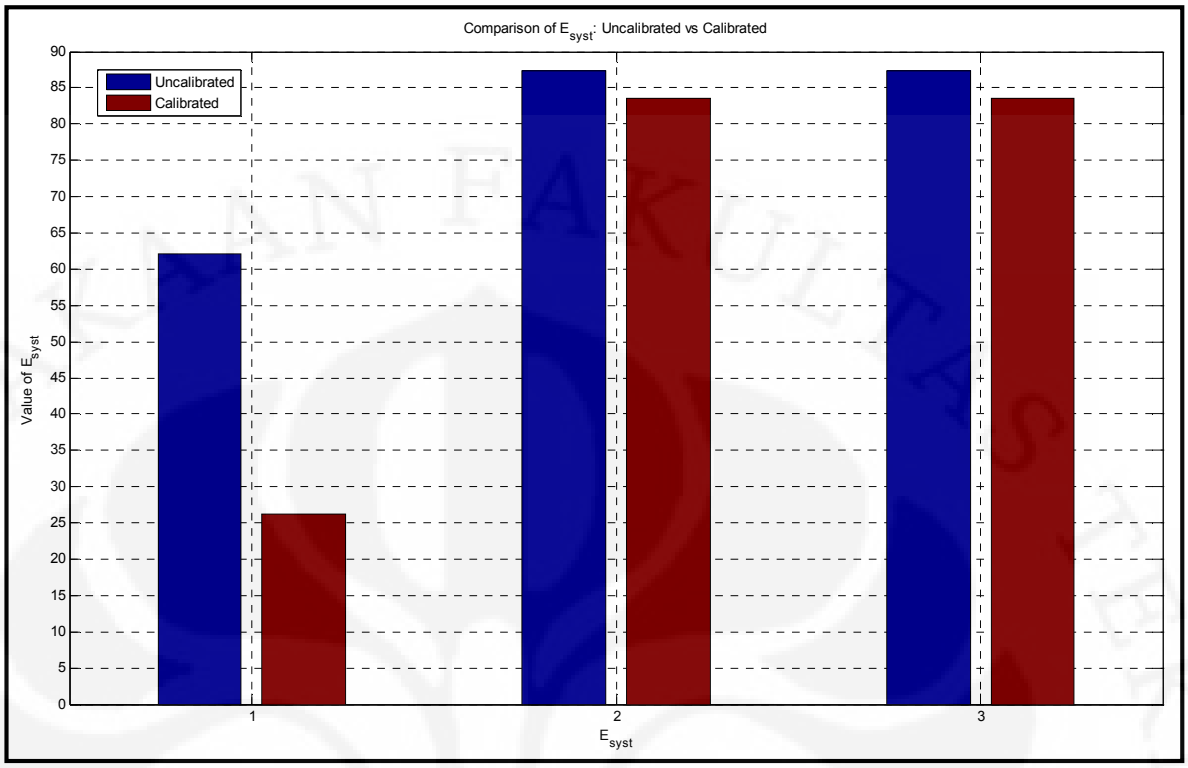
Dengan demikian, definisi wheelbase b pada TMR perlu memiliki istilah lain yaitu auxiliary-wheelbase. Nilai b sebagai auxiliary-wheelbase berarti bahwa nilai tersebut tidak perlu dapat diterjemahkan secara fisis atau nyata, tetapi memiliki peran penting jika model kinematika TMR didekati dengan model kinematika DMR. Selain itu, berdasarkan gambar (4.1), dapat diklaim bahwa ada kesesuaian antara nilai b terkalibrasi dengan letak ICR-track. Oleh karena itu, dinyatakan bahwa definisi auxiliary-wheelbase pada TMR adalah jarak antara dua ICR-track-nya, tidak lagi sebagai jarak dua titik kontak roda dengan lantai sebagaimana pada DMR.

Akhirnya, dapat dihitung systematic-error odometri setelah dilakukan kalibrasi, persamaan (4.17) dan perbandingan dengan nilai sebelum dan sesudah kalibrasi, gambar (4.5). Berdasarkan keduanya, disimpulkan bahwa kalibrasi odometri dengan metode UMBmark yang dilaksanakan sebanyak satu iterasi berhasil mengurangi systematic-error odometri: secara signifikan pada arah-gerak CW (menjadi bernilai 0.4224 kali nilai belum terkalibrasi), tetapi tidak begitu signifikan pada arah-gerak CCW (menjadi bernilai 0.9562 kali nilai belum terkalibrasi).

$$E_{cw,syst,cal} = r_{c.g,cw,cal} = 26.2256 \quad (4.17a)$$

$$E_{ccw,syst,cal} = r_{c.g,ccw,cal} = 83.5085 \quad (4.17b)$$

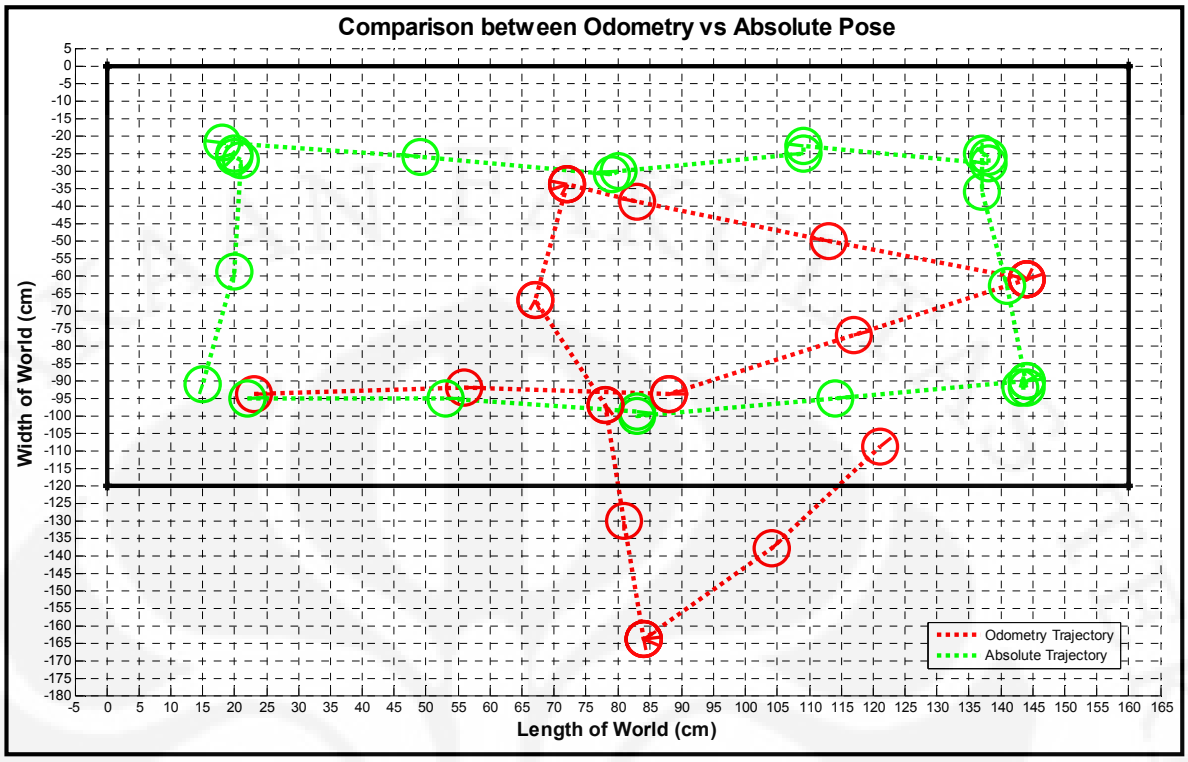
$$E_{max,syst,cal} = \max\left(r_{c.g,cw,cal}, r_{c.g,ccw,cal}\right) = 83.5085 \quad (4.17c)$$



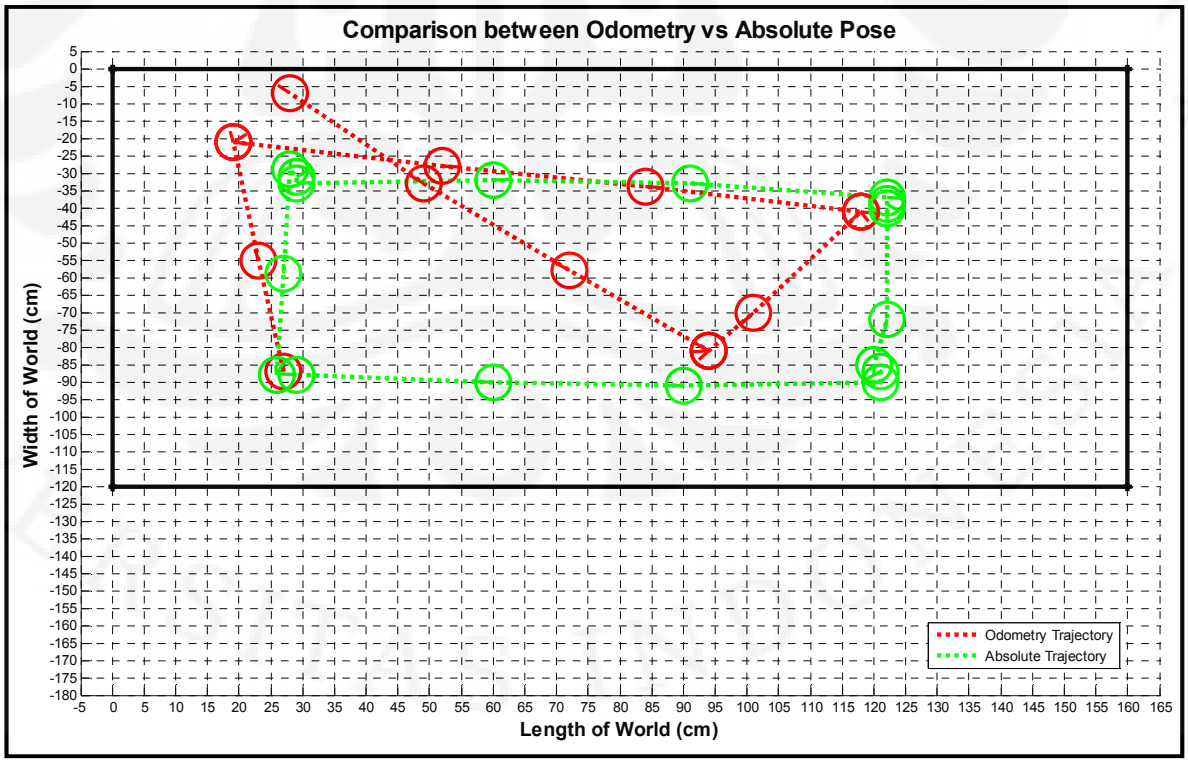
Gambar 4.5 Perbandingan Systematic-Error Odometri: Sebelum dan Sesudah Kalibrasi
Keterangan sumbu E_{syst}: 1. $E_{cw,syst,cal}$; 2. $E_{ccw,syst,cal}$, 3. $E_{max,syst,cal}$

3.4 Pengujian Kinerja Odometry

Pengujian kinerja odometry dilakukan dengan mengontrol robot untuk bergerak mengelilingi lingkungan dalam arah-gerak CCW dan CW. Hasilnya ditunjukkan pada gambar (4.6) dan (4.7). Tampak bahwa, untuk TMR Alfathvrss, error odometry relatif besar, walupun telah dikalibrasi. Error bernilai besar saat robot berputar. Oleh karena itu, dapat disimpulkan bahwa penyebab dominan dari systematic error pada TMR adalah ketidakpastian terhadap wheel-base yang efektif, E_b. Selain itu, berdasarkan gambar (4.6) dan (4.7), dapat ditunjukkan juga bahwa systematic error terkalibrasi arah-gerak CW $E_{cw,syst,cal}$ lebih kecil daripada arah-gerak CCW $E_{ccw,syst,cal}$. Hal tersebut selaras dengan data pada gambar (4.5). Begitu buruknya odometry pada TMR Alfathvrss, sampai-sampai, akumulasi error yang terjadi menyebabkan pose hasil lokalisasi dengan odometry berada di luar lingkungan robot untuk arah-gerak CCW, gambar (4.6).



Gambar 4.6 Perbandingan Pose-Odometry dengan Pose-Absolut dalam arah-gerak CCW



Gambar 4.7 Perbandingan Pose-Odometry dengan Pose-Absolut dalam arah-gerak CW

BAB 5

MODEL GERAKAN PROBABILISTIK TMR ALFATHVRSS

5.1 Pendahuluan

Salah satu sumber ketidakpastian dalam robot adalah kerja actuator-nya. Dalam hal lokalisasi, actuator yang menjadi fokus utama adalah yang menyebabkan perubahan pose, yaitu motor. Lebih lanjut, TMR Alfathvrss mempunyai dua buah motor sehingga tergolong 2-DOF mobile robot. Dengan demikian, gerakannya berupa gerakan maju-mundur (forward dan backward) dan gerakan putar-kanan-kiri (rot-CW dan rot-CCW).

Selanjutnya, dalam rangka mengimplementasikan metode probabilistik untuk lokalisasi, gerakan-gerakan tersebut perlu dimodelkan secara probabilistik; hasilnya disebut model gerakan probabilistik. Representasinya adalah sebuah probability density function (PDF) dari suatu conditional density yang ditunjukkan pada persamaan (5.1).

$$p(pose_t|u_t, pose_{t-1}) \tag{5.1}$$

di mana $pose_t$ dan $pose_{t-1}$ adalah pose-pose robot, sedangkan u_t adalah informasi kontrol robot.

Ada dua jenis model gerakan robot yaitu velocity motion model dan odometry motion model. Dalam model yang pertama, informasi kontrol dispesifikasikan dengan perintah kecepatan yang diberikan ke motor-motor, sedangkan pada model kedua, informasi kontrol dideskripsikan oleh kuantitas perubahan pose robot. Odometry motion model lebih akurat daripada model yang lain, tetapi bersifat post-the-fact karena kebutuhan terhadap pose akhir. Dalam [3] disimpulkan bahwa odometry motion model ditujukan untuk aplikasi prediksi atau estimasi,

sementara itu, velocity motion model untuk aplikasi motion planning. Karena model gerakan probabilistik dalam MCL berperan dalam memprediksi gerakan robot maka dipilih odometry motion model untuk memodelkan gerakan TMR Alfathvrss.

Odometry motion model dijelaskan mendetil pada subbab 5.2. Setelah itu, pada subbab 5.3 dipaparkan PDF baik untuk model parametrik maupun non-parametrik yang diolah dari data hasil eksperimen gerakan TMR Alfathvrss. Sebagaimana disebutkan tadi tentang fungsi model gerakan probabilistik pada algoritma MCL, maka diperlukan ekperimen sampling, di mana hasil dan analisisnya diuraikan pada subbab 5.4.

5.2 Odometry Motion Model (OMM)

Sebagaimana dijelaskan di subbab yang lalu, perbedaan kedua model gerakan terletak pada pendeskripsi informasi kontrol. Pada OMM, dibutuhkan dua pose robot (yaitu $pose_{t-1}$ dan $pose_t$) untuk membentuk informasi kontrol atau untuk mendapatkan perubahan pose.

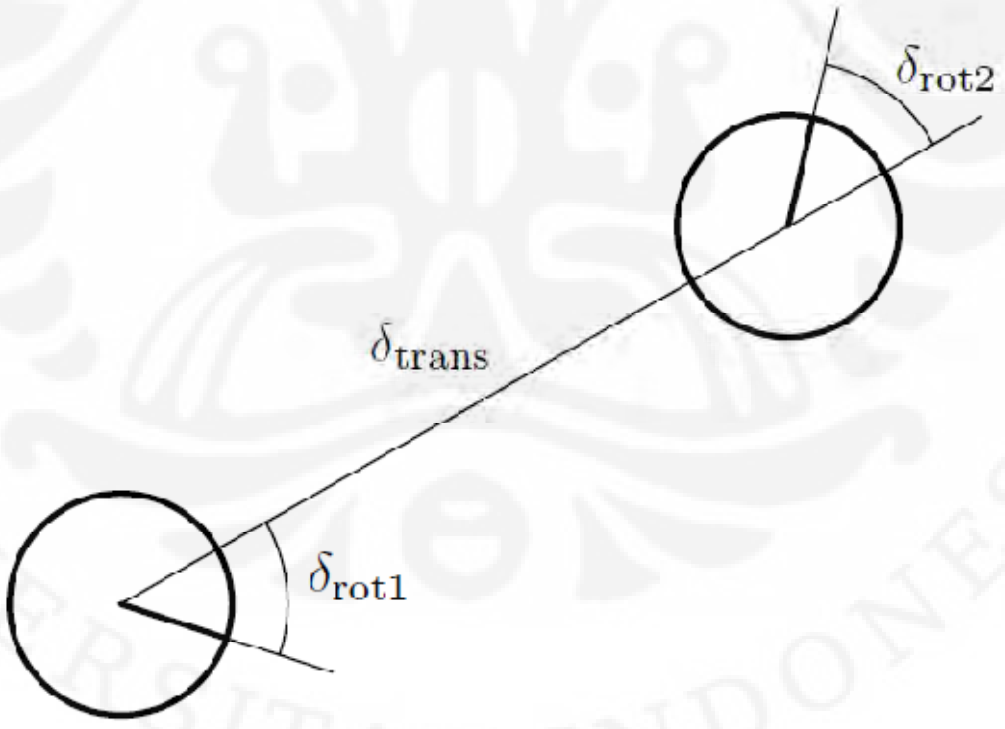
Pose-pose yang dibutuhkan tersebut adalah pose hasil lokalisasi dengan odometry (dijelaskan pada Bab 4). Perlu diketahui bahwa, sesungguhnya, jika odometry tidak memiliki systematic-error dan non-systematic-error, lokalisasi sudah cukup dilakukan dengannya saja. Namun, kenyataannya tidak demikian sehingga diperlukan usaha tambahan yaitu lokalisasi dengan MCL. Dengan kata lain, lokalisasi dengan MCL adalah pelengkap (complement) dari lokalisasi dengan odometry; bukan pengganti (substitute).

Dengan demikian, pada OMM, hasil pembacaan sensor encoder yang diistilahkan odometry-reading dijadikan deskripsi untuk informasi kontrol yang pada akhirnya akan diolah menjadi suatu kuantitas perubahan pose, persamaan (5.2)

$$u_t = \begin{bmatrix} poseOdo_t \\ poseOdo_{t-1} \end{bmatrix} \quad (5.2)$$

di mana $poseOdo_t$ dan $poseOdo_{t-1}$ adalah pose hasil lokalisasi dengan odometry, secara berurutan, pada waktu t dan t-1.

Kemudian, untuk menghitung kuantitas dari perubahan pose, diperlukan suatu dekomposisi dari gerakan. Gambar (5.1) mengilustrasikan proses tersebut.



Gambar 5.1 Dekomposisi Gerakan Robot [3]

Dekomposisi gerakan robot menggunakan prinsip bahwa semua gerakan robot penyebab perubahan pose dapat diurai menjadi tiga sub-gerakan yang tidak dapat diurai lagi (sub-gerakan bersifat atomic) yaitu sub-gerakan putar di awal, sub-gerakan translasi, dan sub-gerakan putar di akhir. Secara berurutan, nilai ketiganya disimbolkan dengan δ_{rot1}, δ_{trans}, dan δ_{rot2}.

Akhirnya, kuantitas perubahan pose (hasil suatu gerakan) dapat dinyatakan dengan ketiga besaran tadi dengan formula sebagai berikut:

$$\delta_{rot1} = atan2\left(y_{Odo,t} - y_{Odo,t-1}, x_{Odo,t} - x_{Odo,t-1}\right) - \theta_{Odo,t-1} \quad (5.3)$$

$$\delta_{trans} = \sqrt{\left(x_{Odo,t} - x_{Odo,t-1}\right)^2 + \left(y_{Odo,t} - y_{Odo,t-1}\right)^2} \quad (5.4)$$

$$\delta_{rot2} = \theta_{Odo,t} - \theta_{Odo,t-1} - \delta_{rot1} \quad (5.5)$$

Dengan demikian,

$$u_t = \begin{bmatrix} poseOdo_t \\ poseOdo_{t-1} \end{bmatrix} = \begin{bmatrix} \delta_{rot1} \\ \delta_{trans} \\ \delta_{rot2} \end{bmatrix} \quad (5.6)$$

Keluaran OMM adalah prediksi pose robot yang didapatkan dengan mengolah informasi kontrol dan pose robot pada satu waktu sebelumnya. Prediksi pose tersebut diharapkan lebih akurat daripada pose hasil lokalisasi dengan odometry karena memperhitungkan odometry-error. Oleh karena itu, untuk memprediksi pose, dibutuhkan pula prediksi terhadap δ_{rot1}, δ_{trans}, dan δ_{rot2}, yaitu:

$$\hat{\delta}_{rot1} = \delta_{rot1} + \text{error}_{\text{rot1}} \quad (5.7)$$

$$\hat{\delta}_{trans} = \delta_{trans} + \text{error}_{\text{trans}} \quad (5.8)$$

$$\hat{\delta}_{rot2} = \delta_{rot2} + \text{error}_{\text{rot2}} \quad (5.9)$$

Selanjutnya, pose terprediksi dapat dihitung berdasarkan model kinematika TMR sebagaimana dijelaskan dalam Bab 4, sebagai berikut.

$$x_t = x_{t-1} + \hat{\delta}_{trans} \cos(\theta_{t-1} + \hat{\delta}_{rot1}) \quad (5.10)$$

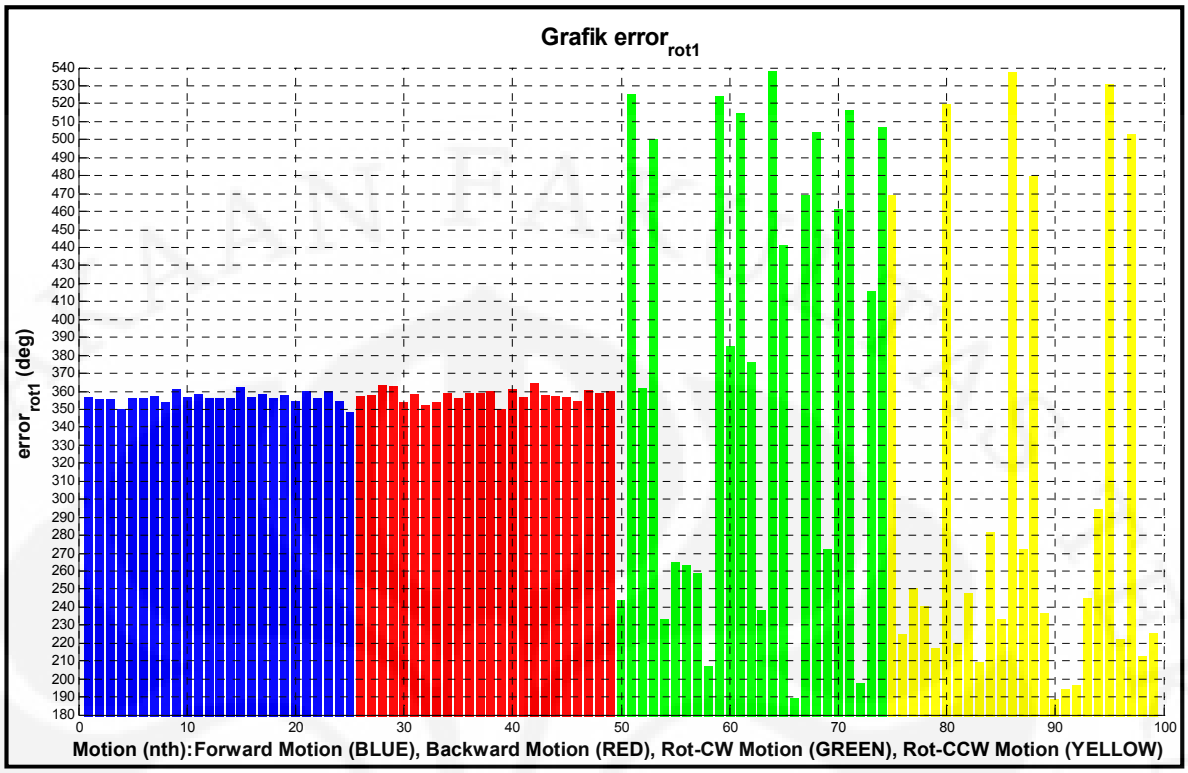
$$y_t = y_{t-1} + \hat{\delta}_{trans} \sin(\theta_{t-1} + \hat{\delta}_{rot1}) \quad (5.11)$$

$$\theta_t = \theta_{t-1} + \hat{\delta}_{rot1} + \hat{\delta}_{rot2} \quad (5.12)$$

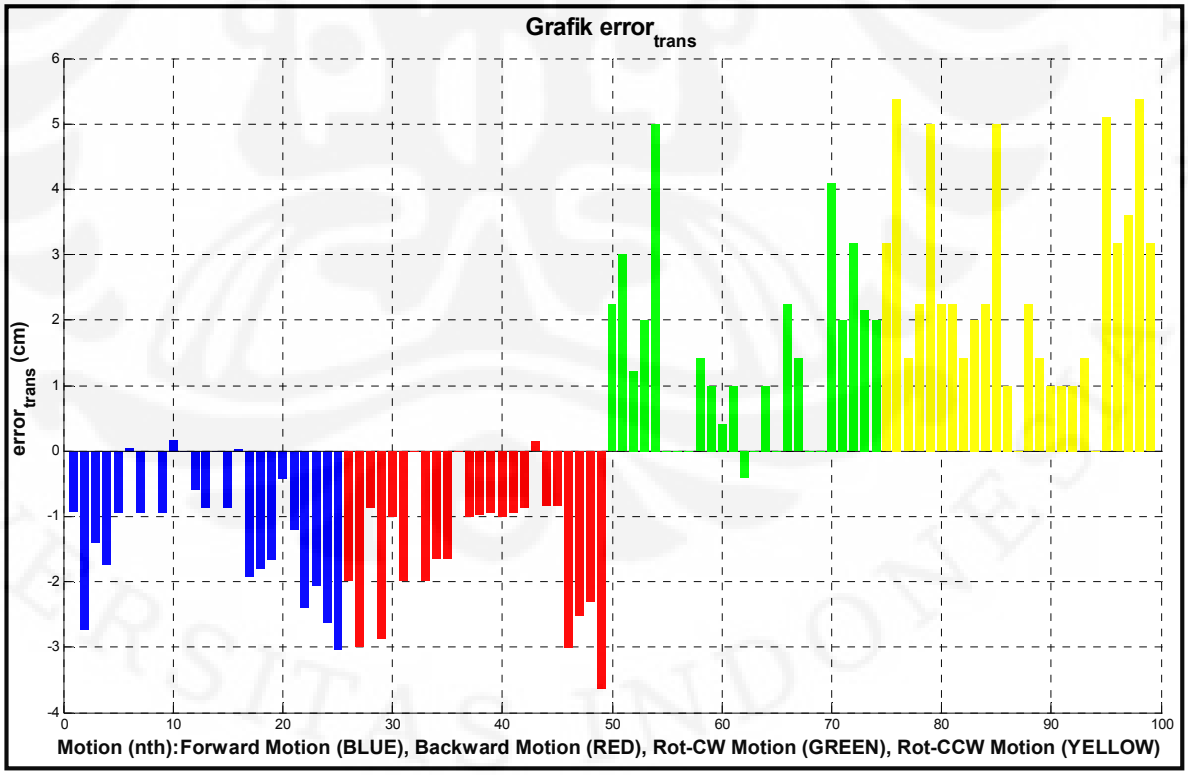
5.3 Eksperimen odometry-error

Eksperimen ini bertujuan mendapatkan data odometry-error. Data tersebut diperoleh dengan menggunakan pose-pose absolut untuk mendapatkan $\hat{\delta}_{rot1}$, $\hat{\delta}_{trans}$, dan $\hat{\delta}_{rot2}$. Sementara itu, δ_{rot1}, δ_{trans}, dan δ_{rot2} diperoleh dari pose-pose hasil lokalisasi dengan odometry.

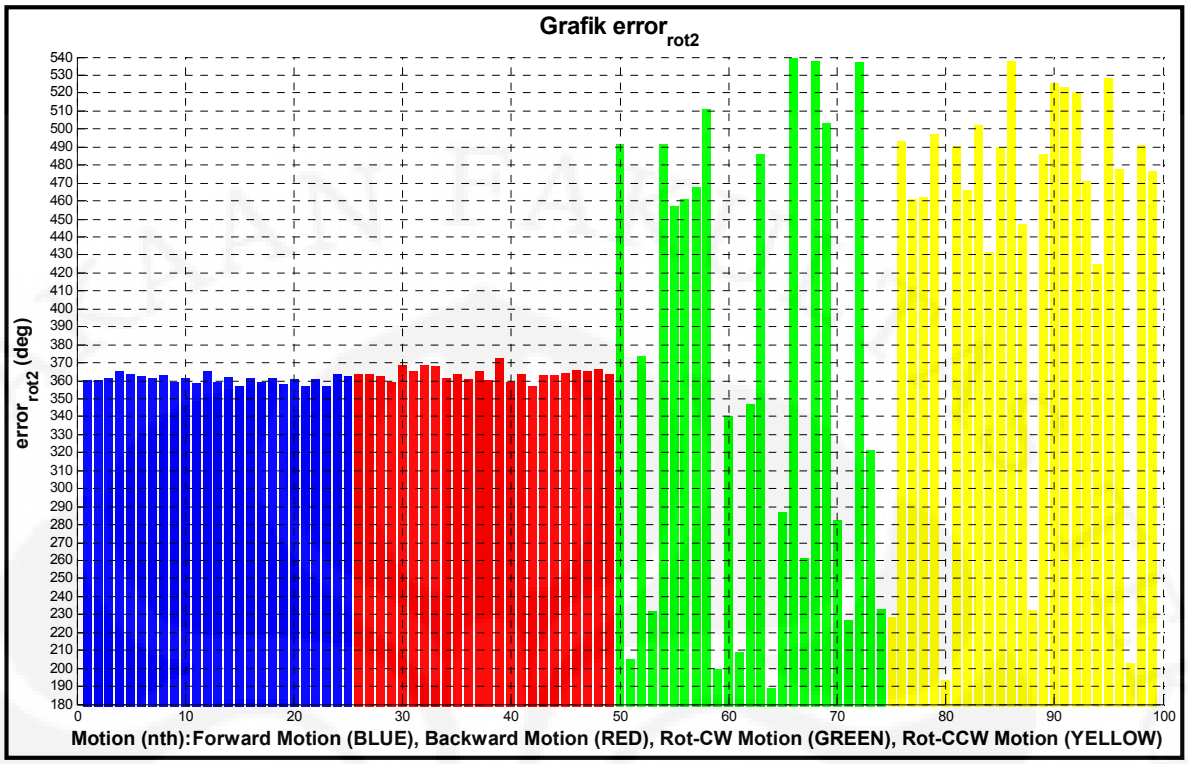
Dalam eksperimen ini, dilakukan 99 gerakan untuk memproduksi perubahan pose, secara berurutan, terdiri dari 25 gerakan maju (forward), 24 gerakan mundur (backward), 25 gerakan putar-kanan (rot-CW), dan 25 gerakan putar-kiri (rot-CCW). Nilai-nilai untuk $\text{error}_{\text{rot1}}$, $\text{error}_{\text{trans}}$, dan $\text{error}_{\text{rot2}}$ ditampilkan dengan bar-graph pada gambar (5.2), (5.3), dan (5.4) berikut.



Gambar 5.2 Grafik $error_{rot1}$



Gambar 5.3 Grafik $error_{trans}$



Gambar 5.4 Grafik $error_{rot2}$

5.4 Probability Distribution untuk Model Gerakan

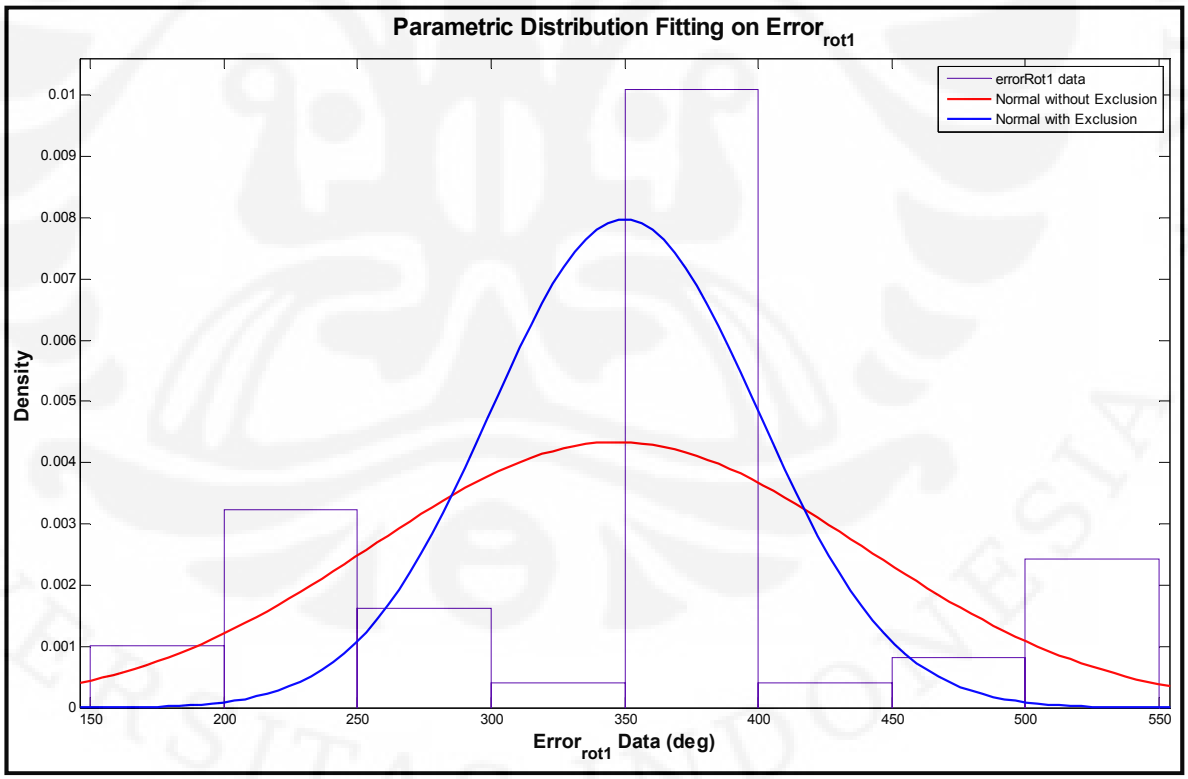
Merujuk pada pembahasan subbab 5.2, dapat disimpulkan bahwa inti dari pemodelan gerakan adalah memodelkan odometry-error yang berarti memilih probability distribution yang sesuai untuk data tersebut.

5.4.1 Model Gerakan dengan Parametric Probability Distribution

Mula-mula, probability distribution yang dipilih untuk mendekati kelakuan data odometry-error adalah Distribusi Normal (Gaussian) karena distribusi tersebut, berdasarkan pengalaman, paling besar kemungkinannya untuk mendapatkan nilai parameter yang mewakili data percobaan. Dengan kata lain, data odometry-error diasumsikan bersifat normal (Gaussian). Lebih detil lagi, nilai parameter yang dipakai adalah yang memiliki maximum likelihood dalam memproduksi data. Jadi, model gerakan diperoleh dengan melakukan fitting antara Parametric Probability Distribution Normal dengan data odometry-error .

Hasil fitting untuk $error_{rot1}$ ditampilkan pada gambar (5.5). Tampak pada gambar tersebut bahwa dilakukan dua jenis fitting untuk distribusi Normal, yaitu without-exlusion dan with-exclusion. Jenis yang terakhir adalah melakukan fitting pada data dengan syarat data tersebut berada pada kisaran lebih dari 240˚ (= 360˚ - 120˚) dan kurang dari 480 (= 360˚ + 120˚). Justifikasi tersebut diambil berdasarkan observasi langsung terhadap pergerakan robot selama eksperimen. Catatan detil mengenai proses dan hasil Normal-fitting terhadap $error_{rot1}$ untuk jenis with-exclusion dan without-exclusion ditampilkan dalam gambar (5.6).

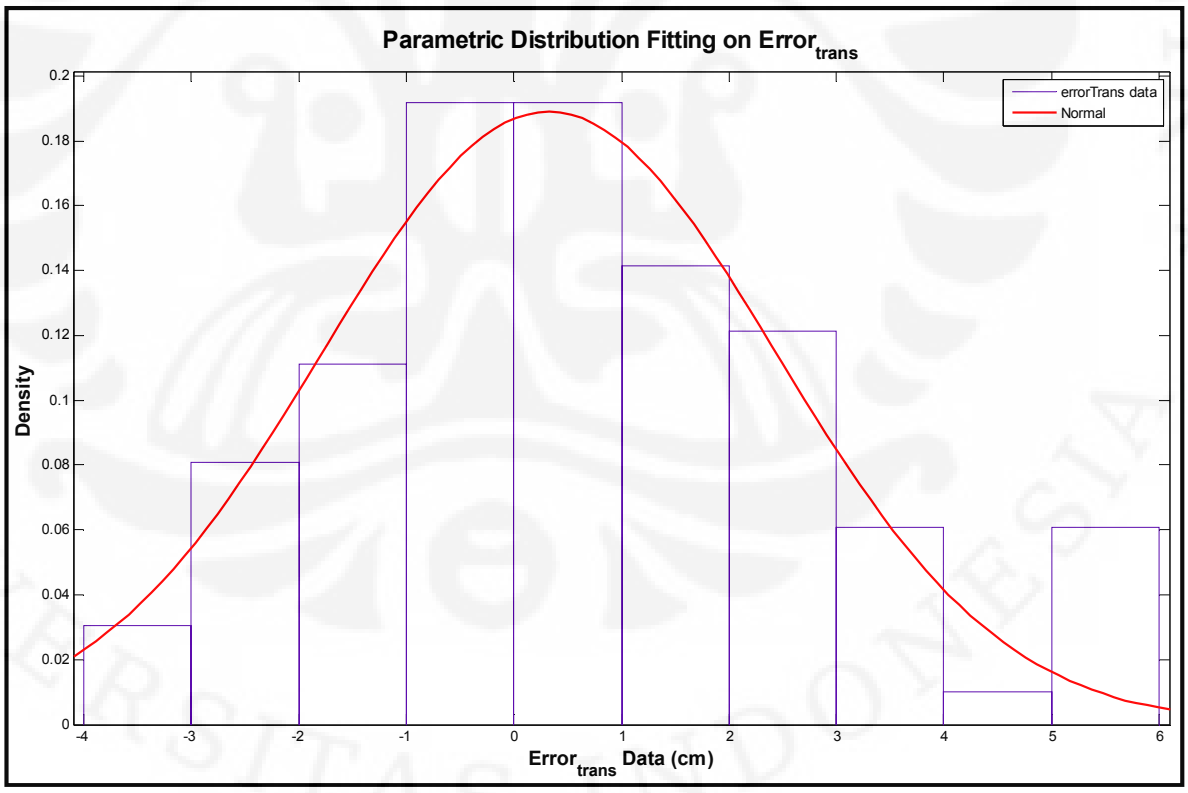
Aktivitas Normal-fitting pada data odometry-error yang lain dilakukan dengan proses yang serupa, ditunjukkan pada gambar (5.7), (5.7), (5.7), dan (5.10). Selain itu, justifikasi exclusion untuk $error_{rot2}$ adalah sama dengan untuk $error_{rot1}$.



Gambar 5.5 Hasil Normal-Fitting untuk Data $error_{rot1}$

Normal-fitting without Exclusion	Normal-fitting with Exclusion
Distribution: Normal Log likelihood: -587.706 Domain: -Inf < y < Inf Mean: 346.988 Variance: 8476.65	Distribution: Normal Log likelihood: -372.778 Domain: -Inf < y < Inf Mean: 349.845 Variance: 2507.89
Parameter Estimate Std. Err. mu 346.988 9.25326 sigma 92.0687 6.59318	Parameter Estimate Std. Err. mu 349.845 5.98556 sigma 50.0788 4.27852
Estimated covariance of parameter estimates: mu sigma mu 85.6228 -3.15893e-014 sigma -3.15893e-014 43.47	Estimated covariance of parameter estimates: mu sigma mu 35.827 3.93695e-014 sigma 3.93695e-014 18.3058

Gambar 5.6 Detil Proses Normal-Fitting pada Data $error_{rot1}$



Gambar 5.7 Hasil Normal-Fitting untuk Data $error_{trans}$

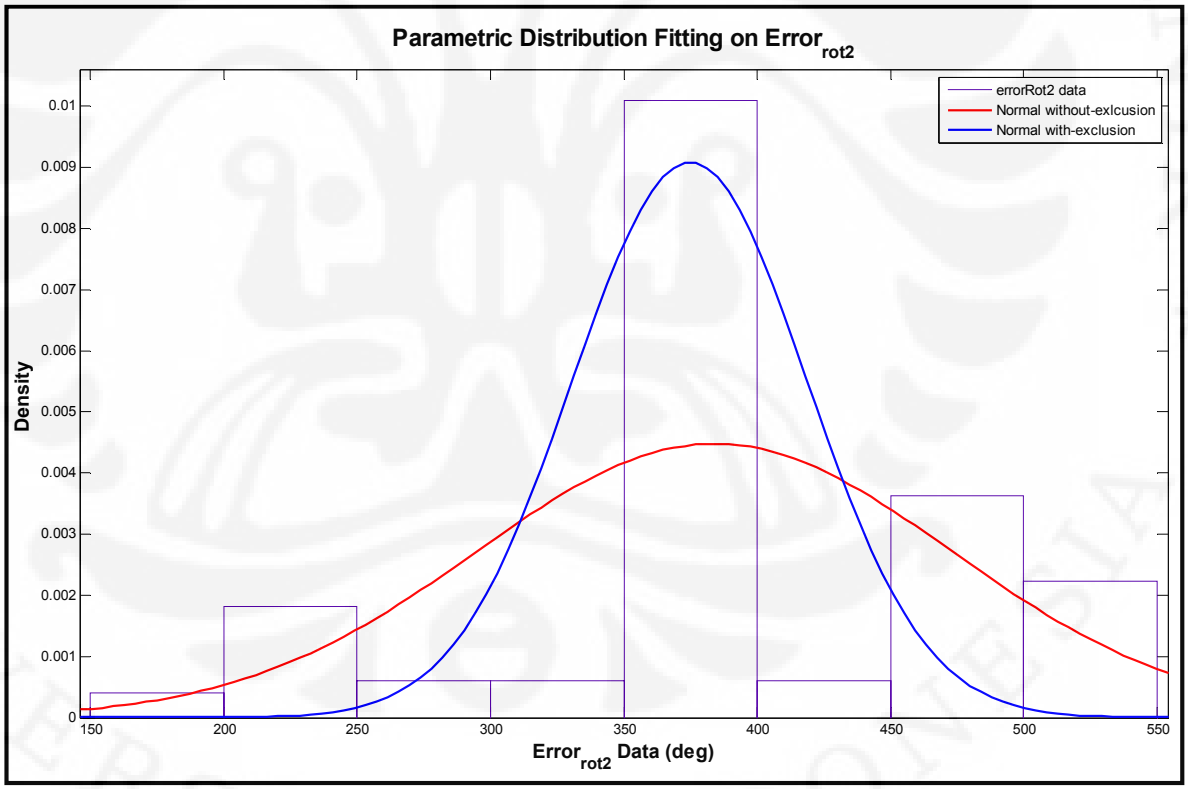
Distribution: Normal
Log likelihood: -213.94
Domain: -Inf < y < Inf
Mean: 0.324073
Variance: 4.45598

Parameter	Estimate	Std. Err.
mu	0.324073	0.212155
sigma	2.11092	0.151166

Estimated covariance of parameter estimates:

	mu	sigma
mu	0.0450099	-5.94527e-018
sigma	-5.94527e-018	0.0228512

Gambar 5.8 Detil Proses Normal-Fitting pada Data $error_{trans}$



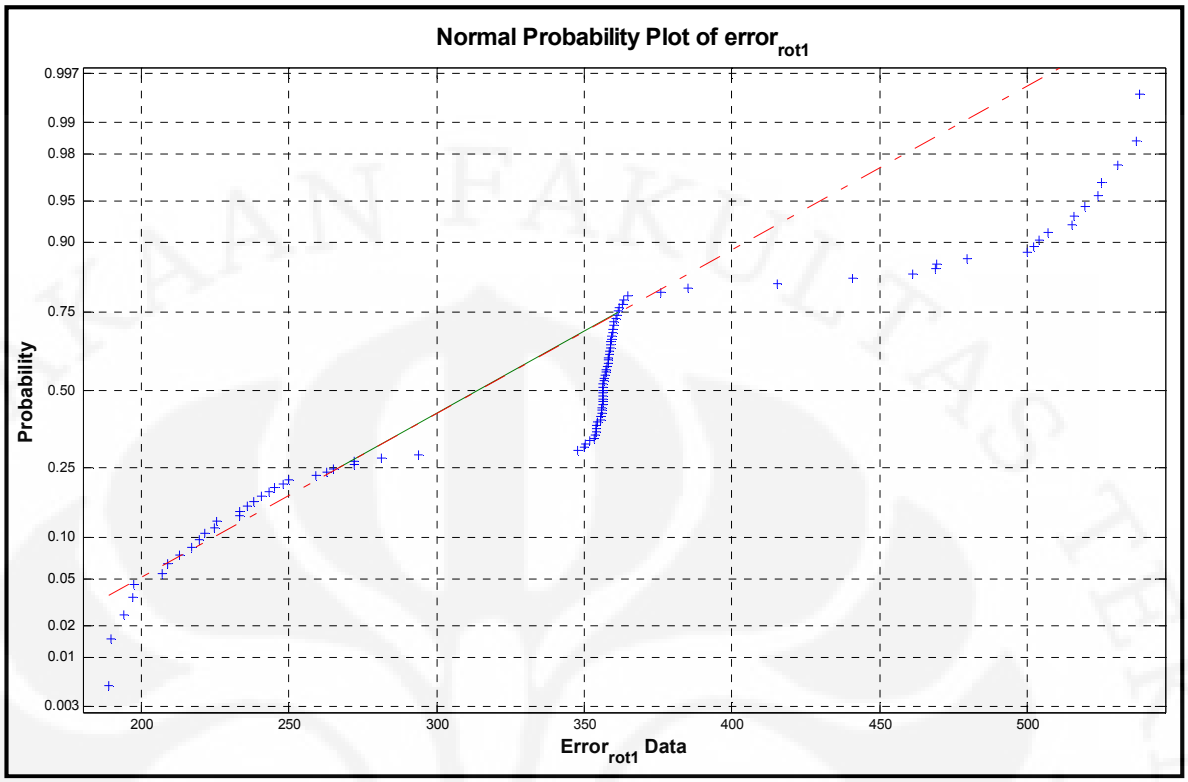
Gambar 5.9 Hasil Normal-Fitting untuk Data $error_{rot2}$

Normal-fitting without Exclusion	Normal-fitting with Exclusion
Distribution: Normal Log likelihood: -584.431 Domain: -Inf < y < Inf Mean: 384.002 Variance: 7933.93	Distribution: Normal Log likelihood: -353.211 Domain: -Inf < y < Inf Mean: 374.712 Variance: 1930.24
Parameter Estimate Std. Err. mu 384.002 8.95213 sigma 89.0726 6.37862	Parameter Estimate Std. Err. mu 374.712 5.32785 sigma 43.9345 3.80961
Estimated covariance of parameter estimates: mu sigma mu 80.1407 -2.92017e-015 sigma -2.92017e-015 40.6868	Estimated covariance of parameter estimates: mu sigma mu 28.3859 -3.8102e-014 sigma -3.8102e-014 14.5131

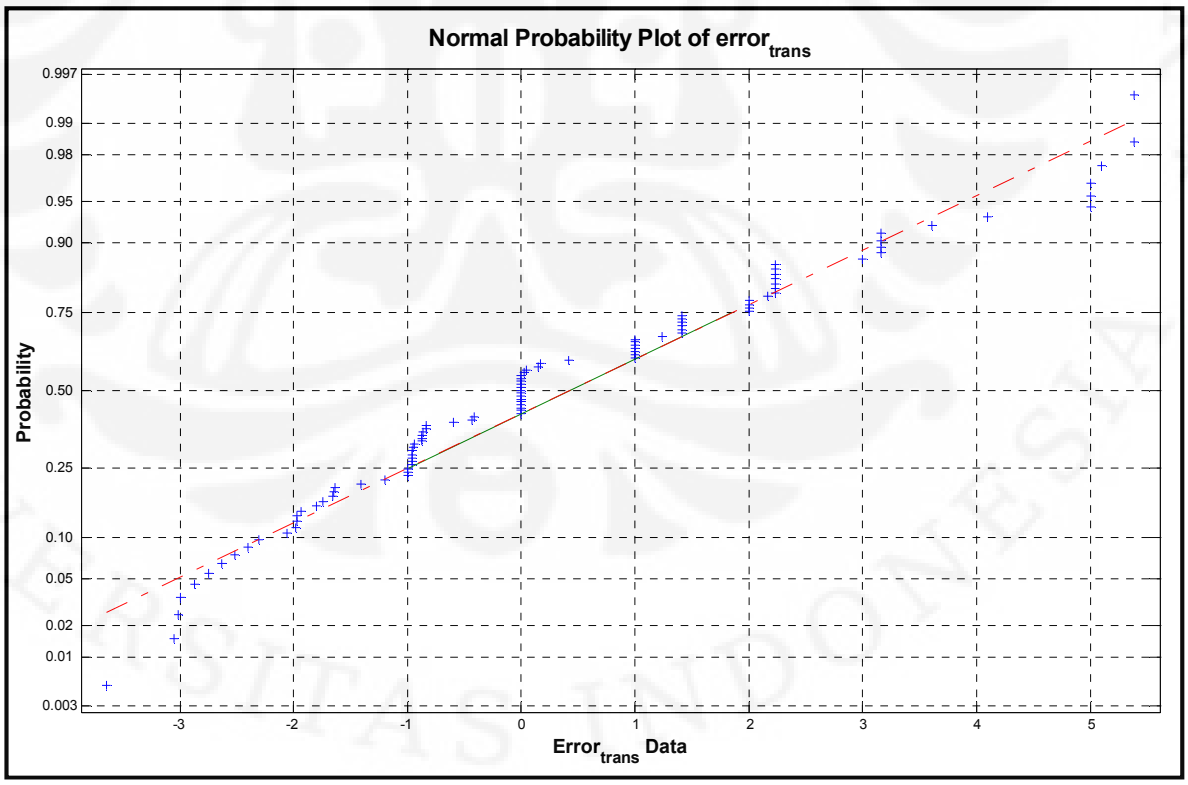
Gambar 5.10 Detil Proses Normal-Fitting pada Data $error_{rot2}$

Lebih lanjut, diperlukan suatu analisis terhadap asumsi yang mendasari dilakukannya proses-proses Normal-fitting di atas dengan tujuan untuk mendapatkan pengetahuan tentang seberapa jauh asumsi tersebut valid serta risiko-risiko yang mungkin timbul dengan diberlakukannya asumsi tersebut. Analisis dilakukan terhadap normal-probability-plot untuk tiap-tiap error-odometry di atas; gambar (5.11), (5.12), dan (5.13).

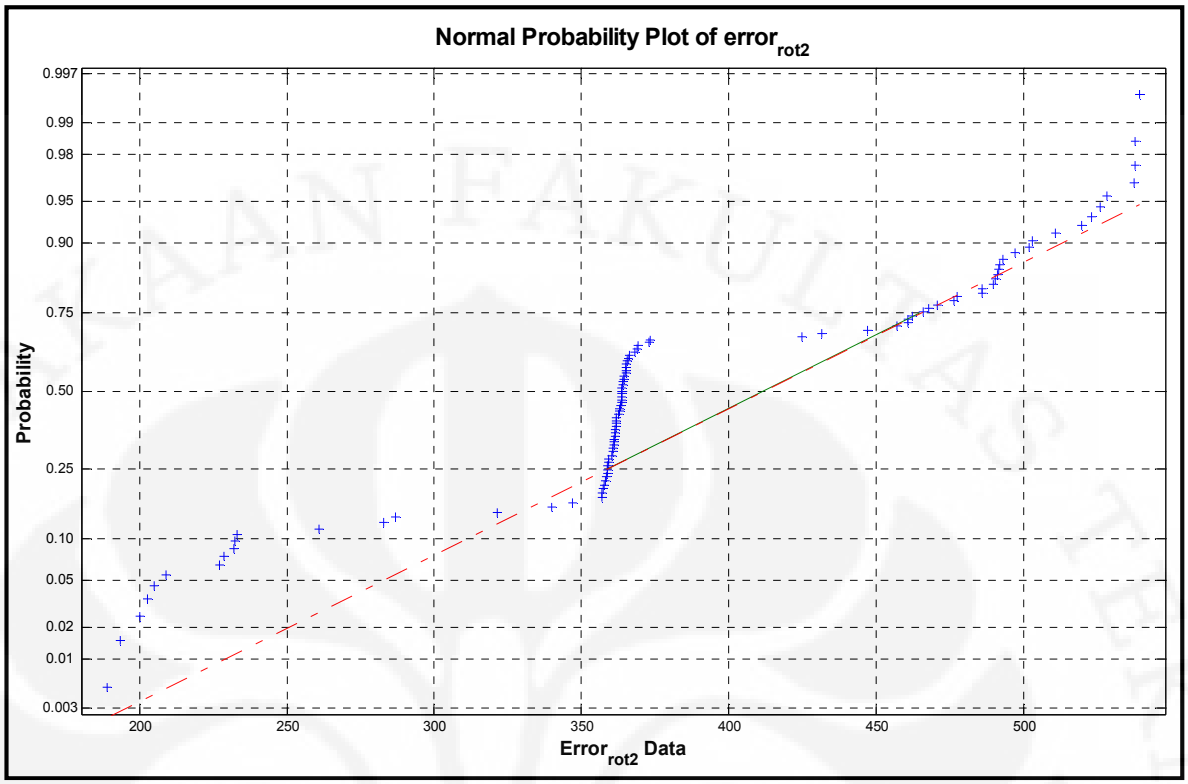
Sebagaimana dijelaskan dalam [65], pada normal-probability-plot, tanda plus memplot empirical probability terhadap nilai data untuk tiap titik pada data. Sementara itu, garis solid menghubungkan percentile ke-25 dan ke-75 dari data dan sebuah garis putus-putus mengekspansinya ke titik akhir data. Nilai-nilai pada sumbu-y (sumbu vertikal) adalah probabilitas dari nol ke satu; tidak dalam skala linear. Jarak antara tanda tick pada sumbu-y bersesuaian dengan jarak antar quantile dari sebuah distribusi normal. Lebih detil, quantiles bersifat dekat satu sama lain di sekitar median yang probabilitasnya 0,5 dan menjauh secara simetris sebanding dengan jaraknya terhadap median. Gambar (5.14) menunjukkan normal-probability-plot untuk data yang benar-benar terdistribusi secara normal dengan mean = 0 dan standard-deviasi = 5. Tampak di sana bahwa semua titik-titik bertanda plus terletak dekat dengan garis lurus.



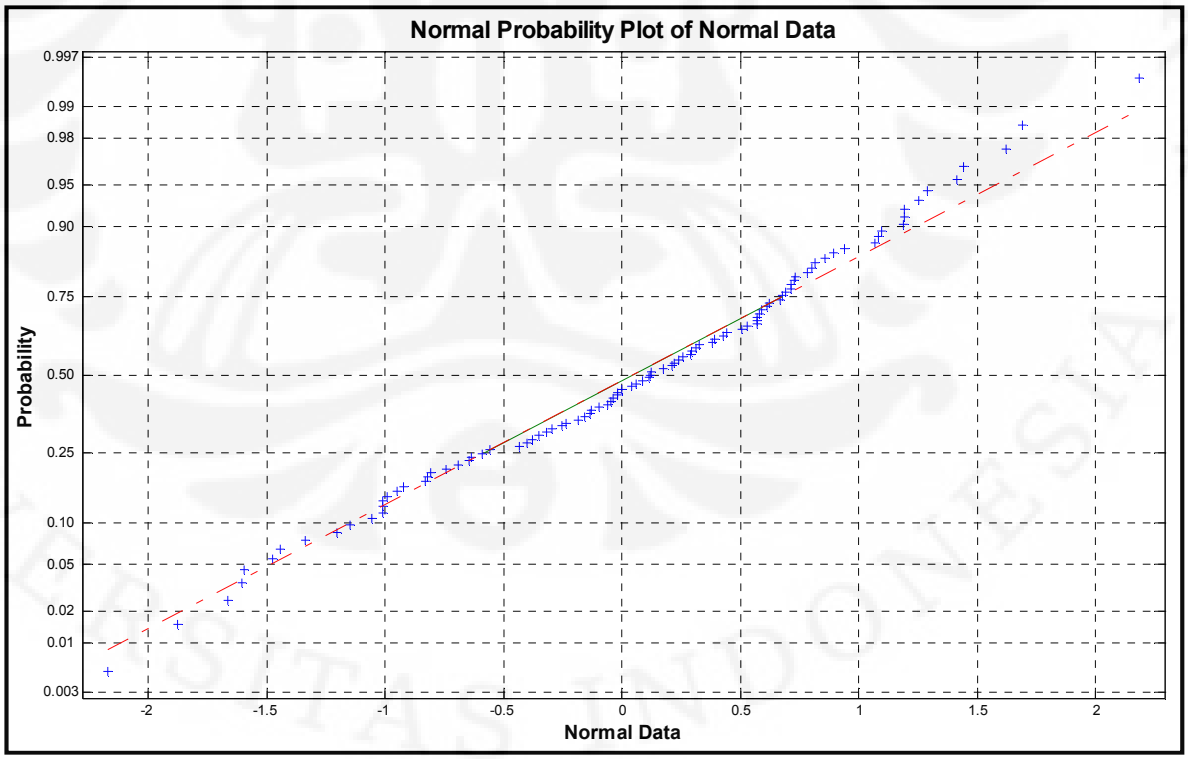
Gambar 5.11 Normal Probability Plot untuk $error_{rot1}$



Gambar 5.12 Normal Probability Plot untuk $error_{trans}$



Gambar 5.13 Normal Probability Plot untuk $error_{rot2}$



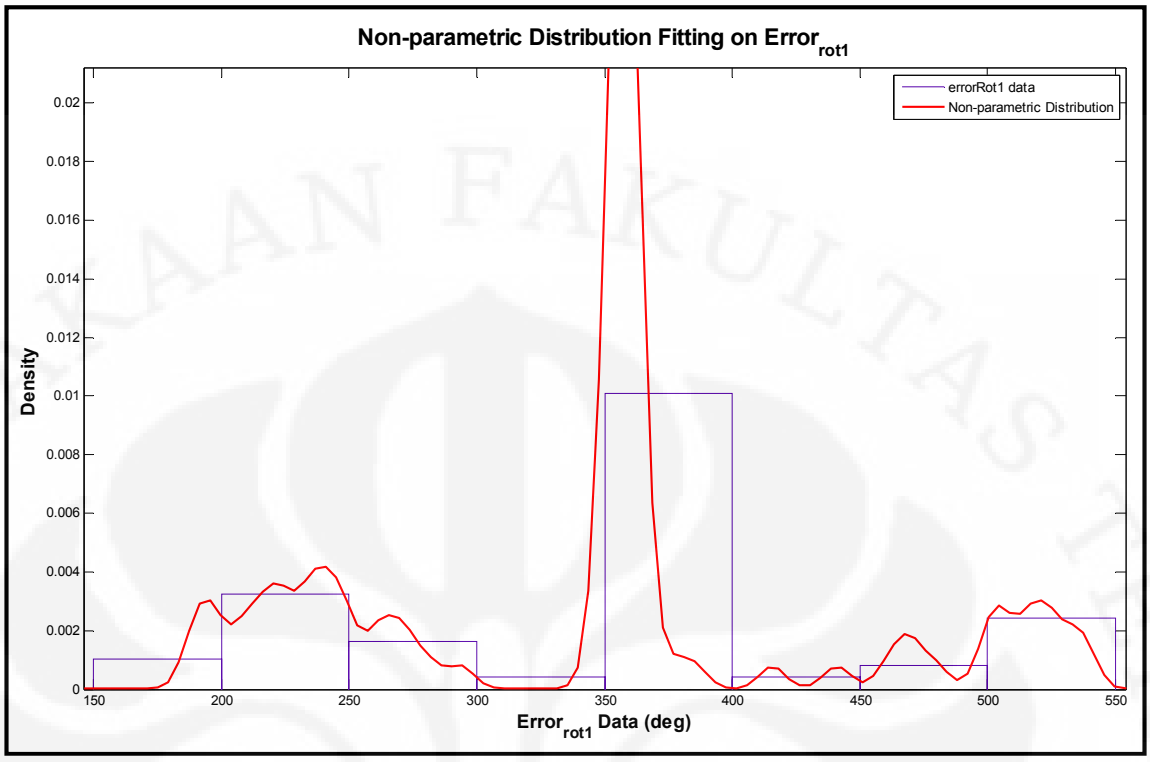
Gambar 5.14 Normal Probability Plot untuk Data Normal

Dengan mengamati letak titik-titik bertanda plus pada normal-probability-plot untuk $error_{rot1}$, gambar (5.11), $error_{trans}$, gambar (5.12), dan $error_{rot2}$, gambar (5.13), sekaligus membandingkannya dengan plot serupa untuk data normal, gambar (5.14), maka dapat disimpulkan bahwa asumsi distribusi normal untuk error-odometry tidak sepenuhnya valid karena terdapat titik-titik bertanda plus yang berada relatif jauh terhadap garis, terlebih untuk data $error_{rot1}$ dan $error_{rot2}$. Dengan pengetahuan ini, risiko yang mungkin timbul akibat ketidakvalidan asumsi telah dapat diidentifikasi.

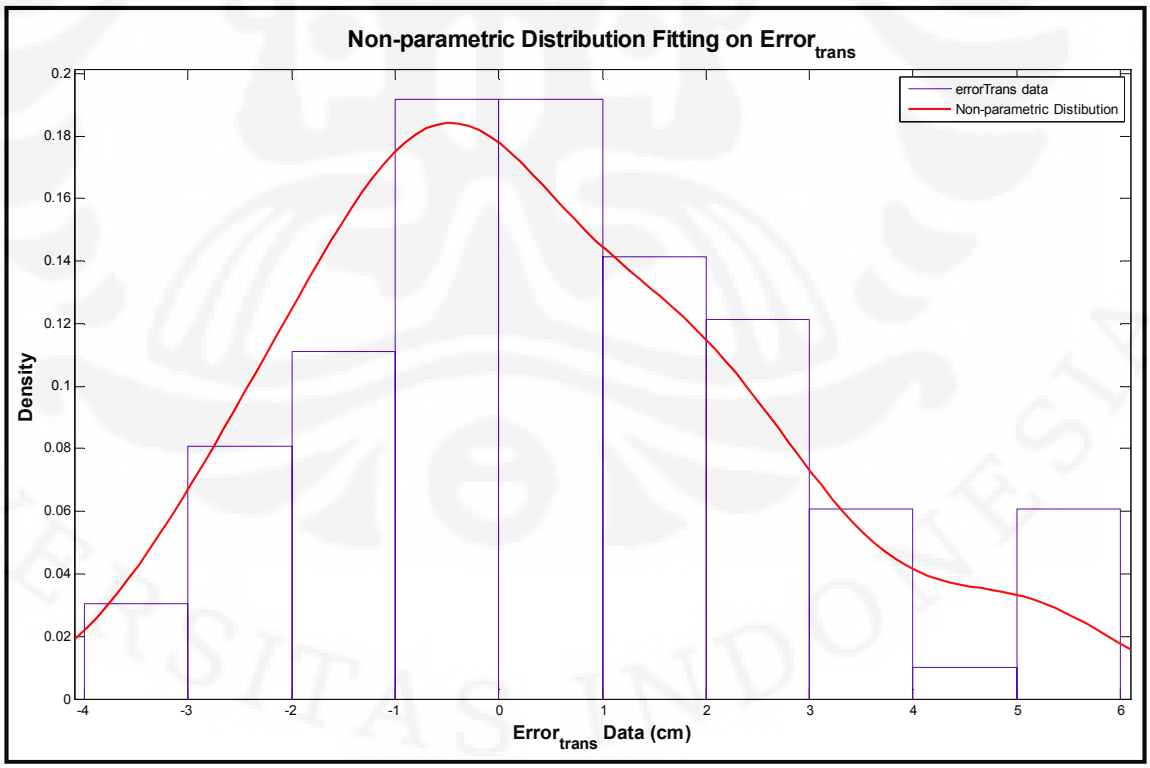
5.4.2 Model Gerakan dengan Non-parametric Probability Distribution

Mengacu pada penjelasan pada subbab sebelumnya, dapat dinyatakan juga bahwa data error-odometry tidak sepenuhnya terdistribusi secara normal, artinya data bersifat multimodal. Oleh karena itu, untuk mengantisipasi hasil yang buruk akibat risiko ketidakvalidan asumsi distribusi normal, dilakukan proses fitting dengan model non-parametric probability distribution. Pada model tersebut, tidak ada mekanisme deterministik untuk menghasilkan data atau bentuk dari suatu probability distribution tertentu sehingga tidak ada pula estimasi parameter.

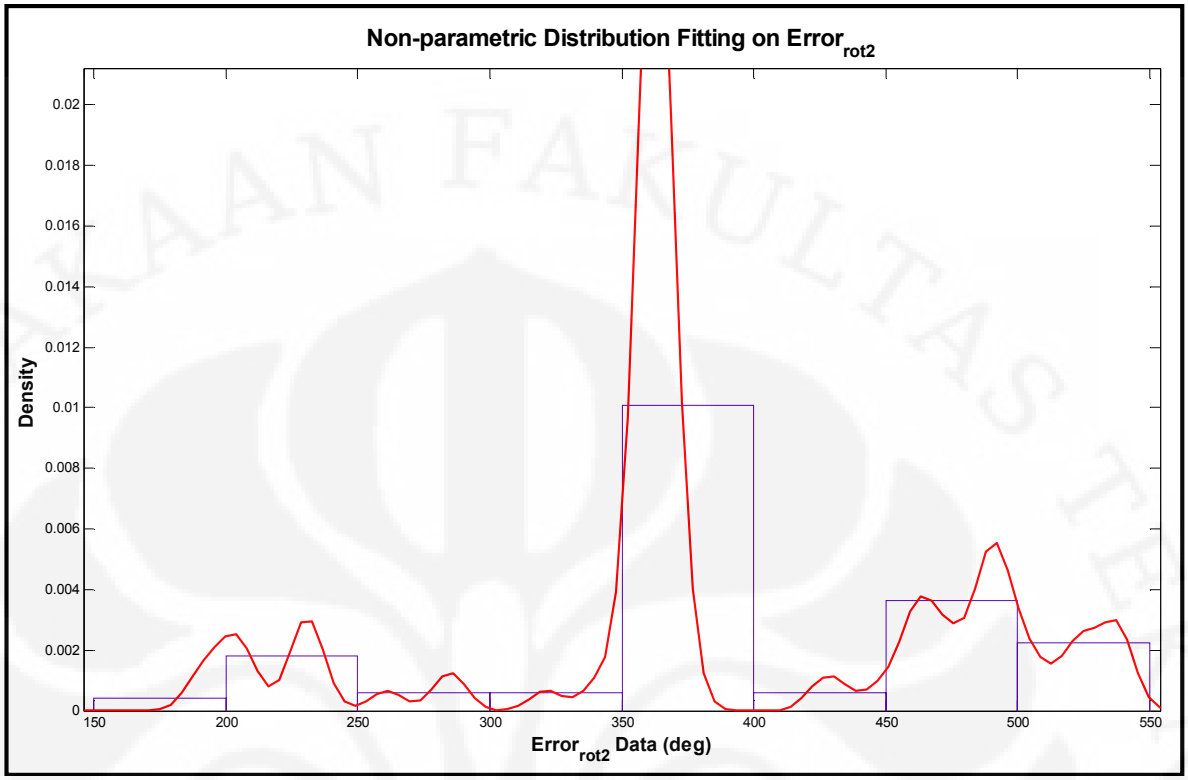
Lebih lanjut, algoritma yang digunakan untuk pemodelan non-parametric bernama ksdensity. Hasil proses fitting non-parametric untuk ketiga data odometry-error ditunjukkan pada gambar (5.15), (5.16), dan (5.17).



Gambar 5.15 Hasil Non-parametric-Fitting untuk Data $error_{rot1}$
(Kernel: normal, Bandwidth: 5.30192, Domain: -Inf < y < Inf)



Gambar 5.16 Hasil Non-parametric-Fitting untuk Data $error_{trans}$
(Kernel: normal, Bandwidth: 0.885918, Domain: -Inf < y < Inf)



Gambar 5.17 Hasil Non-parametric-Fitting untuk Data $error_{rot2}$
(Kernel: normal, Bandwidth: 6.05848, Domain: -Inf < y < Inf)

5.5 Eksperimen Sampling

Mengingat bahwa tujuan akhir riset ini adalah mengimplemetasikan MCL sebagai algoritma self-localization, maka sesegera mungkin perlu ditunjukkan kinerja model gerakan probabilistik sebagai pemrediksi pose pada tahap sampling MCL. Terlebih lagi, ada tiga pilihan model gerakan probabilistik, yaitu Normal with-exclusion, Normal without-exclusion, dan non-parametric, sehingga menuntut diputuskan mana yang teroptimal.

5.5.1 Algoritma Sampling

Karena MCL termasuk algoritma Particle Filter, maka kerja pada tahap sampling adalah memprediksi tiap sampel (particles) berdasarkan informasi kontrol dan pose particle tersebut satu waktu sebelumnya. Di sini, particle adalah representasi

robot tentang pose-nya; tiap particle mempunyai nilai x, y, dan θ. Proses sampling tersebut dapat dinyatakan dengan persamaan (5.13).

$$\text{sample pose}_t \sim p(pose_t | u_t, pose_{t-1}) \quad (5.13)$$

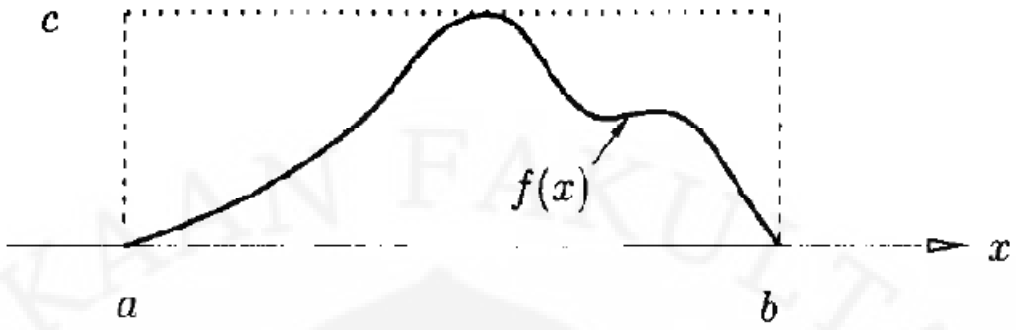
di mana ruas kanan persamaan (5.13) adalah persamaan (5.1) yaitu model gerakan probabilistik.

Aktivitas "sample" dilakukan sebanyak particle yang dilibatkan, n. Seperti disebutkan pada subbab sebelumnya, bahwa inti dari model gerakan probabilistik adalah model dari error-odometry. Oleh karena itu, pada dasarnya, proses sampling berarti mengambil sample yang terdistribusi pada probability-distribution error-odometry. Akhirnya, algoritma sampling pada MCL (untuk satu particle) ditunjukkan pada gambar (5.18).

Pada implementasinya, proses pengambilan sampel dari suatu probability distribution adalah proses pembangkitan random numbers (lebih tepatnya pseudo-random numbers karena diproduksi oleh suatu algoritma yang deterministik) dari probability distribution error-odometry. Dengan demikian, pada riset ini, dibutuhkan algoritma atau metode penghasil random-numbers baik dari probability-distribution parametrik maupun non-parametrik.

```
function [x y theta] = mclSampling(PoseParticlePast,ControlInfo)

// ControlInfo(1) = dRot1; ControlInfo(2) = dTrans; ControlInfo(3) = dRot2;
// PoseParticlesPast(1) = xPast; PoseParticlesPast(2) = yPast; PoseParticlesPast(3) = thetaPast;

errorRot1 = sample from PDF-of-errorRot1;
errorTrans = sample from PDF-of-errorTrans;
errorRot2 = sample from PDF-of-errorRot2;

dRot1Pred = ControlInfo(1) + errorRot1;
dTransPred = ControlInfo(2) + errorTrans;
dRot2Pred = ControlInfo(3) + errorRot2 ;

%COMPUTE PARTICLES------------------------------------------------------------
x = PoseParticlesPast(1) + dTransPred * cos(deg2rad( PoseParticlesPast(3) + dRot1Pred ));
y = PoseParticlesPast(2) + dTransPred * sin(deg2rad( PoseParticlesPast(3) + dRot1Pred ));
theta = PoseParticlesPast(3) +  dRot1Pred + dRot2Pred;theta = rem(theta,360);
```

Gambar 5.18 Algoritma sampling pada MCL



Gambar 5.19 Ilustrasi Acceptance-rejection Method [57]

Lebih lanjut, metode untuk memproduksi pseudo-random numbers biasanya dimulai dengan uniform random numbers, sedangkan random numbers dari distribusi yang lain diproduksi dengan menggunakan metode-metode berikut: directs-method, inversion-method, dan acceptance-rejection-method.

Pada prakteknya, cukup mudah untuk memproduksi random numbers dari parametric probability distribution karena umumnya telah ada fungsi yang disediakan oleh software aplikasi statistik, seperti Statistics-Toolbox MATLAB, misal fungsi "normrnd(mu,sigma)" untuk distribusi Normal. Akan tetapi, lain halnya dengan model non-parametric, perlu dibangun sendiri metode penghasil random numbers. Akhirnya, diputuskan menggunakan acceptance-rejection-method untuk memproduksi random number dari model gerakan probabilistik non-parametrik; pemaparannya adalah sebagai berikut.

Merujuk pada [57], untuk menjelaskan ide dari acceptance-rejection-method dimunculkan ilustrasi yang ada pada gambar (5.19). Sebagaimana terlihat pada gambar (5.19), semisal f merupakan suatu PDF darinya kita ingin mengambil sampel dan ia dibatasi oleh interval terhingga [a,b] dan bernilai nol di luar interval tersebut, serta $c = sup\{f(x) : x \epsilon [a,b]\}$. Dengan demikian, produksi sebuah

random number Z dari distribusi f dapat dilakukan dengan langkah-langkah acceptance-rejection berikut:

a. Menentukan X ~ U(a,b)
b. Menentukan Y ~ U(0,c) secara independent dari X
c. Jika $Y \leq f(X)$, maka Z = X. Jika tidak, kembali ke langkah (a)

Dapat dipahami bahwa tiap vektor yang ditentukan, (X,Y), adalah terdistribusi secara seragam pada segiempat [a.b] x [0,c]. Oleh karena itu, pasangan (X,Y) yang diterima adalah terdistribusi secara seragam di bawah plot f. Dengan demikian, jelaslah bahwa nilai X yang diterima mempunyai PDF f.

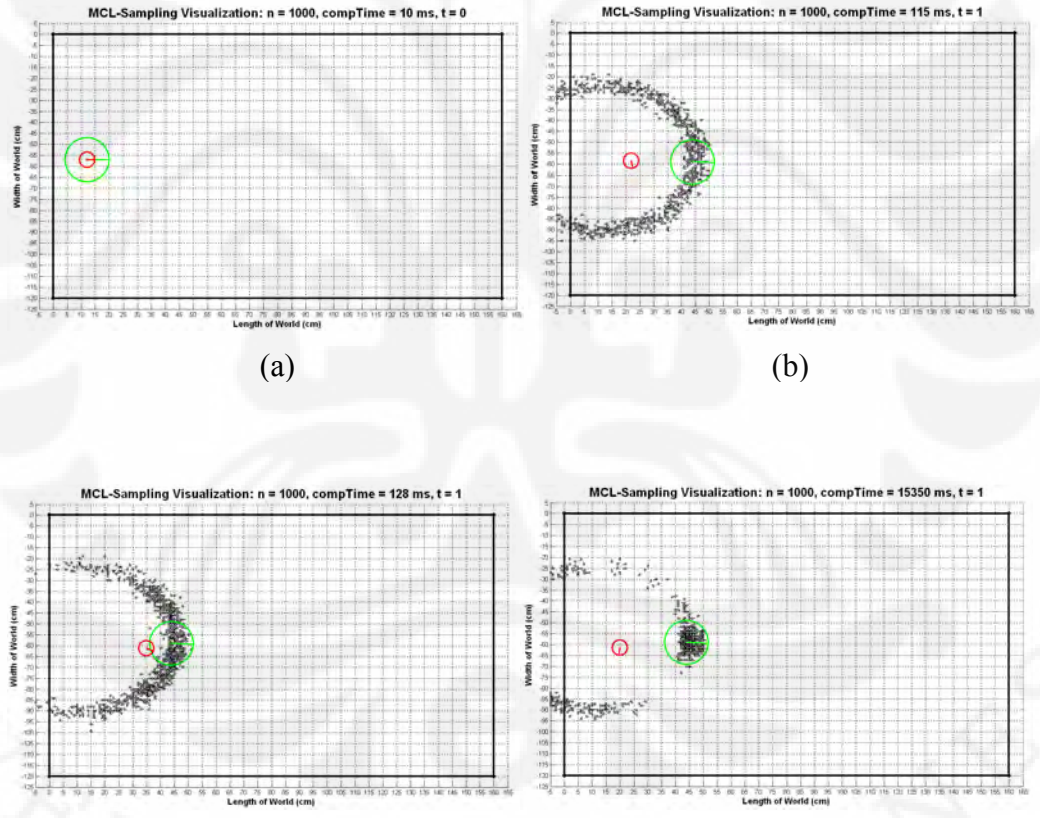
Jadi, untuk model gerakan probabilistik yang parametrik, produksi random numbers dilakukan dengan mengeksekusi fungsi yang biasanya telah tersedia (common) pada suatu library software seperti normrnd(mu,sigma); di mana mu adalah mean dan sigma adalah standard-deviasi dari error-odometry. Sementara itu, untuk model gerakan probabilistik yang non-parametrik produksi random numbers dilakukan dengan mengeksekusi fungsi buatan sendiri yaitu accRejRnd(f,a,b,m,n); di mana f adalah fungsi polynom dari PDF f, a adalah nilai minimal dari error-odometry, b adalah nilai maksimal dari error-odometry, dan m, n merupakan ukuran matrik random numbers yag diproduksi, dalam kesempatan ini m = n =1.

5.5.2 Hasil Eksperimen Sampling

Ekperimen ini bertujuan menguji kinerja model gerakan probabilistik dalam tahap sampling pada MCL. Ada dua parameter pengukur yaitu akurasi prediksi pose dan kecepatan komputasi. Yang pertama dihitung dari selisih antara pose absolut (berasal dari penjejak pose absolut) dengan pose prediksi rata-rata. Perlu disampaikan bahwa waktu komputasi sampling didominasi oleh pembangkitan random number. Akhirnya diputuskan bahwa model gerakan probabilistik yang

teroptimal adalah yang prediksi posenya paling menghampiri pose absolut dan waktu komputasinya tercepat.

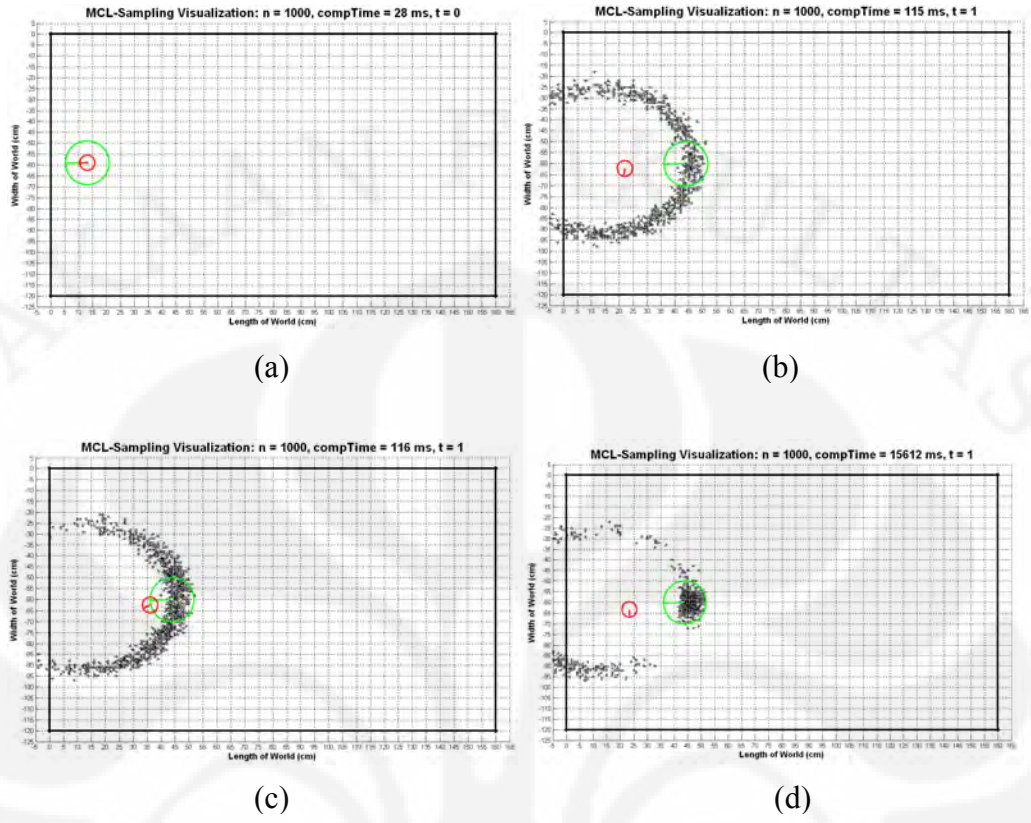
Lebih lanjut, sampling dilakukan pada satu waktu yaitu t = 1 dengan n = 1000 particles. Terdapat empat jenis gerakan robot yang diberlakukan pada t = [0,1), yaitu: forward , backward , rot-CW, dan rot-CCW. Pada tiap jenis gerakan dilakukan tiga kali sampling dengan model gerakan probabilistik yang berbeda. Visualisasi sampling pada eksperimen tersebut ditampilkan pada gambar (5.20), (5.21), (5.22), dan (5.23).



(a) (b)

(c) (d)

Gambar 5.20 Hasil Sampling untuk Jenis Gerakan Forward,
(a) t = 0, (b) t = 1, Normal without-exclusion
(c) t = 1, Normal with-exclusion, (d) t = 1, Non-parametric

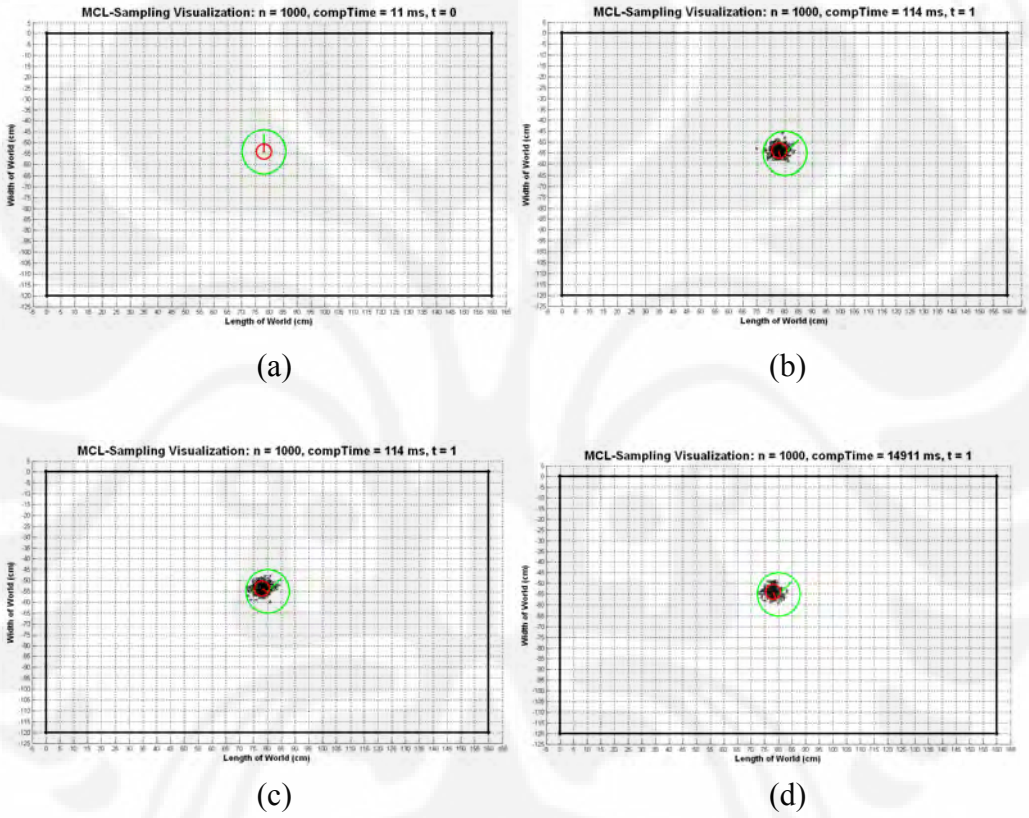


Gambar 5.21 Hasil Sampling untuk Jenis Gerakan Backward,
(a) t = 0, (b) t = 1, Normal without-exclusion
(c) t = 1, Normal with-exclusion, (d) t = 1, Non-parametric

5.5.3 Analisis Hasil Eksperimen Sampling

Untuk parameter tentang akurasi prediksi pose, berdasarkan gambar (5.24), (5.25), dan (5.26), dapat disimpulkan bahwa model parametrik yaitu Normal with-exclusion adalah yang paling akurat. Sementara itu, yang paling buruk akurasinya adalah model non-parametric; yang semula diharapkan akan lebih baik. Akan tetapi, jika mengobservasi gambar (5.20d) dan (5.21d), tampak bahwa hasil sampling dengan model non-parametrik menghasilkan particles yang menggerombol di sekitar pose-absolut; berbeda secara signifikan dengan gambar (5.20b,c) dan (5.21b,c). Selain itu, teknik perhitungan error pose adalah mencari nilai rata-rata yang tidak sepenuhnya sesuai untuk data yang membentuk cluster.

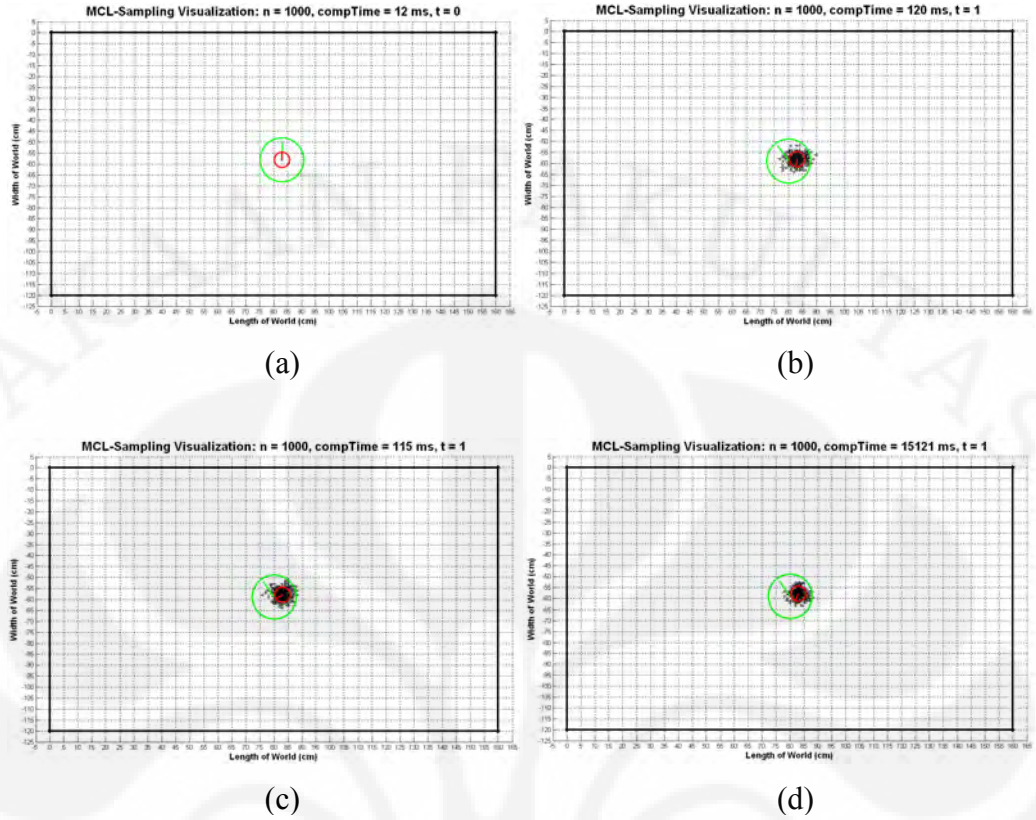
Kedua hal tersebut menimbulkan prediksi bahwa model itu akan menunjukkan kinerja yang superior dalam kinerja MCL secara keseluruhan dengan variasi jumlah particle. Selanjutnya tentang waktu komputasi, berdasarkan gambar (5.27), model non-parametrik jauh lebih besar daripada kedua model yang lain, di mana untuk yang lain itu nilainya relatif sama;



(a) (b)

(c) (d)

Gambar 5.22 Hasil Sampling untuk Jenis Gerakan Rot-CW,
(a) t = 0, (b) t = 1, Normal without-exclusion
(c) t = 1, Normal with-exclusion, (d) t = 1, Non-parametric

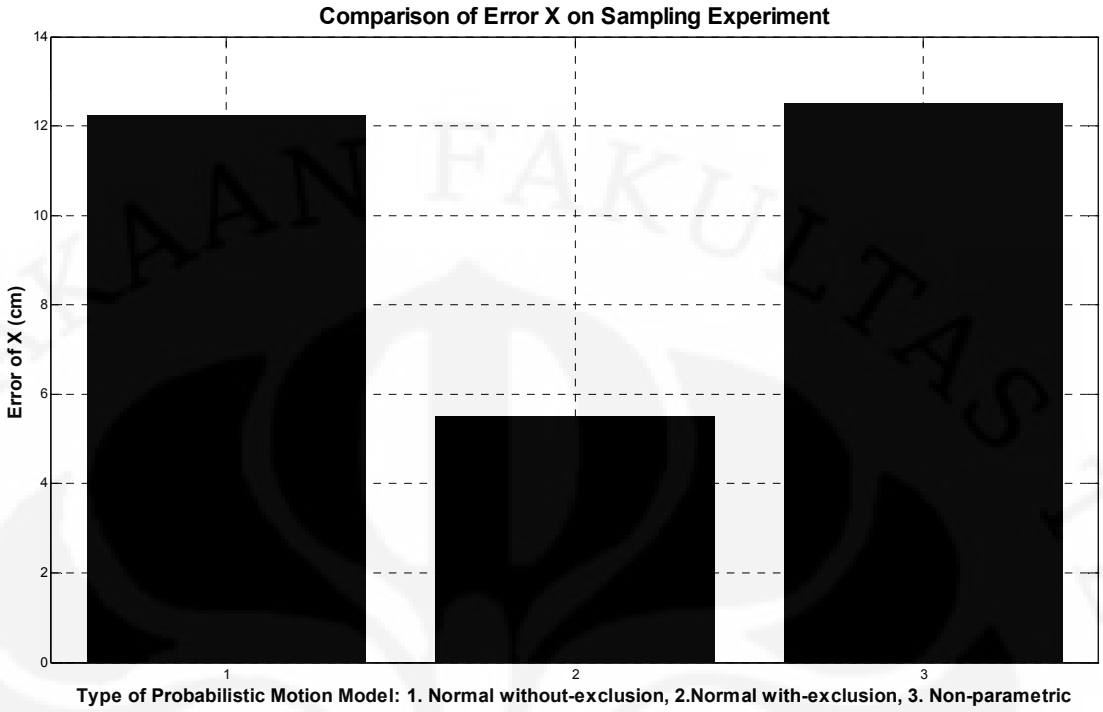


(a) (b)

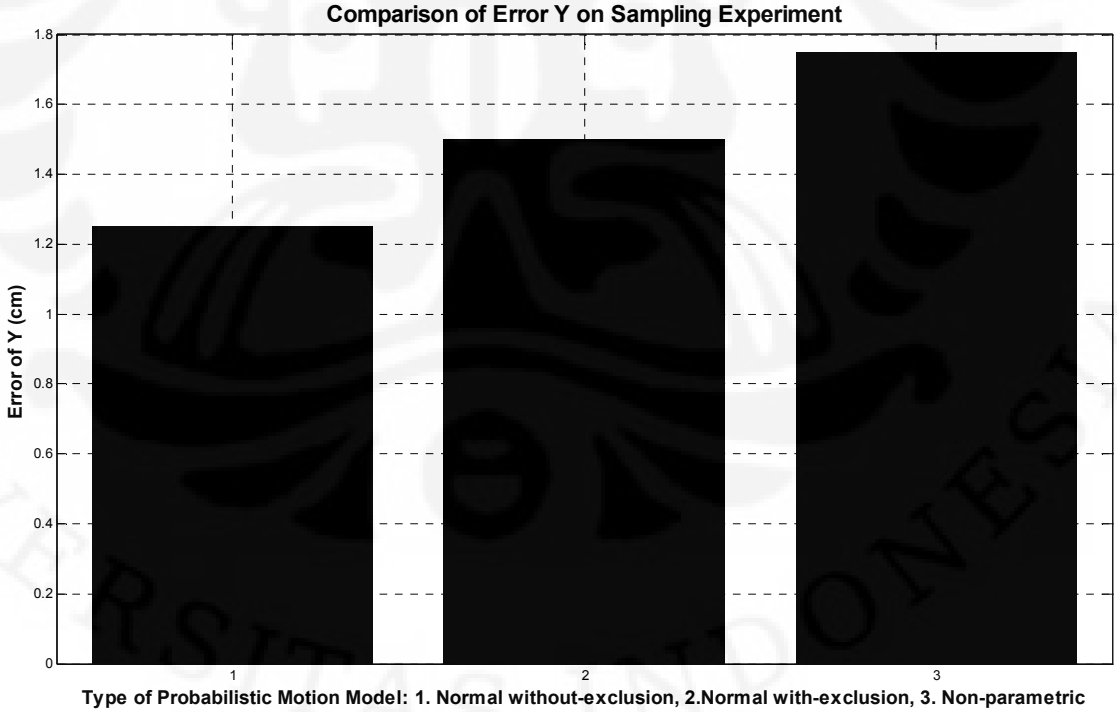
(c) (d)

Gambar 5.23 Hasil Sampling untuk Jenis Gerakan Rot-CCW,
((a) t = 0, (b) t = 1, Normal without-exclusion
(c) t = 1, Normal with-exclusion, (d) t = 1, Non-parametric

Akhirnya, berdasarkan analisis terhadap parameter akurasi dan waktu komputasi sampling, model gerakan probabilistik Normal with-exclusion dinyatakan sebagai yang teroptimal. Akan tetapi, model non-parametrik tetap dipertahankan sebagai alternatif karena adanya prediksi tentang kinerjanya yang membaik pada penerapan MCL secara keseluruhan dengan variasi jumlah particle.

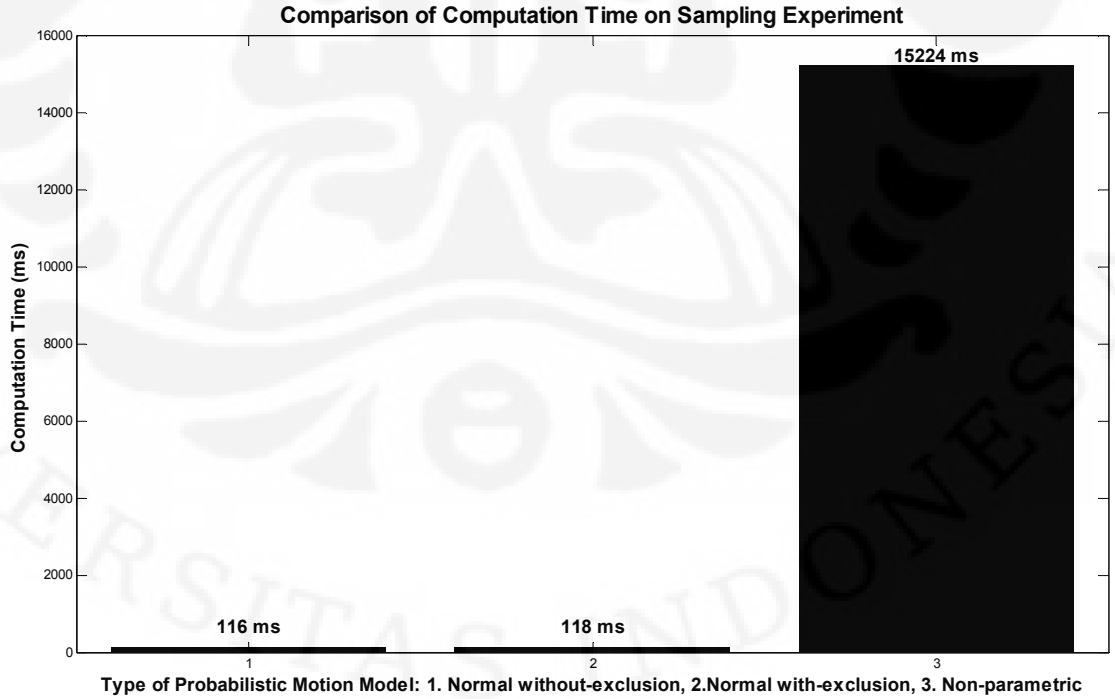


Gambar 5.24 Grafik Perbandingan Error X pada Eksperimen Sampling:



Gambar 5.25 Grafik Perbandingan Error Y pada Sampling

Gambar 5.26 Grafik Perbandingan Error Theta pada Eksperimen Sampling



Gambar 5.27 Grafik Perbandingan Waktu Komputasi pada Eksperimen Sampling

BAB 6

MODEL PERSEPSI PROBABILISTIK TMR ALFATHVRSS

6.1 Pendahuluan

Selain gerakan robot, sumber ketidakpastian dalam robot adalah persepsi robot, yaitu pengetahuan robot tentang keadaan dirinya sendiri dan lingkungannya. Sebuah robot mendapatkan persepsi dengan perantara sensor-sensornya, baik exteroceptive maupun proprioceptive sensor.

Dalam hal lokalisasi dengan Plain-MCL, persepsi mempunyai peranan yaitu memperbaiki hasil prediksi pose, tepatnya dalam tahap weighting (importance). Lebih lanjut, tahap tersebut hanya mengandalkan hasil pembacaan exteroceptive sensor saja karena persepsi dengan proprioceptive sensor (yaitu wheel-encoder) telah dimanfaatkan dalam tahap sampling. Sesungguhnya, tersirat bahwa proses lokalisasi merupakan proses sensor-fusion.

Lebih lanjut, pada prinsipnya, pemodelan persepsi secara probabilistik adalah mencari probability distribution dari suatu conditional density berikut:

$$p(z_t \mid pose_t, m) \tag{6.1}$$

di mana z_t adalah hasil pembacaan suatu sensor pada saat t dan m adalah peta lingkungan robot.

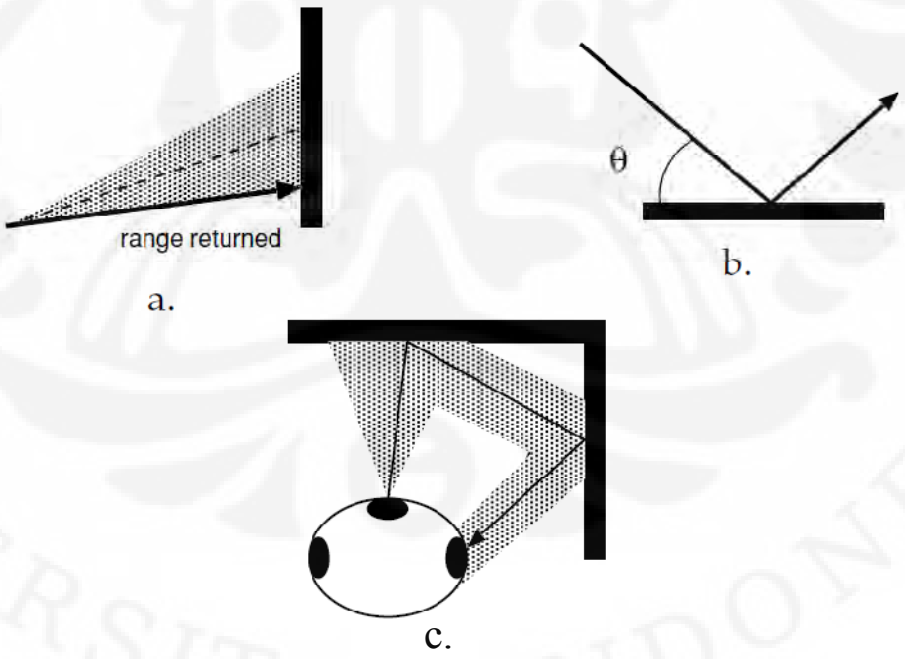
TMR Alfathvrss, sebagaimana dipaparkan dalam Bab 2, mempunyai dua jenis exteroceptive sensor yaitu sensor jarak berupa sonar SRF08 (empat buah) dan sensor arah berupa kompas digital CMPS03 (satu buah). Dengan demikian, hanya dua jenis sensor itu yang perlu dimodelkan untuk mendukung penerapan MCL.

Pada bab ini mula-mula dibahas mengenai pemodelan sonar dan kompas digital secara probabilistik yaitu pada subbab 6.2 dan 6.3. Kemudian, dilanjutkan dengan penjelasan mengenai peranan model persepsi dalam tahap weighting pada MCL yaitu di subbab 6.4.

6.2 Model Probabilistik Sonar-SRF08

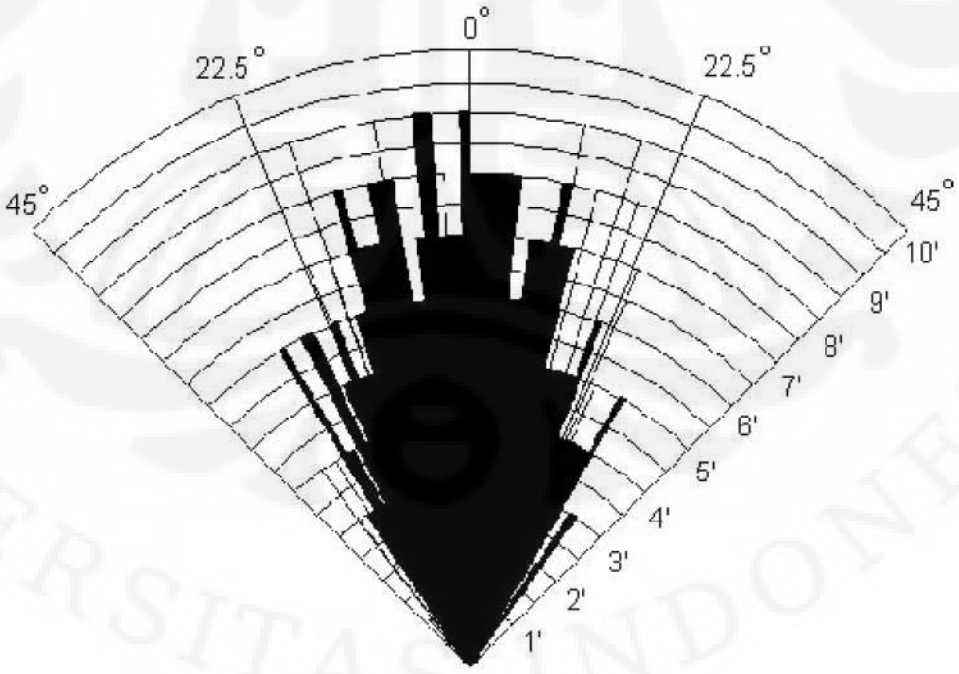
6.2.1 Masalah Dasar Sonar

Masalah dasar sonar didefinisikan sebagai fenomena bawaan (inheritance) sonar yang terjadi saat ia bekerja. Merujuk pada [52], ada tiga masalah dasar tersebut yaitu foreshortening, specular reflection, dan cross-talk. Ilustrasi tentang ketiganya ditampilkan pada gambar (6.1).



Gambar 6.1 Masalah Dasar Sonar [52]

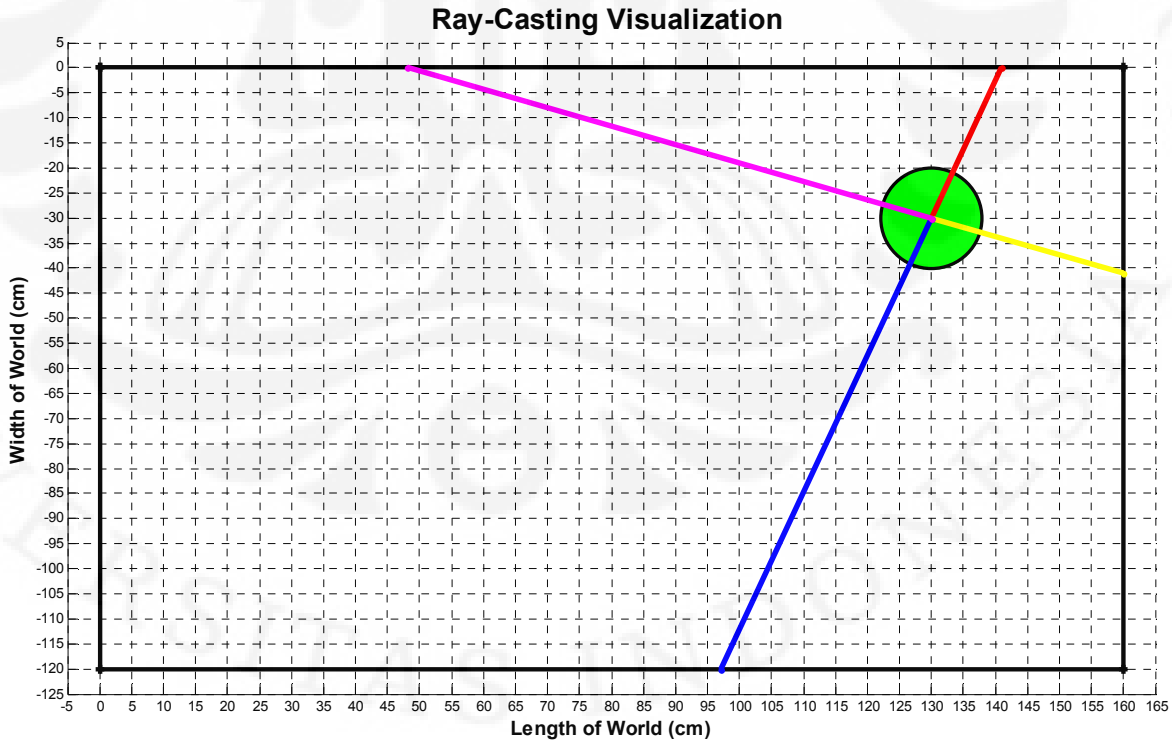
Gambar (6.1a) menunjukkan gejala foreshortening yaitu hasil pembacaan sonar yang lebih pendek dari yang sebenarnya. Hal ini terjadi karena karakteristik sonar yaitu memiliki Field of View (FOV) atau beam pattern yang berbentuk kerucut, sebagai contoh adalah kepunyaan Sonar SRF08, gambar (6.2). Selain itu, terjadi juga masalah specular reflection (gambar 6.1b). Hal tersebut terjadi pada saat sudut datang gelombang akustik sonar terlalu besar sehingga pantulannya tidak kembali ke receiver pada sonar, atau jikapun kembali biasanya telah memantul pada benda-benda lain yang menyebabkan waktu tempuhnya jauh lebih besar daripada yang seharusnya. Masalah dasar ketiga dinamakan cross-talk (gambar 6.1c). Pada umumnya sebuah mobile robot dipasangi beberapa sonar yang spesifikasi individunya identik. Oleh karena itu, sebuah sonar tidak dapat mengenali gelombang akustik pantulan: mana yang berasal dari dirinya sendiri dan mana yang tidak. Akhirnya, ada kemungkinan gelombang pantulan dari pancaran gelombang akustik suatu sonar terbaca oleh tetangganya jika sonar-sonar itu sedang bekerja secara simultan.



Gambar 6.2 Beam Pattern Sonar SRF08 [66]

6.2.2 Metode Ray-Casting

Metode ray-casting (disebut juga ray-tracing) adalah suatu teknik untuk mendapatkan pembacaan sonar sebagai sensor jarak yang ideal (tingkat akurasi 100%) yaitu jika gelombang akustik sonar bersifat seperti cahaya (ray): berupa garis lurus dan sukar memantul, dalam metode ini malah tidak memantul sama sekali. Hasil pembacaan sonar dengan metode ray-casting adalah panjang garis yang terbentuk. Penerapan metode ini pada sonar-sonar TMR Alfathvrss ditunjukkan pada gambar (6.3). Ciri khas metode ini adalah tidak timbulnya tiga masalah dasar sonar yang dijelaskan pada subbab sebelum ini karena sonar diasumsikan sebagai sensor jarak yang ideal. Namun demikian, metode ray casting yang dipakai dalam riset ini tetap memiliki resolusi yang sama dengan sonar SRF08 yaitu 1 cm. Akibatnya, terjadi sedikit perbedaan konfigurasi atau tata letak sonar-sonar antara yang aktual dengan yang tampak pada gambar (6.3); jarak tiap sonar pada gambar itu tidak tepat 90˚. Begitu pula dengan spesifikasi mengenai jarak baca minimum dan maksimum, keduanya diset sama dengan sonar SRF08 yang digunakan.



Gambar 6.3 Visualisasi Ray-Casting pada TMR Alfathvrss

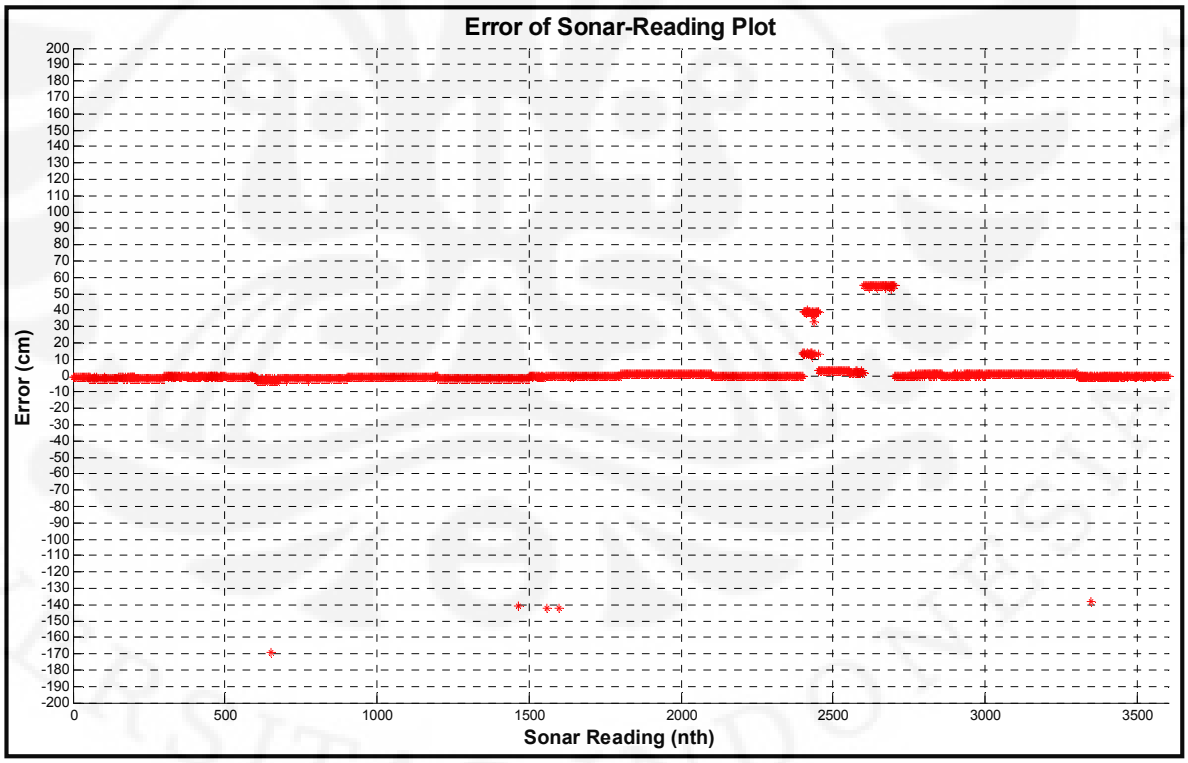
6.2.3 Hasil Eksperimen Pembacaan Sonar

Eksperimen ini bertujuan untuk melihat variasi error dari pembacaan sonar. Rumusan error tersebut ada pada persamaan (6.2).

$$err_{sonar} = z^{sonar,abs} - z^{sonar} \quad (6.2)$$

di mana $z^{sonar,abs}$ adalah hasil pembacaan sonar dengan metode ray-casting dan z^{sonar} adalah hasil asli pembacaan sonar.

Hasil eksperimen berupa data error sonar dari 3600 pembacaan sonar ditampilkan pada gambar 6.4 berikut.



Gambar 6.4 Hasil Eksperimen Pembacaan Sonar

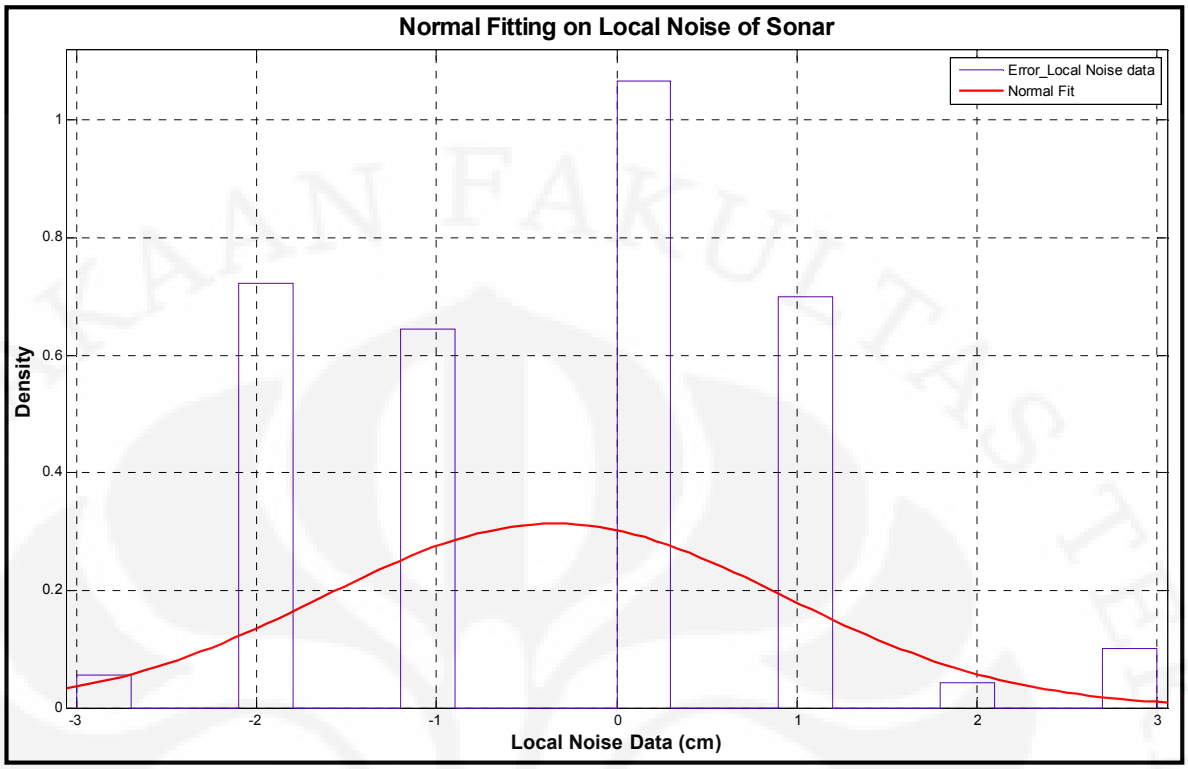
6.2.4 Tipe-tipe Error Sonar-Reading dan Pemodelannya

Sebagaimana dijelaskan dalam [3], ada empat tipe error hasil pembacaan sonar (sonar reading/measurement), yaitu local noise, unexpected-object, failures, dan random measurements. Tipe error unexpected-object diabaikan dalam riset ini dengan alasan sebagai berikut. Error jenis ini diakibatkan oleh lingkungan kerja robot yang bersifat dinamis. Ambil contoh, ada benda yang tiba-tiba berada pada FOV suatu sonar dan benda itu tidak ada pada peta robot, maka pembacaan sonar akan bersifat salah (erroneous); biasanya lebih pendek daripada jarak yang aktual. Akan tetapi, lingkungan robot dalam riset ini bersifat statis; letak objek-objek dalam lingkungan nyata telah diketahui robot melalui peta lingkungan yang tidak berubah-ubah. Dengan demikian, pilihan untuk mengabaikan jenis error ini bersifat logis.

6.2.4.1 Local Noise

Local noise adalah error pembacaan sonar di sekitar jarak sesungguhnya (jarak yang dihitung dengan metode ray-casting). Untuk mendapatkan model local noise yaitu mendapatkan nilai persamaan (6.1), maka dilakukan proses fitting pada data hasil eksperimen pembacaan sonar (subbab 6.2.3).

Berbekal pengalaman terhadap penggunaan sonar SRF08 selama bertahun-tahun, diputuskan untuk membatasi local noise pada kisaran kurang dari atau sama dengan 5 cm. Selanjutnya, data error local noise tersebut dimodelkan dengan probability distribution Normal; gambar (6.5) dan (6.6).



(a)

Distribution: Normal
Log likelihood: -5716.68
Domain: -Inf < y < Inf
Mean: -0.348925
Variance: 1.62277

Parameter	Estimate	Std. Err.
mu	-0.348925	0.0217131
sigma	1.27388	0.0153569

Estimated covariance of parameter estimates:

	mu	sigma
mu	0.00047146	4.94269e-018
sigma	4.94269e-018	0.000235833

(b)

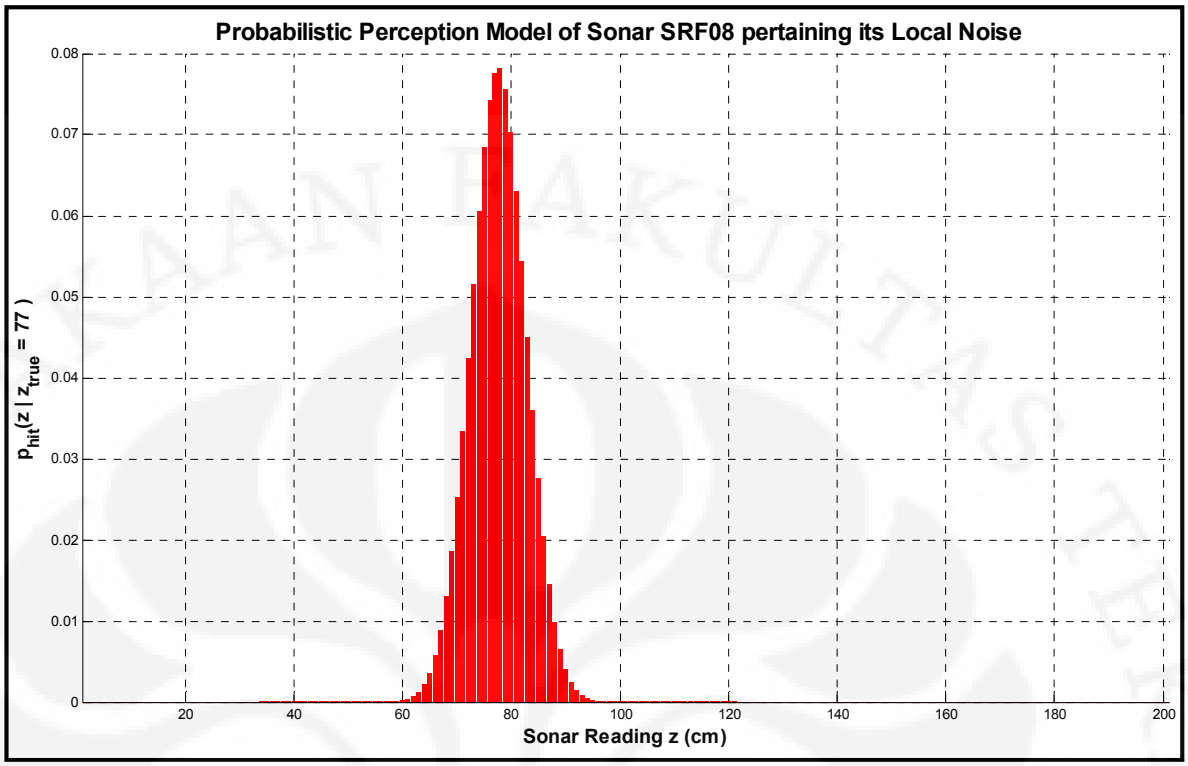
Gambar 6.5 Catatan Normal Fitting pada Local Noise dari Sonar

Akhirnya, model sonar probabilistik untuk local noise dapat dinyatakan sebagai Normal distribution dengan mean = -0.348925 dan standard-deviation = 1.27388. Lebih lanjut, berdasarkan pengalaman penulis tentang local noise ini, sebagaimana digunakan juga dalam menentukan kisaran local noise, maka diperlukan suatu anggapan berlebihan (overestimate) terhadap standard-deviation error tipe ini. Untuk itu, ditetapkan faktor pengali (noise-overestimate factor) sebesar 4 untuknya sehingga standard-deviation = 4 x 1.27388 = 5.0955. Dinyatakan juga bahwa nilai tersebut berlaku seragam dalam kisaran baca sonar dari z_{min} = 0 cm sampai z_{max} = 200 cm yang berlaku pada riset ini. Penulis memprediksi akan terjadi variasi standard-deviation (semakin membesar) dari local noise untuk kisaran baca yang lebih jauh.

Selanjutnya, penerapan pengetahuan tentang local noise pada persamaan (6.1) dapat dituliskan pada persamaan (6.3) di mana peluang untuk persepsi yang melibatkan local noise pada sonar diberi nama p_{hit}.

$$p_{hit}(z_t^{sonar} \mid pose_t, m) = \begin{cases} N\left(\mu=[z_t^{sonar,abs} - 0.348925],\ \sigma=5.0955\right); if\ 0 \le z_t^{sonar} \le z_{max} \\ 0 \qquad ; otherwise \end{cases} \tag{6.3}$$

Untuk mendapatkan visual dari persamaan (6.3), maka dipilih pembacaan sonar pada jarak aktual $z_t^{sonar,abs} = 77\ cm$ yang juga merupakan hasil ray-casting jika dilakukan. Plot untuknya ada pada gambar (6.6).



Gambar 6.6 Visualisasi Model Persepsi Probabilistik Sonar SRF08: Local Noise

6.2.4.2 Failures

Mengingat kembali tentang gejala specular reflection, secara rasional, dapat diklaim bahwa ada peluang terjadinya kegagalan total dalam suatu pembacaan sonar. Jika hal itu terjadi, maka hasil pembacaan tersebut bernilai sama dengan spesifikasi pembacaan maksimal z_{max}. Pada prakteknya, hal ini sering terjadi. Oleh karena itu, error jenis ini perlu diperhitungkan dalam model persepsi probabilistik sonar.

Seperti disarankan dalam [3], pemodelannya adalah dengan sebuah point-mass distribution yang bertitik pusat di z_{max}. Oleh karena itu, penerapan persamaan (6.1) yang peluang persepsinya disebut p_{max} adalah sebagai berikut:

$$p_{max}(z_t^{sonar} \mid pose_t, m) = \begin{cases} 1 & ; if\ z_t^{sonar} = z_{max} \\ 0 & ; otherwise \end{cases} \quad (6.4)$$

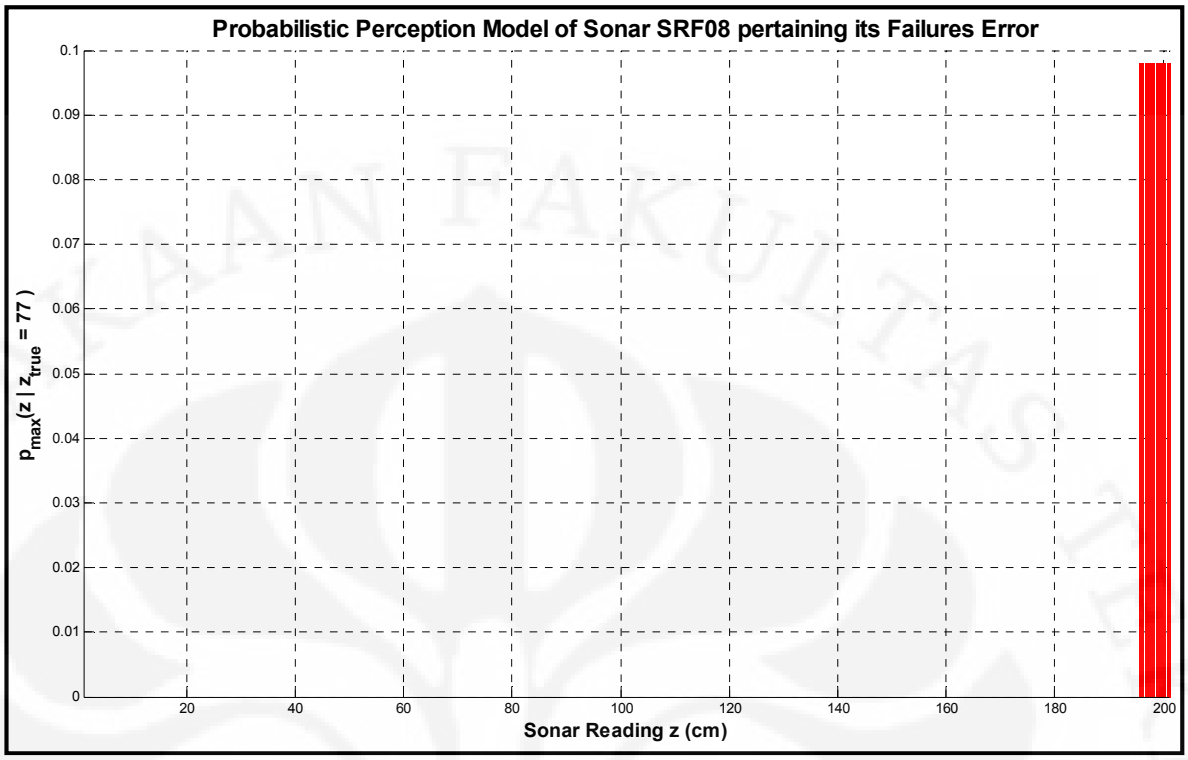
Sesungguhnya, secara teknis, p_{max} tidak mempunyai PDF karena merupakan discrete distribution. Namun demikian, diambil asumsi bahwa p_{max} dapat didekati dengan sebuah uniform distribution yang mempunyai titik tengah z_{max} dengan batas minimum = (z_{max} - 5.0955) dan batas maksimum = (z_{max} + 5.0955); nilai yang ditambahkan adalah standard-deviation dari local noise sonar. Dengan demikian, ada suatu PDF untuk p_{max}, contoh densitas untuk pembacaan sonar pada jarak aktual $z_t^{sonar,abs} = 77\ cm$ ditunjukkan pada gambar (6.7).

6.2.4.3 Random Measurements

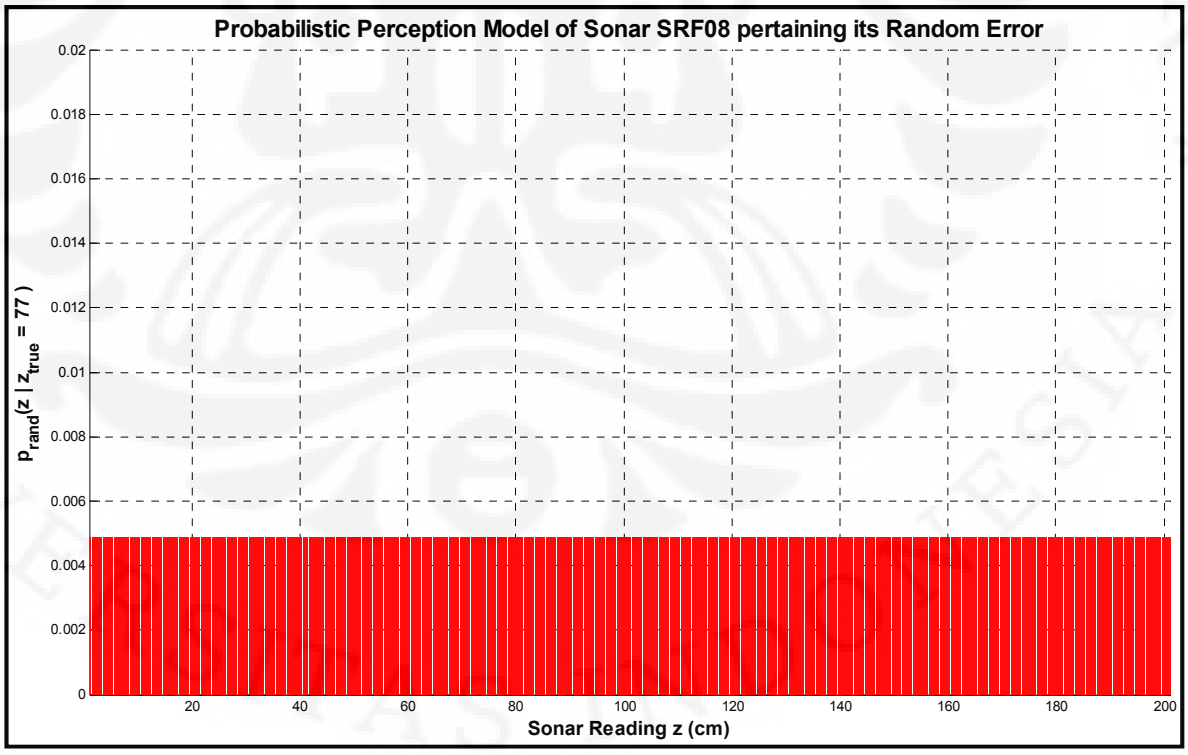
Para praktisi robotika, pada umumnya pernah mengalami gejala pembacaan sonar yang sungguh aneh (penulis menyebutnya sebagai phantom sonar-reading) karena tidak ada penjelasan logis tentang alasan terjadinya. Bisa jadi karena kombinasi dari tiga masalah dasar sonar yang diperburuk dengan karakteristik permukaan pantul untuk gelombang akustik sonar atau sifat udara (sebagai media perambatan) yang terlampau sukar serta tidak praktis untuk diidentifikasi.

Fenomena yang dijelaskan barusan terlalu berisiko untuk diabaikan sehingga perlu dimodelkan secara eksplisit sebagai bagian dari model persepsi sonar; dilambangkan dengan p_{rand}. Jalan yang ditempuh adalah memodelkannya dengan sebuah uniform distribution yang tersebar pada seluruh interval pembacaan sonar, dinyatakan pada persamaan (6.5). Visualisasi untuk pembacaan sonar pada jarak aktual $z_t^{sonar,abs} = 77\ cm$ ditunjukkan pada gambar (6.8).

$$p_{rand}(z_t^{sonar} \mid pose_t, m) = \begin{cases} \frac{1}{z_{max}} & ; if\ 0 \le z_t^{sonar} \le z_{max} \\ 0 & ; otherwise \end{cases} \quad (6.5)$$



Gambar 6.7 Visualisasi Model Persepsi Probabilistik Sonar SRF08: Failure Error

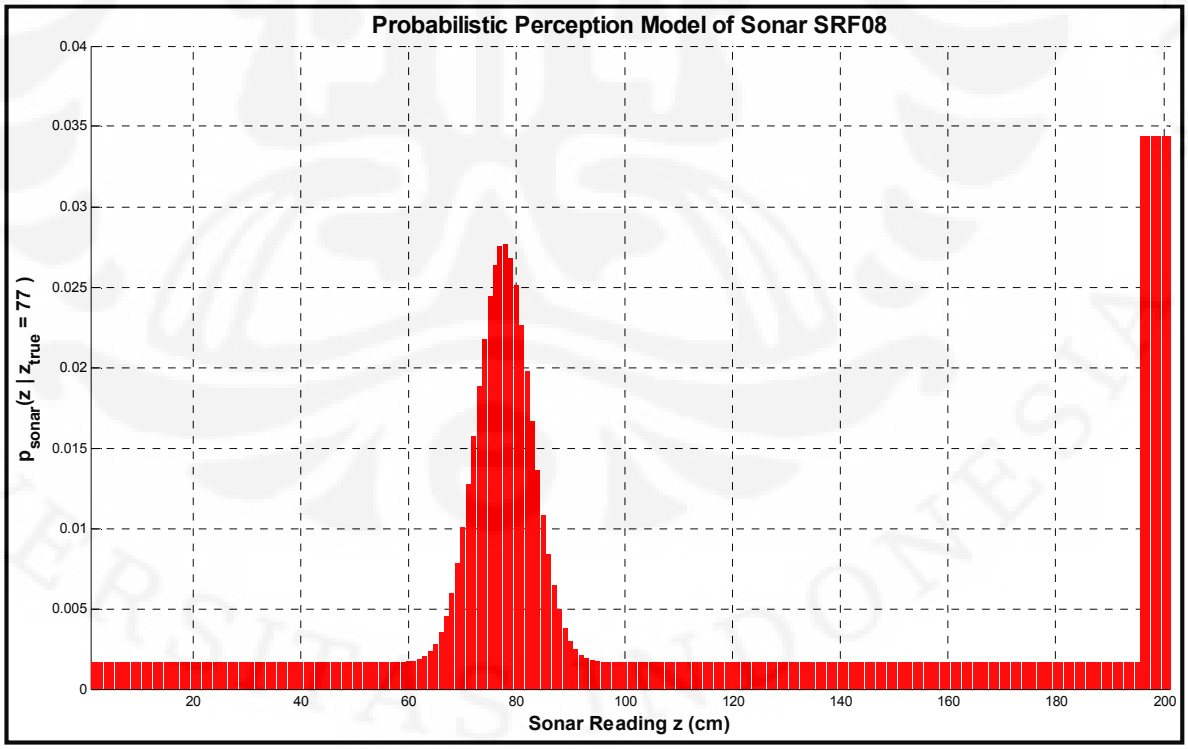


Gambar 6.8 Visualisasi Model Persepsi Probabilistik Sonar SRF08: Random Error

6.2.5 Pemodelan Probabilistik Sonar

Akhirnya, dapat ditentukan model probabilistik sonar yang mencakup ketiga jenis error pembacaannya. Teknik penggabungan ketiganya adalah dengan kombinasi linear sebagaimana disarankan oleh [3] yang diikuti dengan proses normalisasi sehingga memenuhi syarat suatu PDF, yaitu selalu bernilai positif kurang dari atau sama dengan 1 dan integrasi total nilai peluang dari domainnya sama dengan 1. Dengan demikian, model probabilistik sonar dapat ditunjukkan melalui persamaan (6.6), sedangkan visualisasinya dengan mengambil contoh untuk pembacaan sonar pada jarak aktual $z_t^{sonar,abs} = 77\ cm$ ditunjukkan pada gambar (6.9).

$$p_t^{sonar}(z_t^{sonar} \mid pose_t, m) = \frac{1}{3}\left[p_{hit}(z_t^{sonar} \mid pose_t, m) + p_{max}(z_t^{sonar} \mid pose_t, m) + p_{rand}(z_t^{sonar} \mid pose_t, m)\right] \quad (6.6)$$



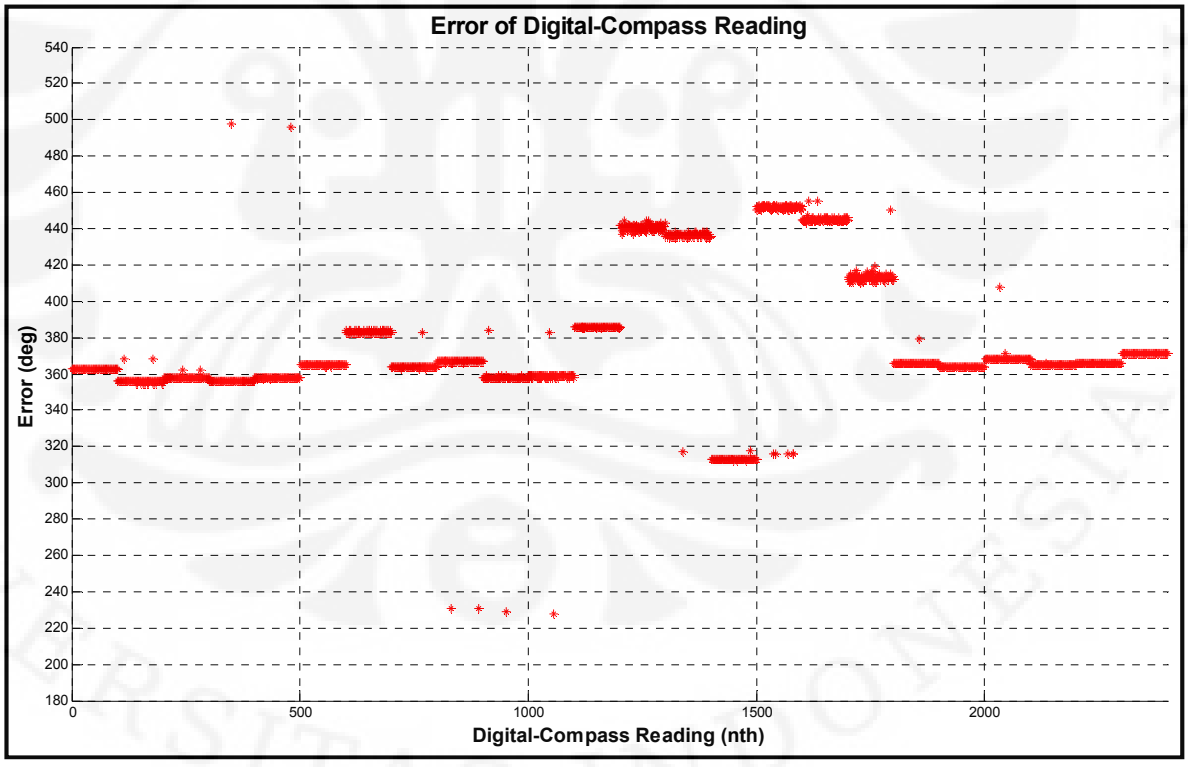
Gambar 6.9 Model Persepsi Probabilistik Sonar SRF08

6.3 Model Probabilistik Kompas-Digital CMPS03

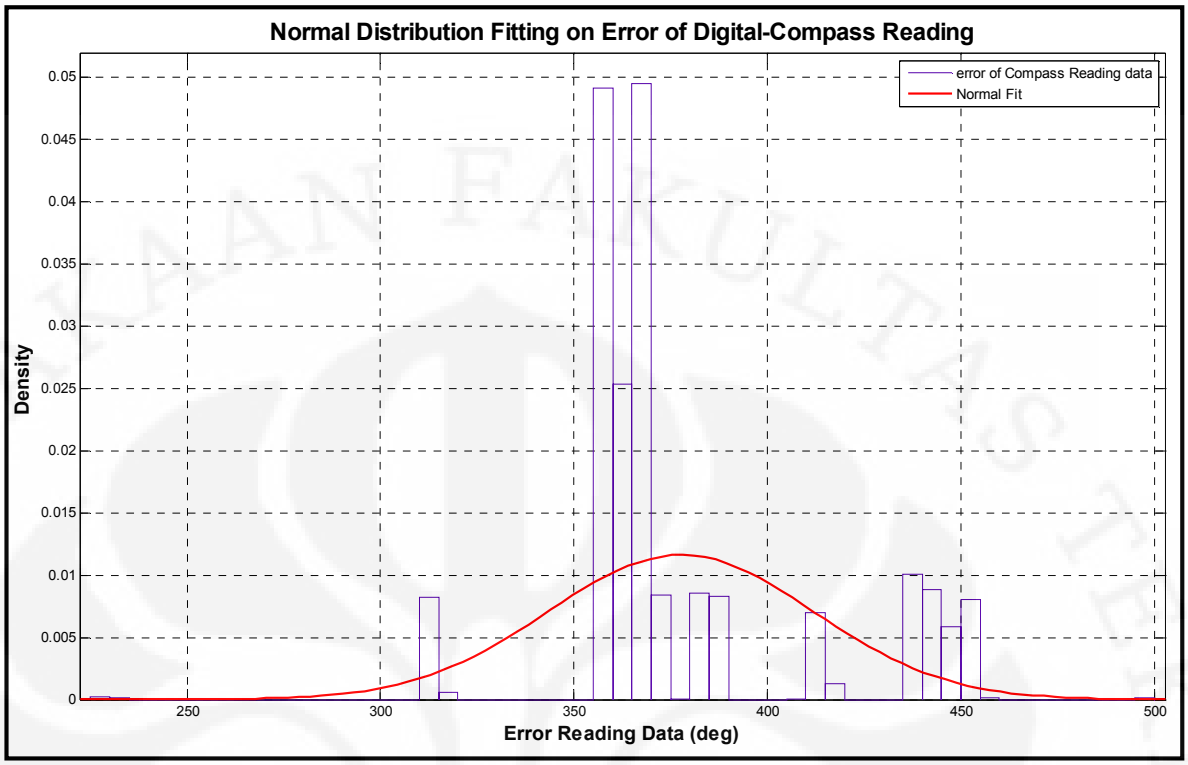
6.3.1 Hasil Eksperimen Pembacaan Kompas-Digital

Seperti halnya pada pembacaan sonar, dilakukan juga eksperimen terhadap pembacaan kompas-digital. Pada eksperimen ini didapatkan 2400 data yang kemudian diolah menjadi nilai error pembacaan kompas-digital dengan mengitung selisihnya dari orientasi-absolut yang merupakan bagian dari pose-absolut; rumusannya ada pada persamaan (6.7), sedangkan plotnya ditunjukkan pada gambar (6.9). Selanjutnya, dilakukan proses fitting data error tersebut dengan probability distribution Normal, hasilnya tersaji pada gambar (6.10) dan (6.11).

$$err_{cmps} = \theta_{abs} - z^{cmps} \tag{6.7}$$



Gambar 6.9 Error Pembacaan Kompas-digital



Gambar 6.10 Normal Fitting pada Error Pembacaan Kompas-Digital

Distribution: Normal
Log likelihood: -11889.4
Domain: -Inf < y < Inf
Mean: 377.628
Variance: 1176.63

Parameter	Estimate	Std. Err.
mu	377.628	0.700187
sigma	34.302	0.495262

Estimated covariance of parameter estimates:

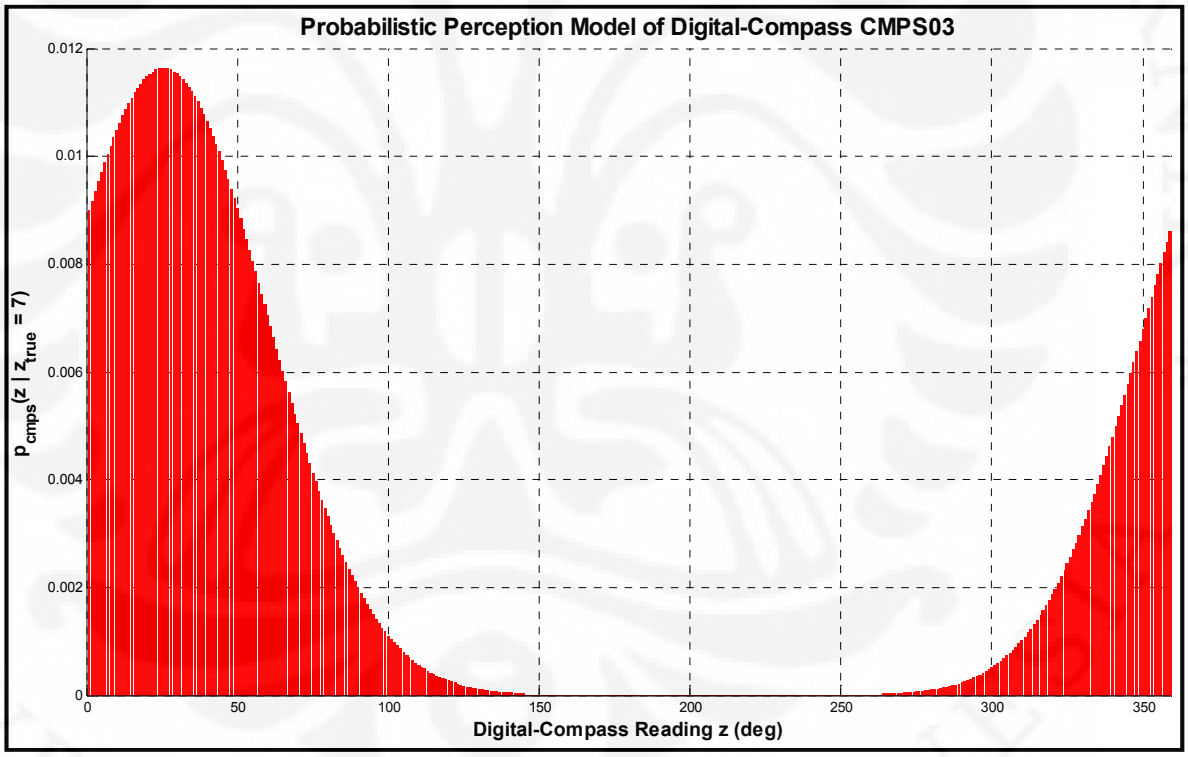
	mu	sigma
mu	0.490262	9.89747e-016
sigma	9.89747e-016	0.245284

Gambar 6.11 Catatan Normal Fitting pada Error Pembacaan Kompas-Digital

6.3.2 Pemodelan Probabilistik Kompas-Digital

Berdasarkan pengolahan data eksperimen pembacaan compass-digital, dapat dirumuskan model probabilistik kompas-digital CMPS03 dengan persamaan (6.8). Selain itu, contoh penerapannya pada pembacaan arah aktual $\theta_{abs} = z_t^{cmps,abs} = 7^\circ$ ditunjukkan pada gambar (6.12).

$$p_t^{cmps}\left(z_t^{cmps} \mid pose_t, m\right) = \begin{cases} N\left(\mu=[z_t^{cmps,abs}+17.628^\circ],\ \sigma=34.302^\circ\right); if\ 0^\circ \le z_t^{cmps} \le 359^\circ \\ ; otherwise \end{cases} \quad (6.8)$$



Gambar 6.12 Model Persepsi Probabilistik Kompas-Digital CMPS03

6.4 Peranan Model Persepsi Probabilistik pada MCL

Informasi persepsi berupa data pembacaan sensor-sensor exteroceptive (sonar SRF08 dan kompas-digital CMPS03) berperan dalam tahap weighting (importance) dalam MCL. Pada tahap tersebut tiap particle diberi bobot berdasarkan model persepsi probabilistik robot.

Sebagaimana diuraikan pada Bab 2, TMR Alfathvrss mempunyai empat buah sonar SRF08 dan sebuah kompas digital CMPS03. Lebih lanjut, keempat sonar itu adalah identik dan independent satu sama lain, begitu pula kompas dan semua sonar. Dengan demikian, model persepsi probabilistik TMR Alfahvrss dapat dinyatakan dengan persamaan (6.7).

$$
\begin{aligned}
& p_t(z_t \mid pose_t, m) = \\
& p_t^{sonar-front}\left(z_t^{sonar-front} \mid pose_t, m\right) \times \\
& p_t^{sonar-right}\left(z_t^{sonar-right} \mid pose_t, m\right) \times \\
& p_t^{sonar-rear}\left(z_t^{sonar-rear} \mid pose_t, m\right) \times \\
& p_t^{sonar-left}\left(z_t^{sonar-left} \mid pose_t, m\right) \times p_t^{cmps}\left(z_t^{cmps} \mid pose_t, m\right)
\end{aligned} \qquad (6.7)
$$

Pada tahap weighting MCL, nilai z_t^{abs} untuk model persepsi sonar adalah nilai hasil ray-casting pada particle, sedangkan untuk model persepsi kompas, nilainya adalah sama dengan nilai θ dari pose particle. Selain itu, pada persamaan (6.1) dan persamaan-persamaan lain yang mengikutinya, terdapat variabel m yang merupakan representasi dari peta lingkungan robot. Pada aplikasinya dalam riset ini, nilai m berperan dalam menentukan particle mana yang berada pada ruang-pose yang mungkin atau dalam lingkungan kerja robot. Dengan kata lain, jika suatu prediksi pose berada di luar lingkungan kerja robot maka bobotnya adalah nol. Lebih lanjut, m digunakan juga dalam proses ray-casting untuk mendapatkan

nilai absolut pembacaan sonar. Perlu disampaikan, bahwa dalam riset ini, m berupa persamaan garis objek-objek pada lingkungan. Akhirnya, algoritma weighting pada MCL ditunjukkan pada gambar (6.13).

```
function [w] = mclWeighting(PercInfo,x,y,theta,map)
    //PercInfo = [sonarFr sonarRi sonarRe sonarLe cmps]

    //(1)Obtain weight from sonars:---------------------------------
     [rcFr rcRi rcRe rcLe] = doRayCasting(round(x), round(y), theta, map);

     wSonarFr = getProbSonar(PercInfo(1),rcFr);
     wSonarRi = getProbSonar(PercInfo(2),rcRi);
     wSonarRe = getProbSonar(PercInfo(3),rcRe);
     wSonarLe = getProbSonar(PercInfo(4),rcLe);

    //(2)Obtain weight from a compass:------------------------------
     stdCmps = 34.302;
     theta = theta + 17.6280;
     upperLimit = theta + (7 * stdCmps);
     lowerLimit = theta - (7 * stdCmps);
     if (upperLimit > 359) && (PercInfo(5) <= (upperLimit-360))
         valCmps = PercInfo(5) + 360;
     elseif (lowerLimit < 0) && (PercInfo(5) >= (360 + lowerLimit))
         valCmps = theta + (360 - PercInfo(5) + theta);
     else
         valCmps = PercInfo(5);
     end

    wCmps = normpdf(valCmps,theta,stdCmps);

    //(3)Obtain total weight----------------------------------------
    w = wSonarFr * wSonarRi * wSonarRe * wSonarLe * wCmps;

    return w;
```

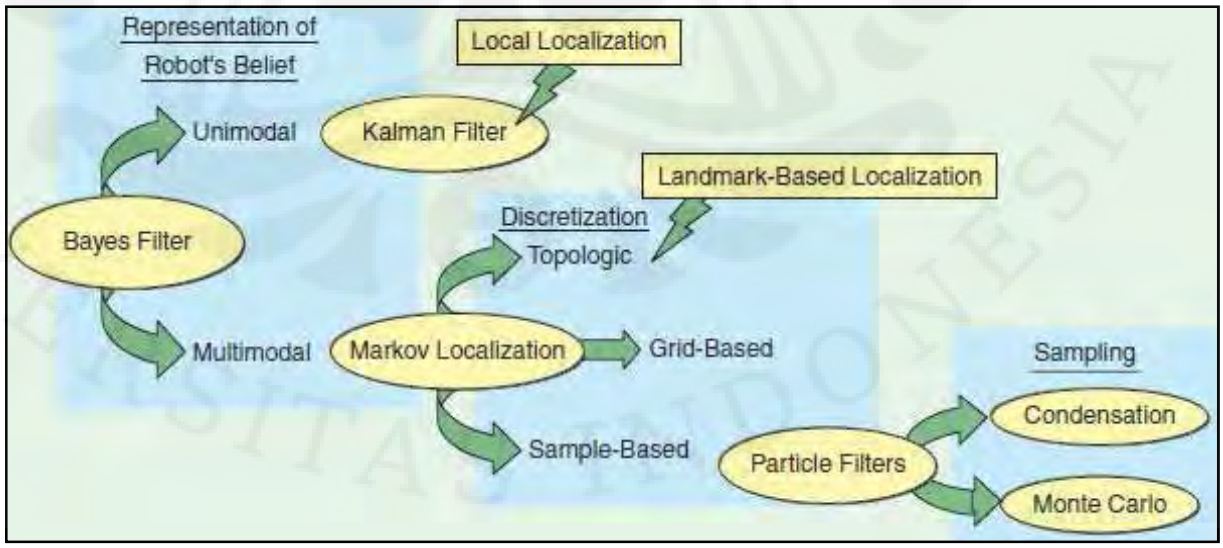
Gambar 6.13 Algoritma Weighting pada MCL

BAB 7

MCL PADA TMR ALFATHVRSS

7.1 Pendahuluan

MCL adalah suatu metode lokalisasi probabilistik yang diturunkan dari Bayes Filter melalui Markov Localization dan Particle Filter, gambar (7.1). Perbandingan yang komprehensif antara Markov Localization dan Kalman Filter dapat dilihat di [19]. Tampak pada gambar tersebut bahwa MCL berakar langsung pada Particle-Filters. Oleh karena itu, inti kerja dari MCL adalah memanfaatkan particle-particle (sebagai representasi robot) yang berpose untuk menyimpulkan pose robot. Versi awal dari MCL disebut Standard/Plain/Regular-MCL. Kelemahan yang paling meresahkan dari versi itu adalah kebutuhan akan sumber daya komputasi yang besar karena aturan dasarnya yaitu semakin banyak particle yang dilibatkan (sehingga kebutuhan komputasi semakin besar), semakin bagus kualitas simpulan pose (hasil lokalisasi). Lebih lanjut, sampai saat ini, telah ada beberapa varian dari Plain-MCL untuk mengatasi kekurangan tersebut atau yang lain (misal isu tentang spesifikasi sensors serta actuators), sebagai contoh: Sample-Adaptive-MCL [12], [23], Dual-MCL [25], Active-Particle-MCL [40], Augmented-MCL, Cluster-MCL [30], [46], dan Mixture-MCL [45].



Gambar 7.1 Hierarki Algoritma Lokalisasi Bayer Filter [13]

Sebagaimana dijelaskan dalam Bab 1, riset ini bertujuan untuk mula-mula mengimplementasikan Plain-MCL pada TMR Alfathvrss guna memecahkan masalah lokalisasinya; subbab 7.3. Oleh karena itu, muncul pertanyaan: seberapa baguskan kinerja Plain-MCL pada TMR Alfathvrss dengan fakta bahwa (a) odometry padanya berkategori buruk dan (b) sumber daya sensor yang minim yaitu empat buah sonar SRF08 dan sebuah kompas-digital CMPS03. Untuk itu, diperlukan standard kualitas algoritma lokalisasi untuk mengukur kinerja Plain-MCL, hal tersebut diuraikan pada subbab 7.2. Selanjutnya, dikembangkan suatu varian MCL yang mengoptimalkan pemakaian sumber daya komputasi. Varian tersebut dijelaskan secara detil pada subbab 7.4. Akhirnya, analisis perbandingan antara Plain-MCL dengan varian-barunya itu disajikan pada subbab 7.5.

7.2 Standard Kualitas Algoritma Lokalisasi

Solusi terbaik untuk lokalisasi adalah algoritma self-localization dengan kebutuhan komputasi seminimal mungkin dan kualitas hasil lokalisasi sebaik mungkin. Istilah "self-localization" berarti algoritma lokalisasi yang hanya mengandalkan sensor-sensor yang ada pada tubuh robot, baik exteroceptive maupun proprioceptive, untuk menyimpulkan pose robot; tidak membutuhkan informasi dari piranti khusus lokalisasi di luar robot, seperti GPS. Mengenai hal itu, algoritma MCL beserta variannya tergolong algoritma self-localization. Kemudian, berkaitan dengan kebutuhan komputasi, dalam riset ini, diasumsikan bahwa ia dapat diprediksi atau diwakili dengan waktu komputasi (computation-time, ct). Jadi semakin cepat waktu komputasi, maka kualitas algoritma tersebut semakin bagus.

Secara umum, kualitas hasil lokalisasi ditentukan oleh kuantitas error pose robot yang dihitung dengan persamaan (7.1a). Pada persamaan tersebut, dibutuhkan sebuah simpulan tentang pose robot hasil lokalisasi, $pose_t$, yang kemudian dicari selisihnya dengan sebuah pose-absolut, $pose_t^{abs}$. Akan tetapi, hasil algoritma lokalisasi turunan dari Particle-Filter berupa particle-particle sehingga terdapat

lebih dari satu pose hasil lokalisasi, $pose_t^n$; gambar (7.13). Oleh karena itu, dalam riset ini, didefinisikan dua jenis kuantisasi error-pose hasil algoritma MCL yaitu:

a) $err1_t^{pose}$ merupakan hitungan error bentuk umum; persamaan (7.1a), di mana sebuah pose simpulan adalah nilai rata-rata dari pose particle-particle; persamaan (7.1b)
b) $err2_t^{pose}$ merupakan proporsi jumlah particle yang berada di luar daerah fisik robot terhadap jumlah particle secara total; persamaan (7.2a), di mana particle yang di luar itu ditentukan dengan persamaan (7.2b).

$$err1_t^{pose} = abs(pose_t^{abs} - pose_t) \quad (7.1a)$$

$$pose_t = average(pose_t^n) \quad (7.1b)$$

$$err2_t^{pose} = \frac{\sum pose_t^{outside}}{\sum pose_t} \quad (7.2a)$$

$$pose_t^{n,outside} \epsilon\, pose_t^n \mid abs\left(pose_t^{abs} - pose_t^n\right) > R \quad (7.2b)$$

di mana $pose_t^{abs}$ adalah pose robot sebenarnya, nyata, atau aktual yang diberikan oleh penjejak pose-absolut (dipaparkan pada Bab 3), dan R adalah jari-jari chassis fisik robot. Untuk TMR Alfathvrss, ditetapkan R = 10 cm.

Untuk riset ini, ditetapkan kualitas algoritma lokalisasi yang ingin dicapai adalah sebagai berikut:

- $ct \leq 500ms$
- $err2_t^{pose} \leq 10\%$
- $err1_X \leq 3cm$
- $err1_Y \leq 3cm$
- $err1_\theta \leq 5°$

7.3 Plain-MCL

7.3.1 Algoritma Plain-MCL

Sesuai dengan asal mulanya, gambar (7.1), algoritma Plain-MCL merupakan pengembangan dari algoritma Bayes Filter dan Particle-Filter, keduanya ditunjukkan pada gambar (7.2) dan (7.3).

Secara matematis, algoritma lokalisasi Bayes-Filter dapat dijelaskan sebagai berikut. Dalam kerangka kerja probabilistik, lokalisasi dapat diartikan sebagai aktivitas memprediksi suatu posterior-belief bel($pose_t$) dengan berbekal informasi gerakan (kontrol) u dan persepsi z, persamaan (7.3). Kemudian, dengan Bayes-rule, asumsi-Markov, dan aturan PDF, maka persamaan tadi dapat ditransformasi menjadi persamaan (7.4) yang recursive, yang kemudian menjadi dasar algoritma Bayes-Filter. Lebih detil, asumsi Markov menyatakan bahwa probabilitas pada saat sekarang hanya bergantung pada probabilitas satu waktu sebelumnya [40], yang berarti bel($pose_t$) merupakan fungsi hanya dari bel($pose_{t-1}$) dan informasi gerakan u serta persepsi z yang terbaru [2].

$$bel(pose_t) = p(pose_t \,|z_{0:t}\,, u_{1:t}) \tag{7.3}$$

$$bel(pose_t) = \eta p(z_t \,|pose_t) \int p(pose_t|pose_{t-1,} u_t)\; bel(pose_{t-1})\, dpose_{t-1} \tag{7.4}$$

di mana η adalah konstanta normalisasi agar bel($pose_t$) memenuhi aturan PDF.

Lebih lanjut, pada algoritma Bayes-Filter, baris ketiga adalah tahap prediksi dengan model gerakan probabilistik, sedangkan baris keempat adalah tahap koreksi dengan model persepsi probabilistik. Sementara itu, pada algoritma Particle-Filter, kedua tahap tersebut didekomposisi menjadi tiga tahap yaitu sampling (S), importance/weighting (I), dan resampling (R). Akhirnya, algoritma Plain-MCL ditunjukkan pada gambar (7.4).

Tahapan pada Plain-MCL sama seperti Particle-Filter yaitu SIR. Dua tahap pertama telah dijelaskan pada Bab 5 dan Bab 6; baris keempat dan kelima pada gambar (7.4). Yang belum adalah tahap R (resampling) yang berada pada baris kedelapan sampai dengan kesebelas pada gambar yang sama. Oleh karena itu, akhir dari subbab ini difokuskan untuk pemaparan algoritma resampling.

1: **Algorithm Bayes_filter**($bel(x_{t-1}), u_t, z_t$)**:**
2: for all x_t do
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx$
4: $bel(x_t) = \eta\ p(z_t \mid x_t)\ bel(x_t)$
5: endfor
6: return $bel(x_t)$

Gambar 7.2 Algoritma Bayes-Filter [3]

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$)**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t\ 1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m - 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Gambar 7.3 Algoritma Particle-Filter [3]

1: **Algorithm MCL**($\mathcal{X}_{t-1}, u_t, z_t, m$)**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t\ 1}^{[m]})$
5: $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Gambar 7.4 Algoritma Plain-MCL [3]

Prinsip kerja tahap resampling adalah menyeleksi particle-particle hasil tahap sampling berdasarkan bobot mereka, di mana particle-particle digandakan sesuai dengan besar bobot (hasil tahap Weighting yang bersifat unnormalized), dan akhirnya particle-particle berbobot rendah kemungkinan besar terbuang (faktor duplikasi bernilai nol), sedangkan yang berbobot tinggi berpeluang tereplikasi berulang kali. Dengan kata lain, keluaran tahap resampling adalah particle-particle yang terkonsentrasi pada daerah yang mempunyai probabilitas terbesar.

Tahap resampling dapat dipandang sebagai satu kesatuan dengan tahap weighting/importance, keduanya biasa disebut dengan tahap koreksi. Lebih lanjut, telah terdapat beberapa algoritma resampling, seperti disebutkan dalam [59], yaitu multinomial resampling, stratified resampling, systematic resampling (SR), dan residual resampling (RR).

Untuk riset ini, dipilih algoritma resampling yang diajukan dalam [62] bernama residual systematic resampling (RSR), gambar (7.5), dengan alasan sebagai berikut:

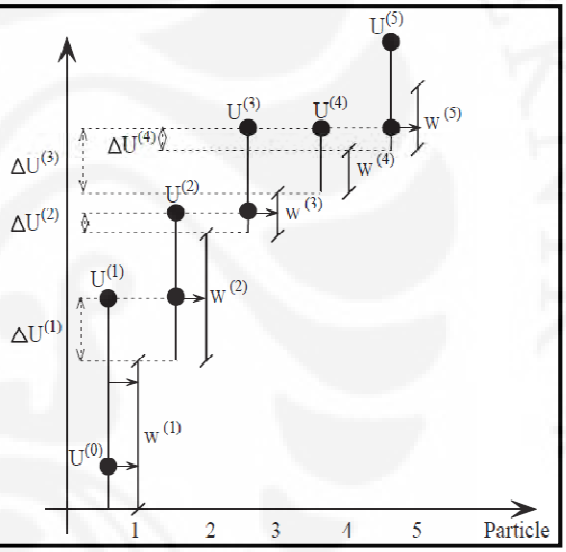
- RSR lebih unggul daripada RR karena ia menghindari iterasi kedua dari RR ketika residual-particle butuh untuk di-resample.
- Uniform random number pada RSR diproduksi dengan cara yang berbeda dengan RR sehingga memungkinkan hanya terjadi satu iterasi dan waktu komputasi bersifat independen terhadap distribusi bobot pada input.
- RSR lebih baik daripada SR karena while-loop pada SR diganti dengan for-loop dengan jumlah iterasi yang diketahui.
- RSR mengakomodasi variasi jumlah particle keluaran (hasil resampling) dengan mudah.
- RSR menghasilkan replication factor sehingga memungkinkan digunakannya particle allocation dengan replication factor dan index tertentu.

Purpose: Generation of an array of replication factors $\{r^{(m)}\}_{m=1}^{N}$ at time instant n, $n > 0$.
Input: An array of weights $\{w_n^{(m)}\}_{m=1}^{N}$, input and output number of particles, N and M, respectively
Method:

$(r) = RSR(N, M, w)$
$\Delta U^{(0)} \sim \mathcal{U}[0, \frac{1}{M}]$ //Generate random number $\Delta U^{(0)}$
for $m = 1$ **to** N // The main loop
$r^{(m)} = \lfloor (w_n^{(m)} - \Delta U^{(m-1)}) \cdot M \rfloor + 1$ // Computation of the replication factors
$\Delta U^{(m)} = \Delta U^{(m\ \ 1)} + \frac{r^{(m)}}{M} - w_n^{(m)}$ // updating the random number
end

Gambar 7.5 Algoritma Residual Systematic Resampling (RSR) [62]

m	$w^{(m)}$	$r^{(m)}$
1	7/20	2
2	6/20	1
3	2/20	1
4	2/20	0
5	3/20	1



Gambar 7.6 Contoh Penerapan Algoritma RSR dengan N = M = 5 particles [62]

Gambar (7.6) mengilustrasikan kerja algoritma RSR dengan jumlah particle masukan N sama dengan keluaran M sebanyak 5 buah, dimana bobot tiap particle diberikan pada tabel di sebelah kiri. Tampak pada gambar bahwa particle keempat terbuang (replication factor = 0) karena $\Delta U(3) = U(4) > w(4)$. Perlu disampaikan bahwa SR dan RSR menghasilkan hasil resampling yang relatif identik; perbedaan keduanya terletak pada jenis loop.

7.3.2 Eksperimen Local-Localization dengan Plain-MCL

7.3.2.1 Tujuan Ekperimen

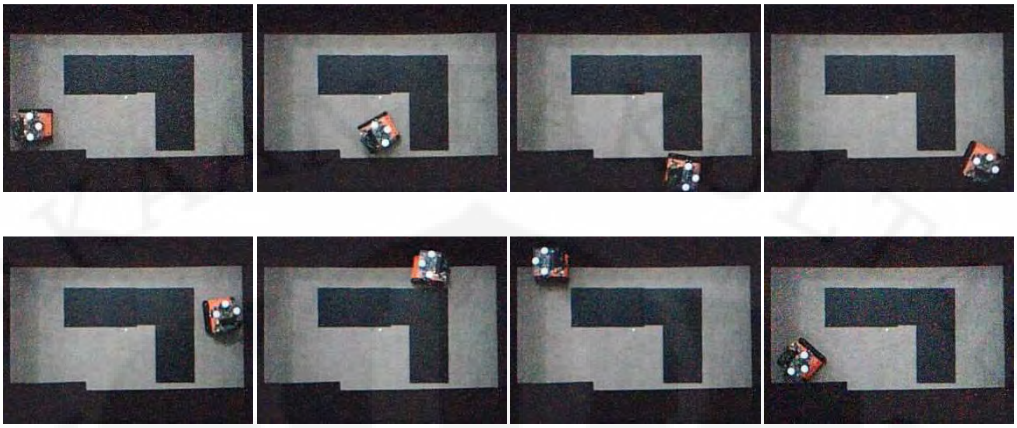
Eksperimen ini bertujuan sebagai berikut:

- Menerapkan Plain-MCL untuk local-localization (pose tracking) pada TMR Alfahvrss.
- Mendapatkan hubungan antara jumlah particle yang digunakan dengan kualitas hasil lokalisasi berupa parameter err1 dan err2 (subbab 7.2).
- Menentukan interval jumlah particle yang optimal untuk Plain-MCL berkaitan dengan standard kualitas algoritma lokalisasi.
- Membandingkan kinerja antara model gerakan probabilistik parametrik (Normal with-exclusion) dengan non-parametrik.

7.3.2.2 Detil Eksperimen

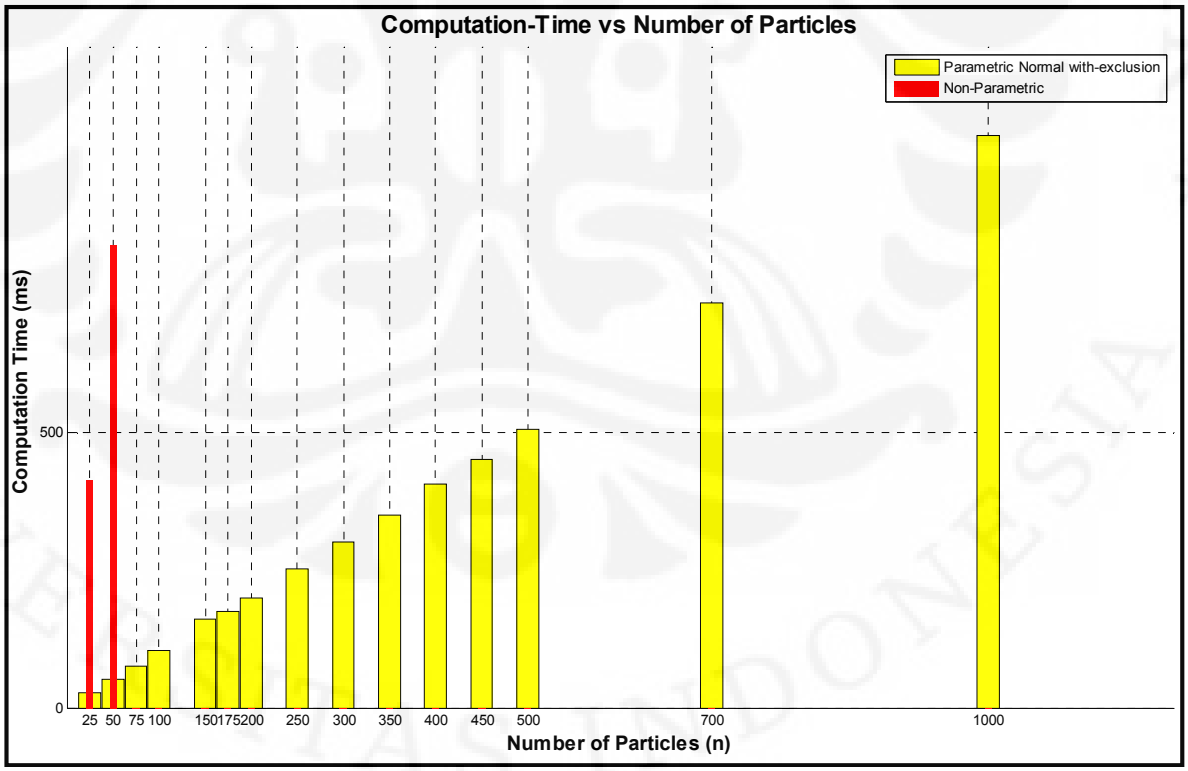
Untuk ekperimen ini, robot dikontrol secara manual untuk bergerak pada lingkungan eksperimen. Dalam pergerakannya melintasi trajectory, setiap berada pada titik-referensi-pose, data mengenai odometry (informasi kontrol) dan sensor (informasi persepsi) diakuisisi oleh komputer; terdapat 37 titik-referensi-pose, beberapa di antaranya ditunjukkan pada gambar (7.7). Kemudian, kedua data tersebut diolah dengan komputer eksternal. Aktivitas yang dimaksud adalah menjalankan algoritma Plain-MCL dengan variasi jumlah particle dan model gerakan probabilistik. Perlu disampaikan bahwa komputer yang digunakan mempunyai spesifikasi sebagai berikut: processor Intel®CoreTM Duo 1.66 GHz, 667 MHz FSB, 2 MB L2 Cache, dan 2 GB DDR2 RAM.

Kemudian, nilai parameter kinerja Plain-MCL (err1 dan err2) untuk jumlah particle tertentu dihitung dengan mendekati himpunan data err1 dan err2 dari tiap titik-referensi-pose dengan probability distribution Normal sehingga didapatkan nilai mean dan standard-deviation dari ketigapuluh-tujuh hasil lokalisasi tadi.

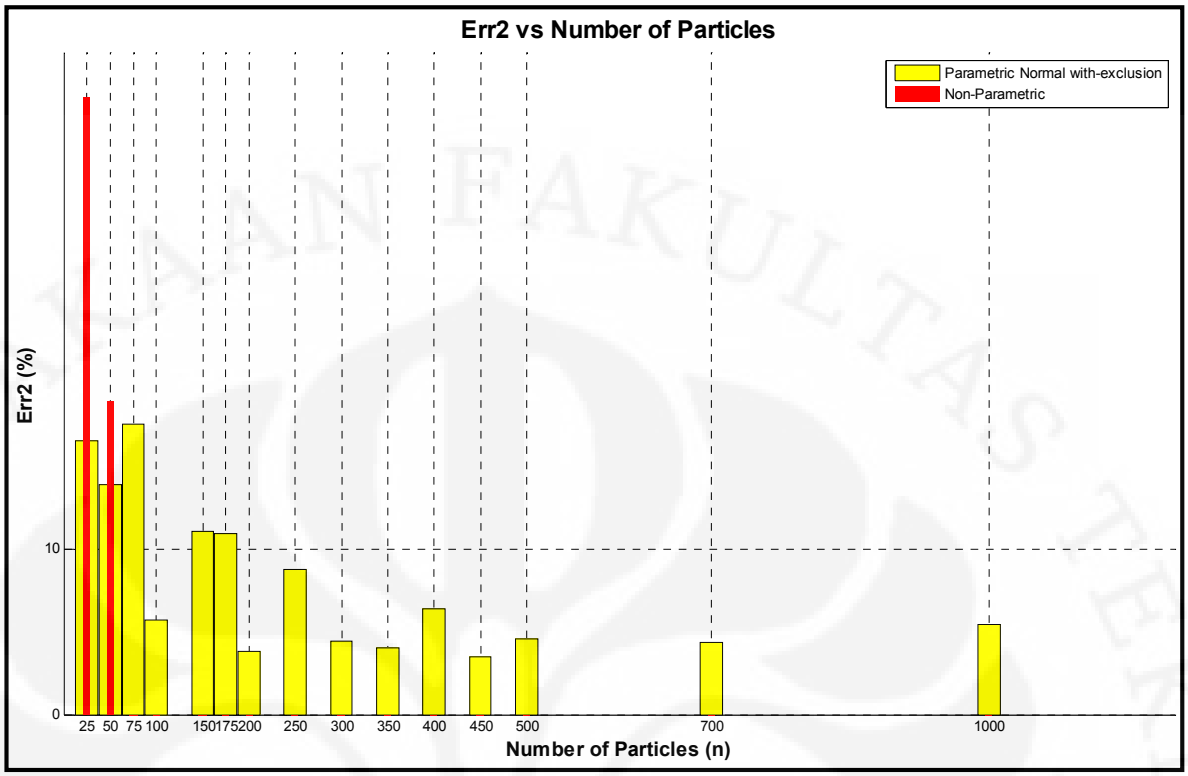


Gambar 7.7 Beberapa titik-referensi-pose pada Lingkungan-A

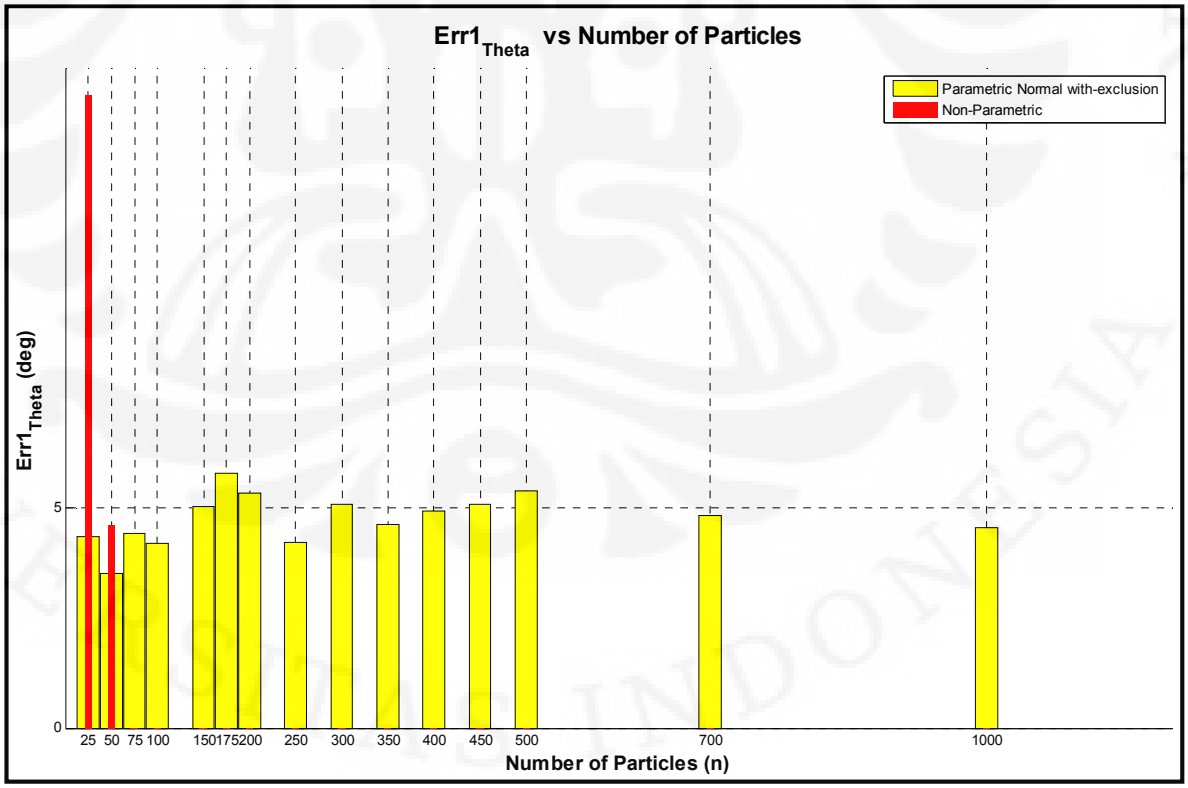
7.3.2.3 Hasil Eksperimen



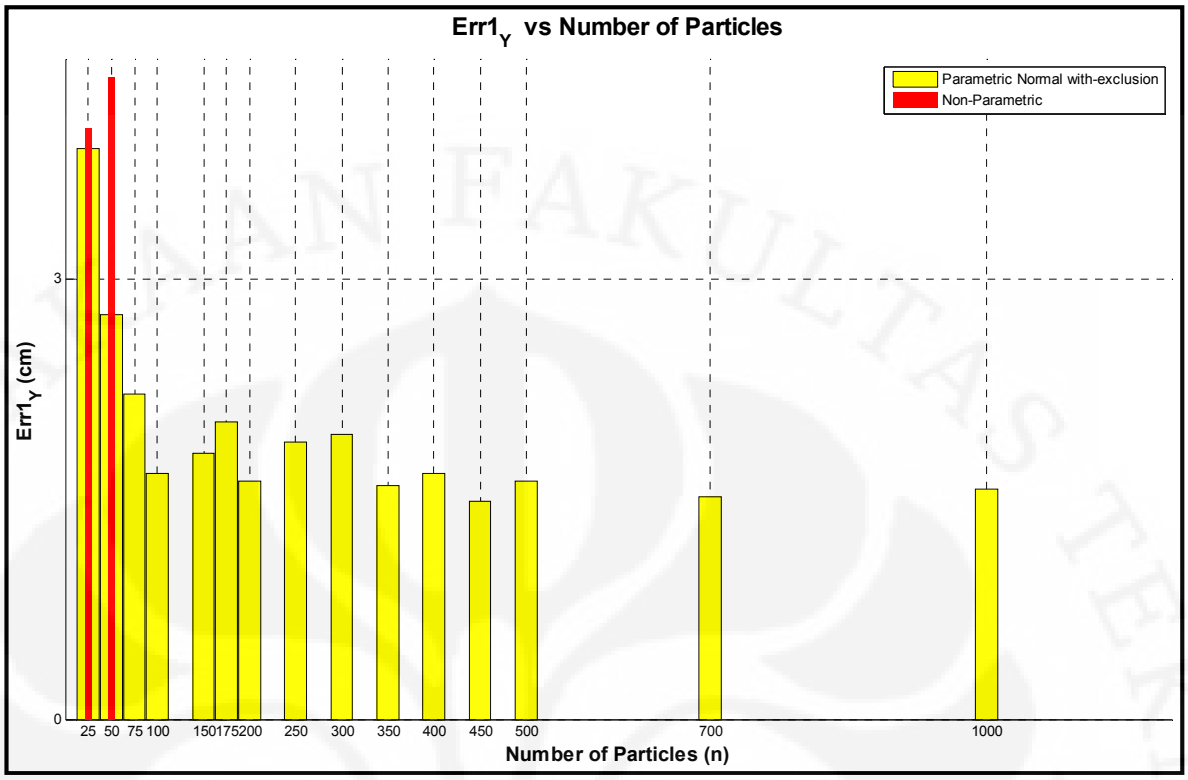
Gambar 7.8 Grafik Hubungan Waktu Komputasi dengan Jumlah Particle



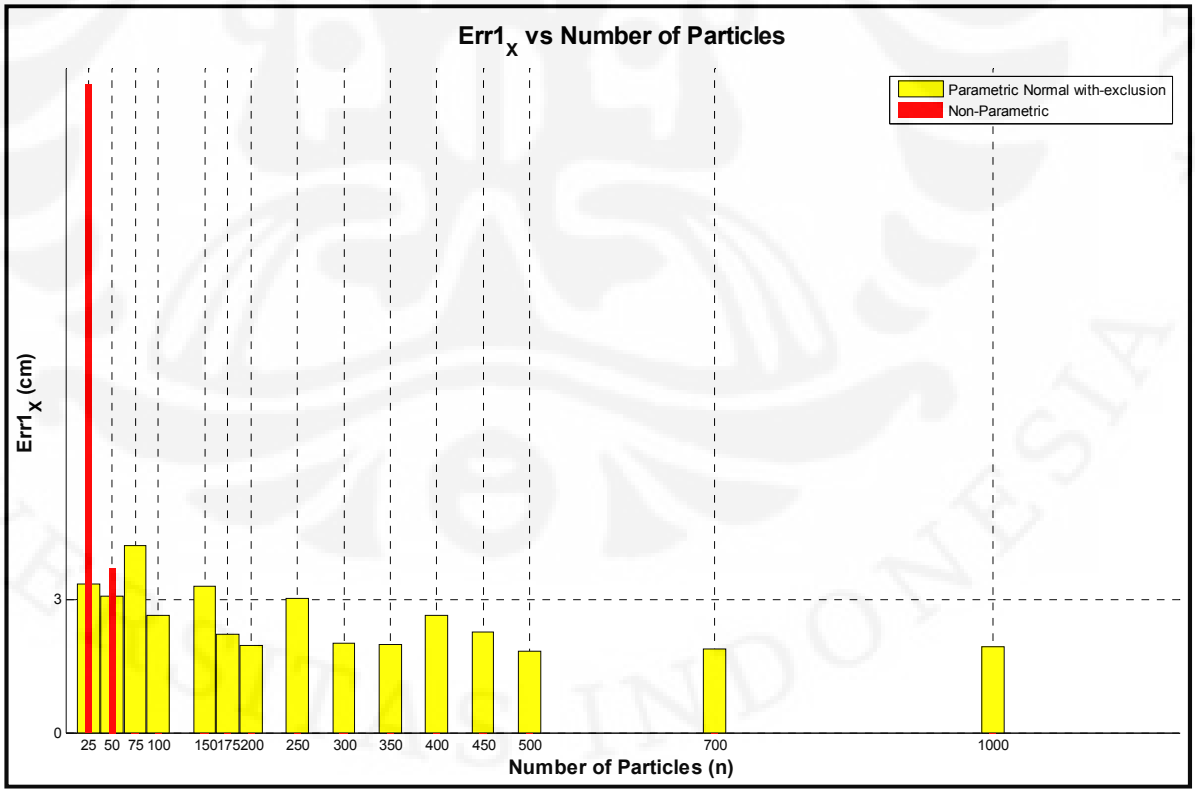
Gambar 7.9 Grafik Hubungan Err2 dengan Jumlah Particle



Gambar 7.10 Grafik Hubungan $Err1_{Theta}$ dengan Jumlah Particle



Gambar 7.11 Grafik Hubungan $Err1_Y$ dengan Jumlah Particle



Gambar 7.12 Grafik Hubungan $Err1_X$ dengan Jumlah Particle

7.3.2.4 Analisis Hasil Eksperimen

Salah satu tujuan dari eksperimen ini adalah membandingkan kinerja dua model gerakan probabilistik yaitu Normal with-exclusion dan non-parametrik. Bar berwarna merah pada gambar (7.8), (7.9), (7.10), (7.11), dan (7.12) adalah nilai parameter untuk Plain-MCL dimana tahap sampling dilakukan dengan model gerakan non-parametrik, sedangkan yang berwarna kuning sampling dengan model gerakan parametrik. Tampak bahwa bar berwarna merah selalu melebihi bar berwarna kuning pada kelima grafik. Dapat disimpulkan bahwa kinerja model parametrik lebih bagus daripada model non-parametrik. Kegagalan model non-parametrik disebabkan oleh kesulitan mendapatkan persamaan garis yang mendekatinya dengan baik. Dengan demikian, pembahasan selanjutnya hanya ditujukan untuk Plain-MCL dengan model gerakan probabilistik Normal with-exclusion.

Seperti terlihat pada gambar (7.8), besar waktu komputasi Plain-MCL sebanding dengan jumlah particle yang digunakan. Merujuk pada standard kualitas algoritma, di mana waktu komputasi dibatasi pada 500ms, maka dapat disimpulkan bahwa jumlah particle terbesar yang diperbolehkan adalah 500 buah particle; n_{max} = 500. Perlu ditekankan bahwa waktu komputasi ini sangat bergantung pada spesifikasi processor yang menjalankan algoritma. Seperti disampaikan pada bagian detil eksperimen (subbab 7.3.2.2), untuk implementasi Plain-MCL pada TMR Alfathvrss, algoritma dijalankan dengan processor di luar robot (offline computation).

Mengenai err2, yaitu prosentase particle yang berada di luar chassis fisik robot (outsiders), berdasarkan gambar (7.9), dapat diklaim bahwa nilainya relatif sama (di bawah 10%) untuk interval jumlah particle dari 200 sampai 1000 buah particle. Namun, menjadi di atas 10% dengan jumlah particle kurang dari 200 buah. Sementara itu, untuk err1 yang didekomposisi menjadi $err1_X$, $err1_Y$, dan $err1_\theta$, trend kenaikan nilai err1 dengan berkurangnya jumlah particle dapat diobservasi

dari gambar (7.10), (7.11), dan (7.12). Untuk $err1_X$, nilainya melonjak di atas 3 cm dengan jumlah particle 75 buah. Gejala itu terjadi juga pada $err1_Y$ dengan jumlah particle kurang dari 50 buah. Sementara itu, untuk $err1_\theta$, nilai di atas 5˚ terjadi saat jumlah particle sekitar 175 buah. Akhirnya untuk memenuhi kualitas hasil lokalisasi yang telah ditetapkan (subbab 7.2), diputuskan jumlah particle terkecil untuk impementasi Plain-MCL adalah n_{min} = 200 buah particle.

Ciri khas Plain-MCL adalah jumlah particle yang digunakan bersifat tetap sekali setelah diinialisasi atau dalam satu skenario lokalisasi. Untuk itu, berdasarkan kesimpulan analisis di atas, jumlah particle optimal untuk Plain-MCL pada TMR Alfathvrss adalah nilai tengah dari n_{min} dan n_{max} yaitu n = 350 particles. Berdasarkan hasil ekperimen, Plain-MCL dengan jumlah particle tersebut mempunyai kuantitas $err1_X$ = 2.0000 cm (σ = 1.2472 cm), $err1_Y$ = 1.5946 cm (σ = 1.4992 cm), $err1_\theta$ = 4.6216˚ (σ = 17.2133˚), err2 = 4.0541% (σ = 6.6612%), dan ct = 348.6757 ms (σ = 50.3808 ms). Akhirnya, dapat diklaim bahwa algoritma Plain-MCL dapat mengatasi masalah local-localization pada TMR Alfathvrss.

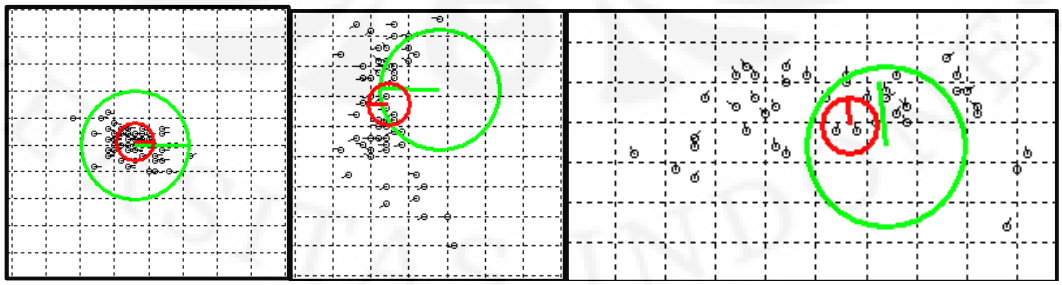
7.4 Dynamic-MCL

7.4.1 Ide Dasar

Sebagaimana disebutkan tadi, pada Plain-MCL, jumlah particle tetap selama satu skenario kerja robot. Sementara itu, telah disepakati bahwa solusi terbaik lokalisasi adalah “self-localization yang tercepat dengan kualitas hasil terbagus”. Kemudian, berdasarkan analisis mengenai Plain-MCL, terdapat suatu rentang jumlah particle di mana Plain-MCL bekerja secara optimal, yaitu 200 sampai 500 particles. Selain itu, telah jelas bahwa jumlah particle berbanding lurus dengan waktu komputasi. Akhirnya, timbul sebuah gagasan untuk membuat jumlah particle bersifat dinamis; jumlahnya berubah-ubah selama satu scenario. Varian MCL yang demikian diberi nama Dynamic-MCL.

Perlu disampaikan bahwa pada [71], disebutkan pula istilah dynamic pada MCL. Akan tetapi, kata “dynamic” pada literatur itu berasosiasi dengan sifat lingkungan yang berubah-ubah konfigurasinya selama robot bekerja; bukan pada MCL itu sendiri. Oleh karena itu, dapat diklaim bahwa ide dinamis di sini berbeda dengannya. Selain itu, dalam [12], [76] diajukan varian yang juga memvariasikan jumlah sample selama robot bekerja. Perbedaan mendasar antara varian itu dengan Dynamic-MCL terletak pada pendekatan atau acuan dalam menentukan jumlah sample dan proses integrasi dengan algoritma MCL.

Inti kerja dari MCL adalah memanfaatkan banyak particle untuk merepresentasinya pose. Idealnya, di mana err1 dan err2 bernilai nol, particle-particle tersebut terkonsentrasi di satu titik yaitu pose yang sama dengan pose-absolut. Jika particle-particle tersebut tersebar, maka semakin besar sebaran particle, semakin besar kemungkinan pose hasil lokalisasi berbeda dengan pose-absolut; robot semakin tidak pasti terhadap posenya, gambar (7.13). Pada saat itu, secara wajar, diperlukan tambahan particle dengan harapan bahwa robot akan lebih percaya diri tentang posenya pada waktu berikutnya. Logika itu didasarkan pada prinsip dasar MCL yaitu semakin banyak jumlah particle yang dilibatkan, semakin bagus kualitas hasil lokalisasi. Selanjutnya, jika robot telah yakin kembali dengan posenya, particle-particle terkonsentrasi lagi, maka jumlah particle dapat dikurangi. Jadi acuan perubahan jumlah particle adalah persebaran particle terhadap sebuah pose-simpulan, yaitu satu titik pose hasil lokalisasi yang mengikuti persamaan (7.1b).



Gambar 7.13 Ilustrasi Persebaran Particles

7.4.2 Proses Implementasi

Untuk merealisasikan gagasan di atas, maka perlu dicari jawaban tentang acuan perubahan jumlah particle dan proses integrasinya dengan algoritma MCL. Subbab ini membahas yang pertama, sedangkan yang kedua dibahas di subbab berikutnya.

Secara garis besar, gagasan itu adalah persebaran particle-particle mempengaruhi jumlah particle. Pada pendekatan ini, persebaran diindikasikan dengan nilai parameter Spread-Factor (S). Dengan demikian, jawaban pertanyaan tadi dapat dirumuskan oleh persamaan (7.5) berikut.

$$n = f_n(S) \tag{7.5}$$

Dalam ilmu statistika, persebaran untuk suatu data yang mengikuti distribusi Normal ditunjukkan dengan nilai standard-deviation-nya. Dengan mengambil asumsi bahwa persebaran particle-particle cukup valid untuk didekati dengan kurva Normal, maka ditetapkan bahwa S adalah standard-deviation dari data jarak (R) antara pose-simpulan dengan pose particle-particle; persamaan (7.1b), (7.6), dan (7.7). Contoh variasi nilai S pada suatu scenario lokalisasi menggunakan Plain-MCL dengan 37 titik-referensi-pose ditunjukkan pada gambar (7.14).

$$S = std(\{R^1, R^2, R^3, \dots, R^n\}) \tag{7.6}$$

$$R^n = \sqrt{(pose_x^n - pose_x)^2 + \left(pose_y^n - pose_y\right)^2} \tag{7.7}$$

Selanjutnya, merujuk pada ide dasar, persebaran particle berpengaruh pada kualitas lokalisasi yaitu nilai parameter err1 dan err2. Hal tersebut berarti terdapat

suatu hubungan di antara keduanya yang dapat dinyatakan dalam persamaan (7.8) dan (7.9) dan ditunjukkan pada gambar (7.15), (7.16), (7.17), dan (7.18).

$$err1_x = f_{err1_x}(S) \tag{7.8a}$$

$$= 2.1\,S^7 - 36.1\,S^6 + 259\,S^5 - 1000.5\,S^4 + 2245.1\,S^3 - 2922.9\,S^2 + 2043.3\,S - 592.3$$

$$err1_y = f_{err1_y}(S) \tag{7.8b}$$

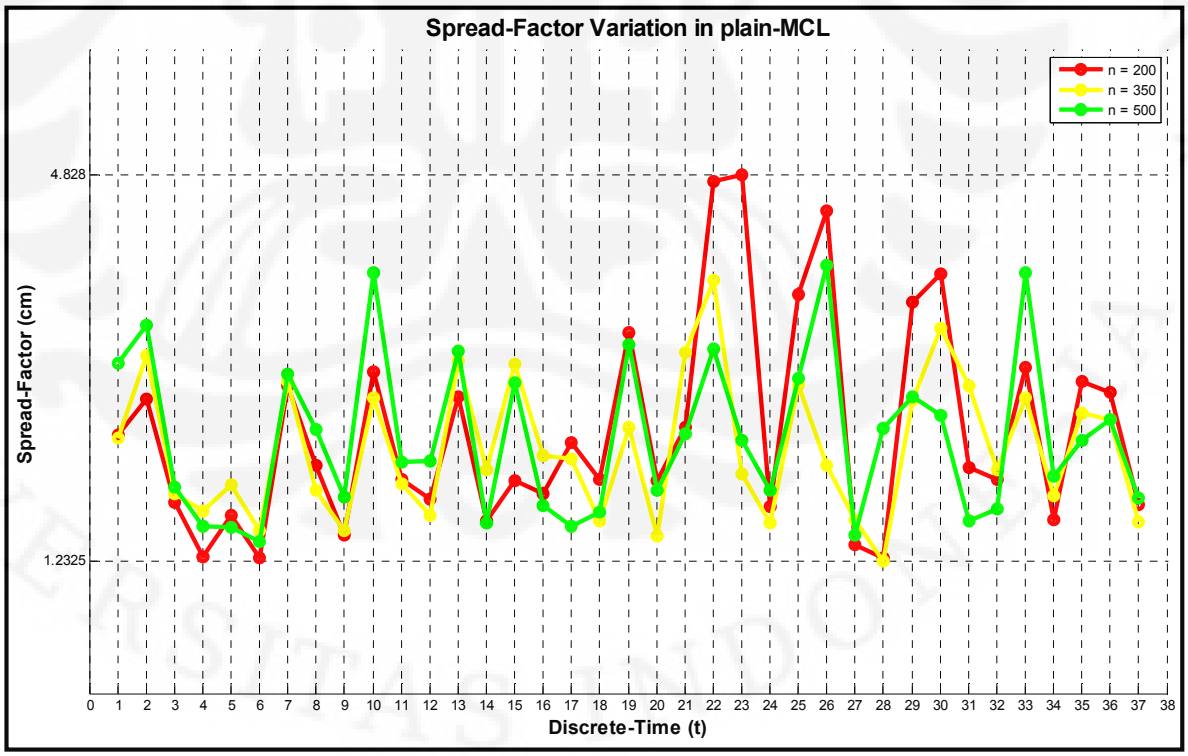
$$= 2.5\,S^7 - 42.3\,S^6 + 298.7\,S^5 - 1134.8\,S^4 + 2299.1\,S^3 - 3186.4\,S^2 + 2179.2\,S - 618.0$$

$$err1_\theta = f_{err1_\theta}(S) \tag{7.8c}$$

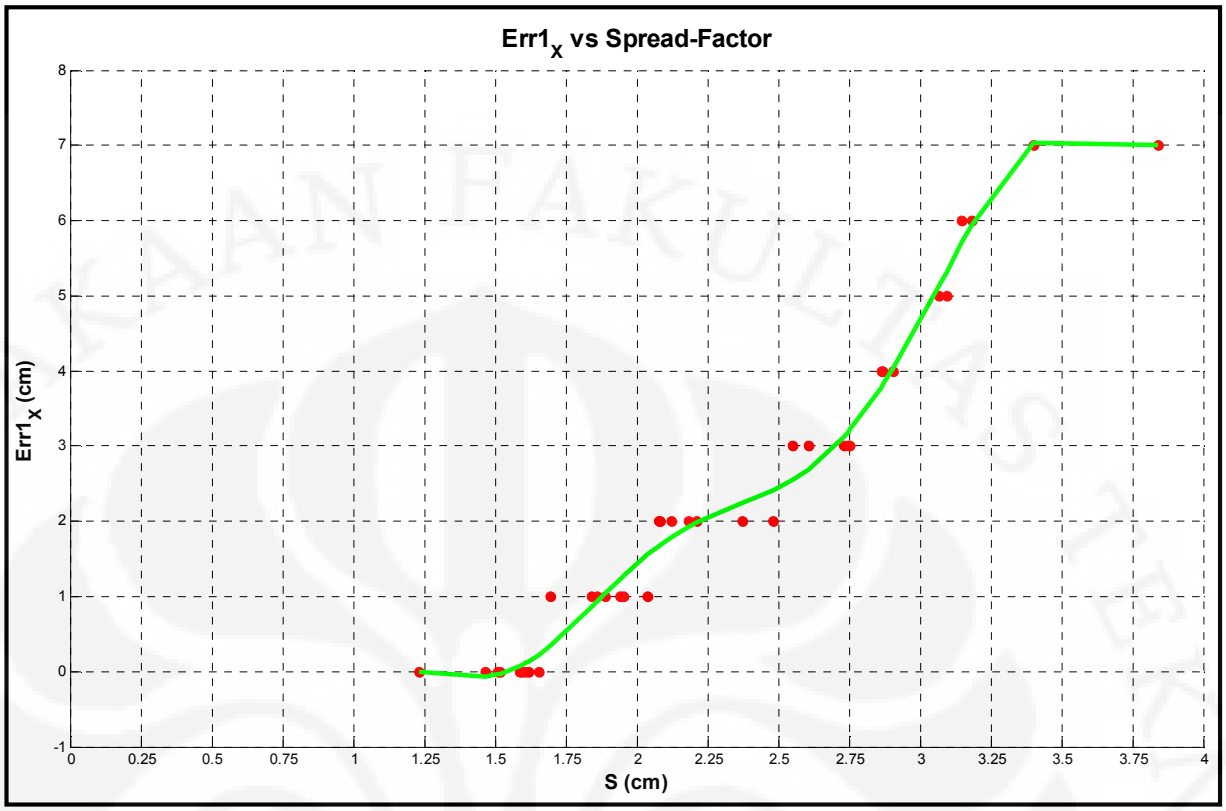
$$= 8.3\,S^7 - 136.9\,S^6 + 936.2\,S^5 - 3432.6\,S^4 + 7278.5\,S^3 - 8921.7\,S^2 + 5864.7\,S - 1600.6$$

$$err2 = f_{err2}(S) \tag{7.9}$$

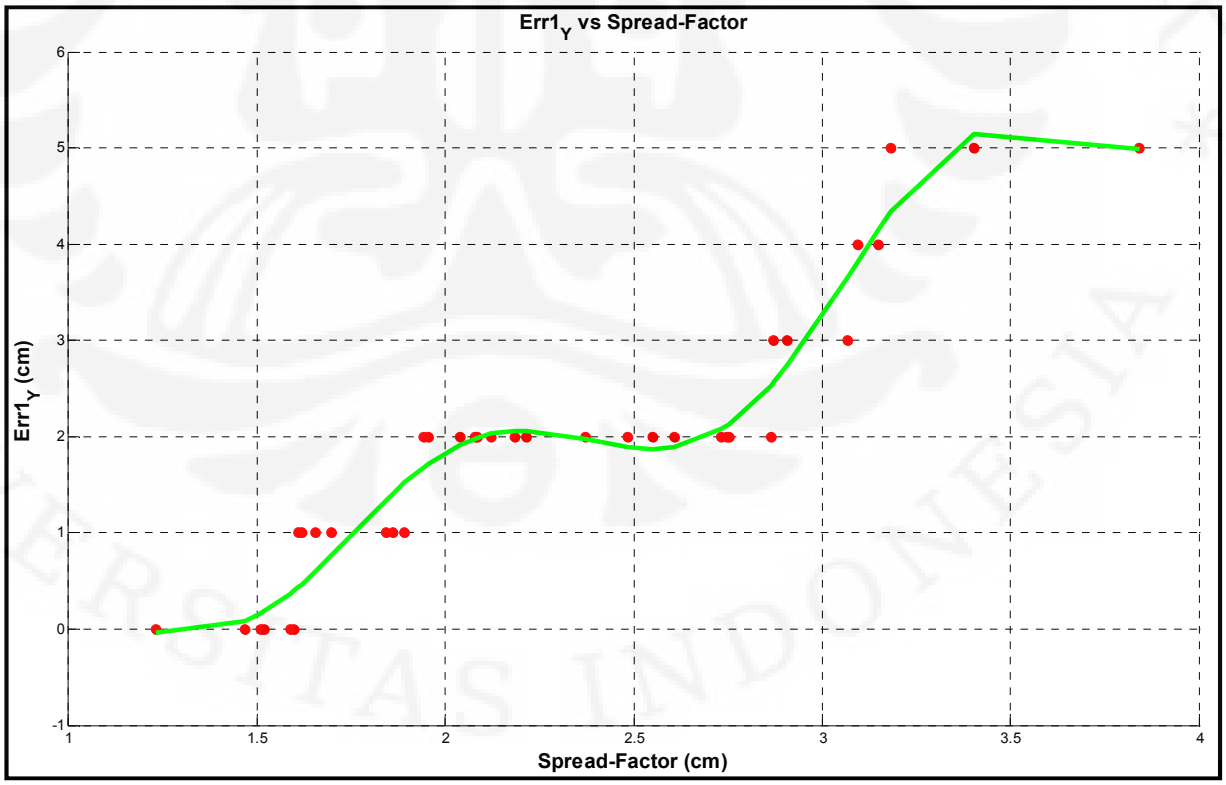
$$= -2.2\,S^7 + 33.0\,S^6 - 203.6 + 678.6 - 1327.2\,S^3 + 1533.9\,S^2 - 976.9S + 256.8$$



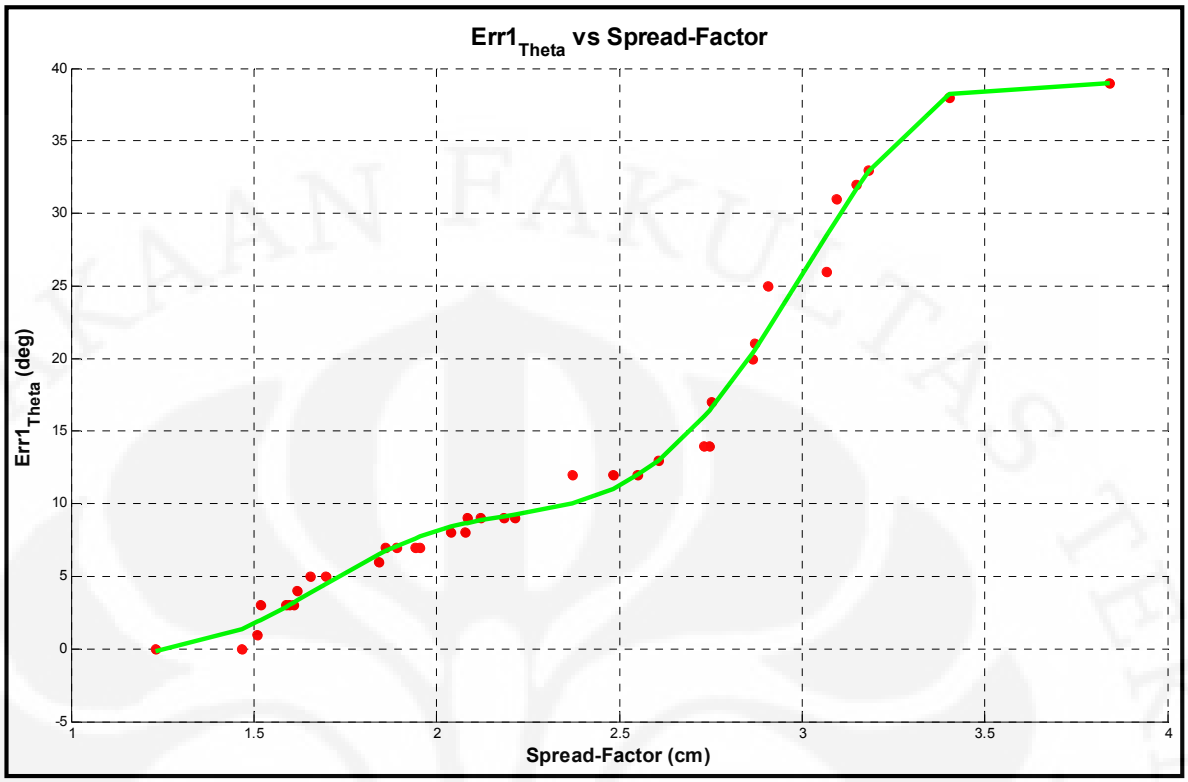
Gambar 7.14 Variasi S dalam Plain-MCL



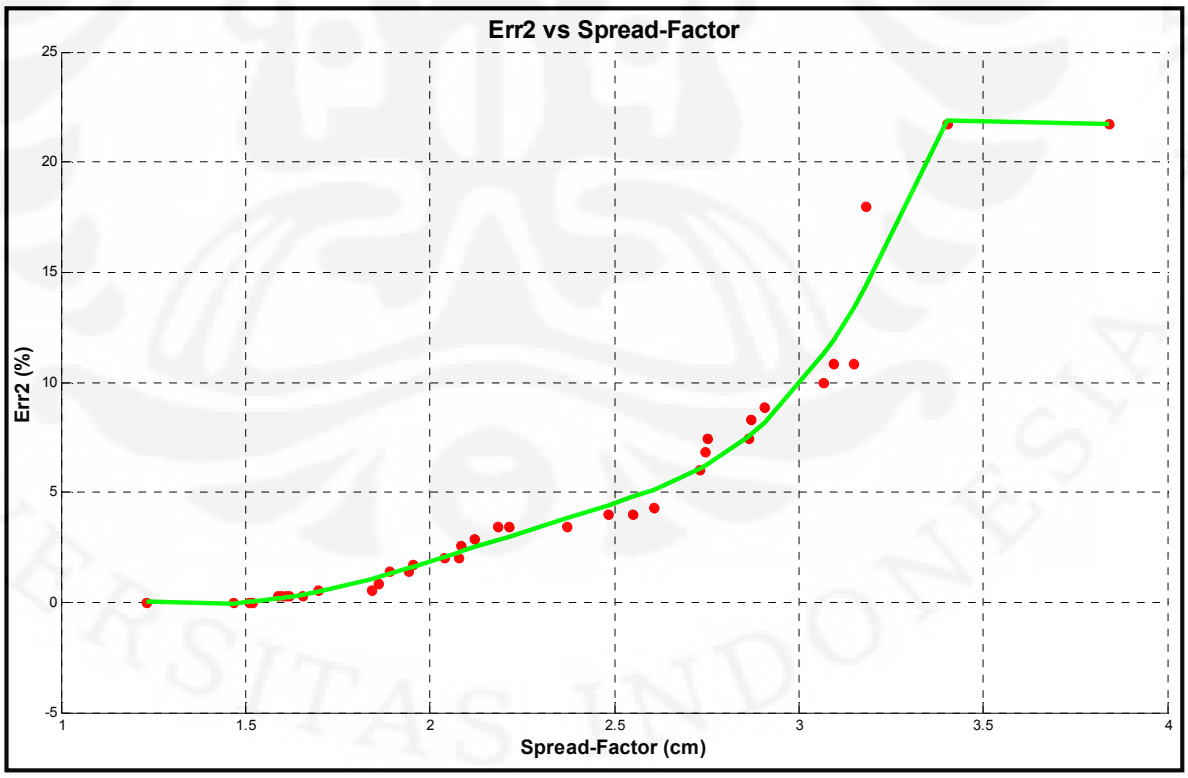
Gambar 7.15 Hubungan $Err1_X$ dengan Spread-Factor pada Plain-MCL n = 350



Gambar 7.16 Hubungan $Err1_Y$ dengan Spread-Factor pada Plain-MCL n = 350



Gambar 7.17 Hubungan $Err1_{Theta}$ dengan Spread-Factor pada Plain-MCL n = 350



Gambar 7.18 Hubungan Err2 dengan Spread-Factor pada Plain-MCL n = 350

Lebih lanjut, secara lugas, error lokalisasi dengan MCL dapat diatasi dengan menambah jumlah particle yang dilibatkan. Jadi, jika diketahui nilai error maka jumlah particle yang perlu ditambahkan (untuk mengurangi error itu) dapat ditentukan. Oleh karena itu, dibutuhkan prediksi err1 dan err2 dengan dasar S. Analisis tentang Plain-MCL memberikan rentang jumlah particle yang optimal, yang mempunyai nilai tengah n = 350. Dengan demikian, cukup beralasan jika persamaan yang memprediksi err1 dan err2 adalah persamaan garis pada gambar (7.15), (7.16), (7.17), dan (7.18) yaitu persamaan (7.8) dan (7.9), dengan catatan: nilai error maksimal adalah nilai error yang dihasilkan oleh S terbesar pada Plain-MCL (n = 350), yaitu S = 3.8413 cm dan nilai error minimal adalah nilai error yang dihasilkan oleh S terkecil pada Plain-MCL (n = 350), yaitu S = 1.2325 cm.

Selanjutnya, nilai-nilai prediksi error digunakan untuk menentukan jumlah particle n, di mana hubungan keduanya didekati secara linear dalam batas-batas tertentu; ditunjukkan oleh persamaan (7.10), (7.11), dan (7.12). Batas-batas tersebut adalah nilai maksimal tiap-tiap jenis error sebagaimana telah ditetapkan sebagai kuantitasi dari kualitas hasil lokalisasi. Akhirnya, persamaan (7.5) dapat diplot seperti ditunjukkan pada gambar (7.19).

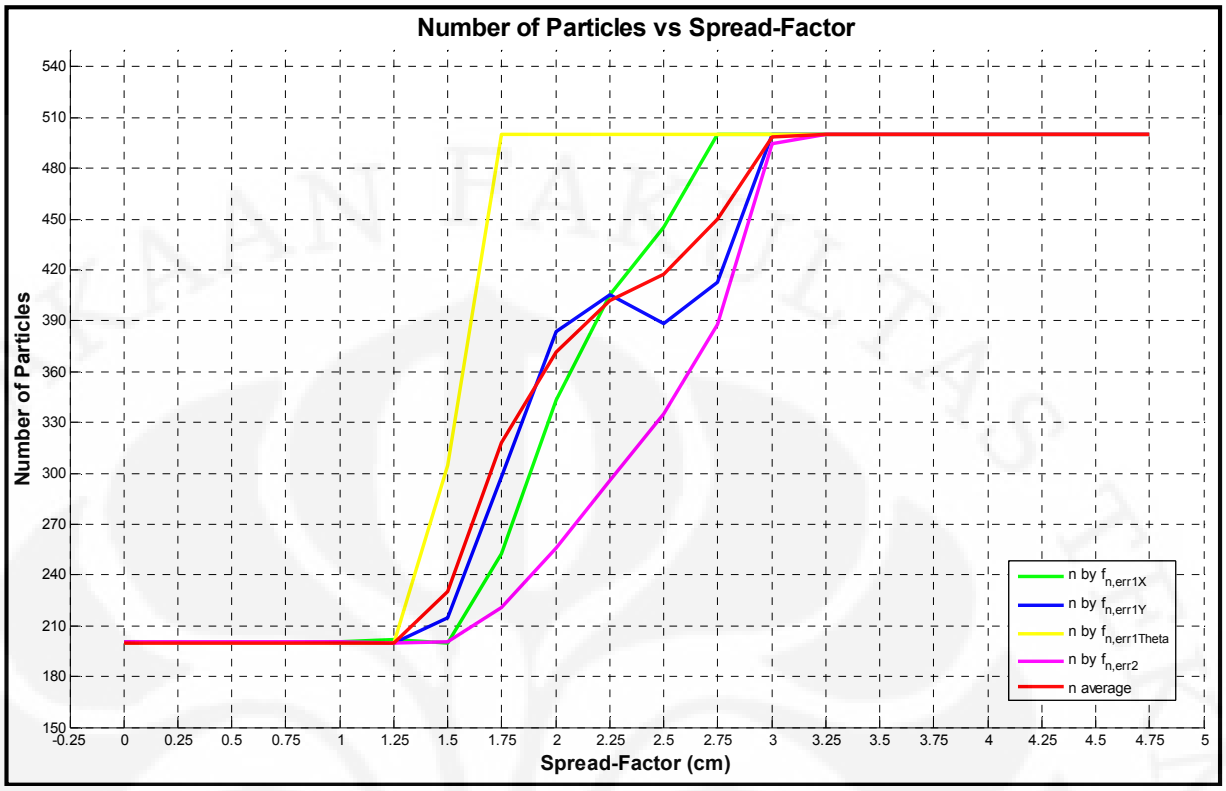
$$n_{err1_x} = f_{n,err1_x}\left(\widehat{err1}_x\right) = \begin{cases} 100\ \widehat{err1}_x + 200 & ; if\ \widehat{err1}_x \le 3\ cm \\ 500 & ; otherwise \end{cases} \quad (7.10a)$$

$$n_{err1_y} = f_{n,err1_y}\left(\widehat{err1}_y\right) = \begin{cases} 100\ \widehat{err1}_y + 200 & ; if\ \widehat{err1}_y \le 3\ cm \\ 500 & ; otherwise \end{cases} \quad (7.10b)$$

$$n_{err1_\theta} = f_{n,err1_\theta}\left(\widehat{err1}_\theta\right) = \begin{cases} 60\ \widehat{err1}_\theta + 200 & ; if\ \widehat{err1}_\theta \le 5^\circ \\ 500 & ; otherwise \end{cases} \quad (7.10b)$$

$$n_{err2} = f_{n,err2}\left(\widehat{err2}\right) = \begin{cases} 30\ \widehat{err2} + 200 & ; if\ \widehat{err2} \le 10\% \\ 500 & ; otherwise \end{cases} \quad (7.11)$$

$$n = average(n_{err1_x}, n_{err1_y}, n_{err1_\theta}, n_{err2}) \quad (7.12)$$



Gambar 7.19 Hubungan Spread-Factor dengan Jumlah Particle pada Dynamic-MCL

7.4.3 Algoritma

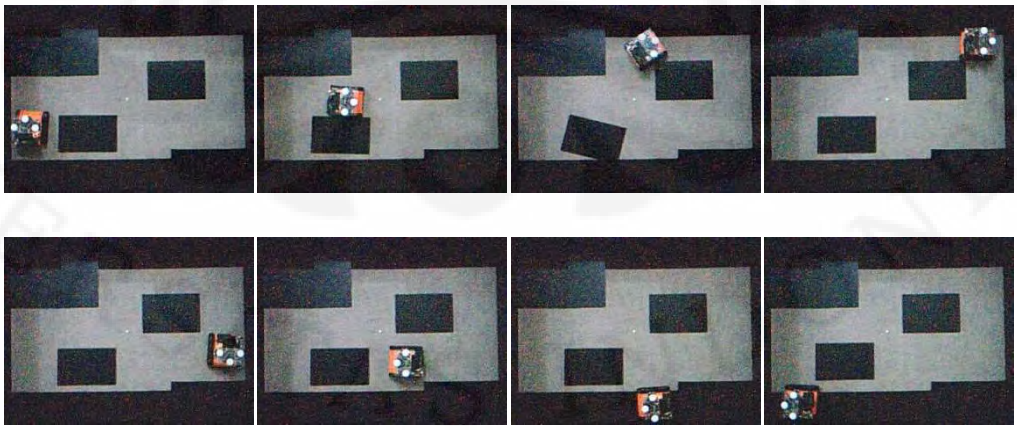
Dasar algoritma Dynamic-MCL, tentu saja, adalah algoritma Plain-MCL. Modifikasi terhadapnya dilakukan sehingga jumlah particle dapat berubah-ubah sebagai karakteristik khas Dynamic-MCL. Lebih lanjut, jumlah particle diubah pada tahap resampling. Hal tersebut adalah keuntungan penggunaan algoritma RSR sebagaimana disebutkan dalam subbab 7.3.1.

Sebagai contoh, jika pada waktu t = 3 dihasilkan 300 particles yang terlalu menyebar, maka sampling pada t = 4 tetap menggunakan 300 particles. Selanjutnya, jumlah particle berubah di tahap resampling t = 4. Oleh karena itu, proses sampling pada t = 5 telah menggunakan jumlah particle yang baru tadi.

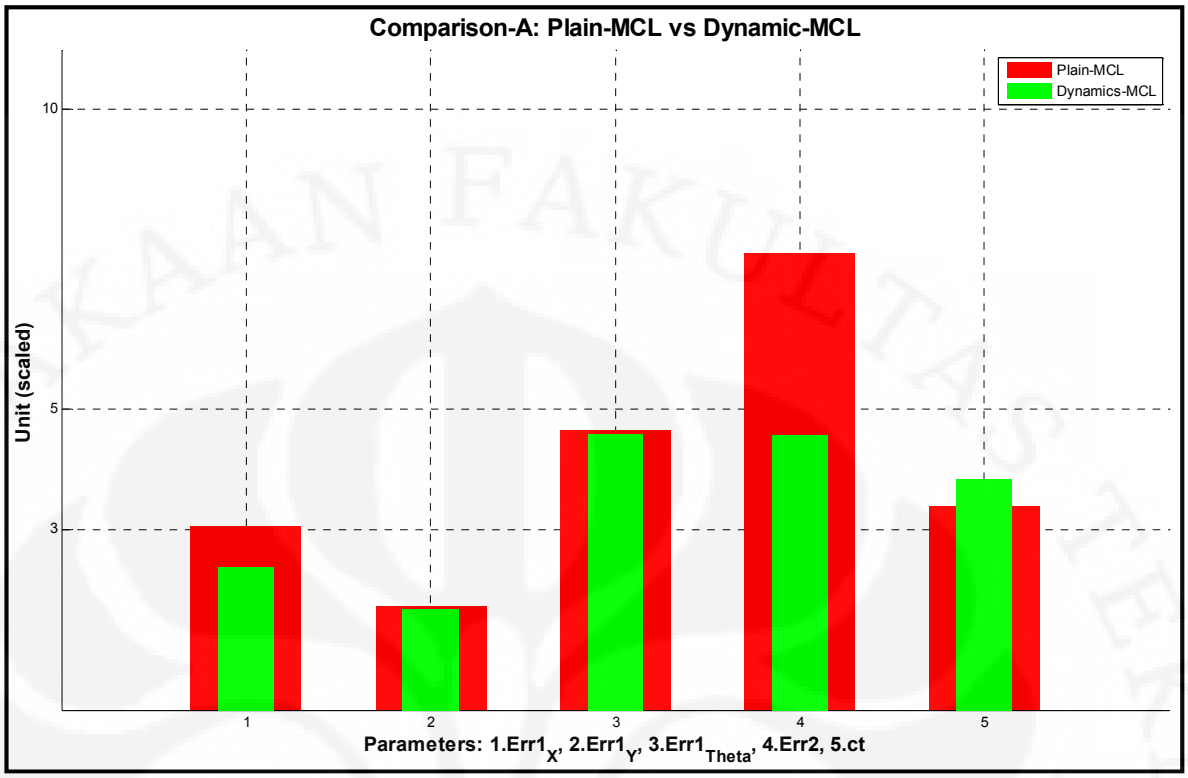
7.5 Analisis Perbandingan Plain-MCL dengan Dynamic-MCL

Pada subbab ini, akan ditunjukkan perbandingan kinerja Plain-MCL (n = 350) dengan Dynamic-MCL (n_{init} = 350) sebagai solusi dari local-localization. Ada dua faktor referensi perbandingan yaitu rata-rata kecepatan komputasi (ct) dan rata-rata kualitas hasil lokalisasi (err1, err2). Selain itu, untuk menguji keduanya secara extensive, tiap algoritma dijalankan pada dua buah lingkungan kerja yang berbeda dengan 37 titik-referensi-pose; keduanya disebut sebagai lingkungan-A dan lingkungan-B, secara berurutan, beberapa titik-referensi-pose pada tiap lingkungan ditunjukkan pada gambar (7.7) dan (7.20). Perbandingan nilai-nilai parameter untuk Plain-MCL dan Dynamic-MCL pada kedua lingkungan ditunjukkan pada gambar (7.21) dan (7.22).

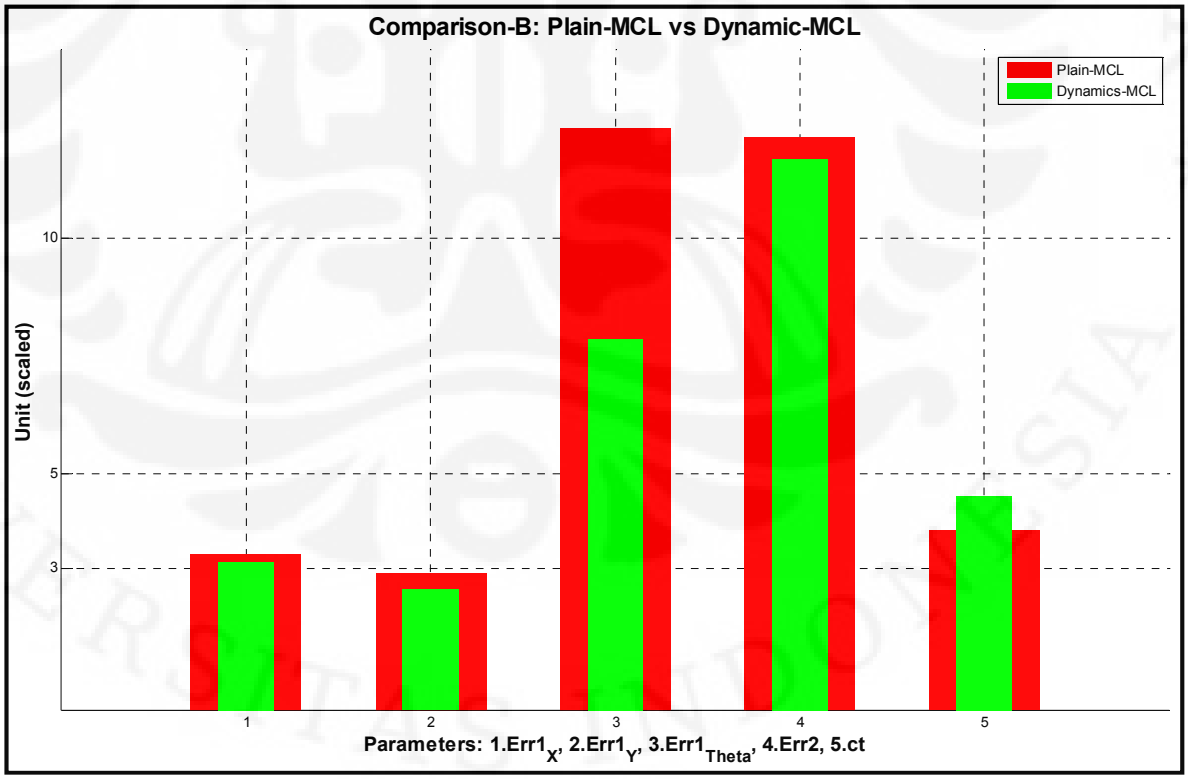
Berdasarkan grafik-grafik parameter uji, dapat diklaim bahwa Dynamic-MCL berkinerja lebih baik daripada Plain-MCL sebagai solusi local-localization. Tampak bahwa semua parameter error Dynamic-MCL (nilai rata-rata) pada kedua lingkungan kerja bernilai lebih rendah daripada milik Plain-MCL. Sementara itu, perbandingan waktu komputasi (nilai rata-rata) relatif berimbang. Akhirnya, ditetapkan bahwa untuk local-localization pada TMR Alfathvrss, Dynamic-MCL dengan jumlah particle awal sebesar 350 particle sebagai solusi teroptimal.



Gambar 7.20 Beberapa titik-referensi-pose pada Lingkungan-B



Gambar 7.21 Perbandingan Plain-MCL dengan Dynamic-MCL pada Lingkungan-A



Gambar 7.22 Perbandingan Plain-MCL dengan Dynamic-MCL pada Lingkungan-B

BAB 8

DYNAMIC-MCL UNTUK LOKALISASI GLOBAL DAN KIDNAPPED-ROBOT

8.1 Pendahuluan

Untuk single-robot, ada tiga modalitas dalam lokalisasi, yaitu (a) pengetahuan robot terhadap pose awalnya: local dan global, (b) sifat lingkungan kerja robot: statis dan dinamis, dan (c) hubungan algoritma lokalisasi dengan yang lain pada sistem kecerdasan robot: active dan passive; penjelasan lengkapnya dapat disimak dalam [3]. Pada riset ini, sebagaimana disebutkan dalam Bab 1, berkaitan dengan modalitas (b) dan (c), maka masalah yang akan dipecahkan adalah lokalisasi pada lingkungan statis dan bersifat pasif. Sementara itu, berkaitan dengan modalitas (a), yang belum ditunjukkan adalah kinerja Dynamic-MCL pada global-localization, di mana robot tidak diberi pengetahuan tentang pose awal.

Lebih lanjut, ada tantangan menarik yaitu kidnapped-robot, yaitu aktivitas memindahkan (menteleportasi) robot secara diam-diam, tanpa diketahui oleh robot; tidak ada informasi kontrol yang merepresentasikan gerakan dari pose sebelum ke sesudah dipindah (diculik). Secara sekilas, barangkali, masalah kidnapped robot dianggap sebagai sebuah khayalan belaka karena jika benar-benar terjadi berarti ada penculik robot. Akan tetapi, dalam perkembangan dunia robotika, masalah itu bisa saja diformalkan, sebagai contoh pada kompetisi Robot-Soccer di mana ada kemungkingan juri memindahkan robot karena melakukan suatu pelanggaran. Sesungguhnya, tantangan ini ditujukan untuk menguji kehandalan suatu algoritma lokalisasi. Seperti disebutkan di awal bahwa kidnapping dilakukan pada waktu robot bekerja, lebih tepatnya beberapa saat setelah inisialisasi sehingga dapat diasumsikan robot telah yakin terhadap posenya. Algoritma yang handal adalah yang dapat melokalisasi dirinya setelah diculik. Untuk itu dibutuhkan beberapa tahap yaitu (a) mengidentifikasi apakah dirinya telah diculik dan (b) melakukan aksi setelah penculikan.

Bab ini pertama-tama membahas bagaimana Dynamic-MCL mengatasi masalah global-localization (subbab 9.2). Kemudian, pada subbab (9.3) ditunjukkan kehandalan Dynamic-MCL dalam menghadapi tantangan kidnapped-robot.

8.2 Global Localization

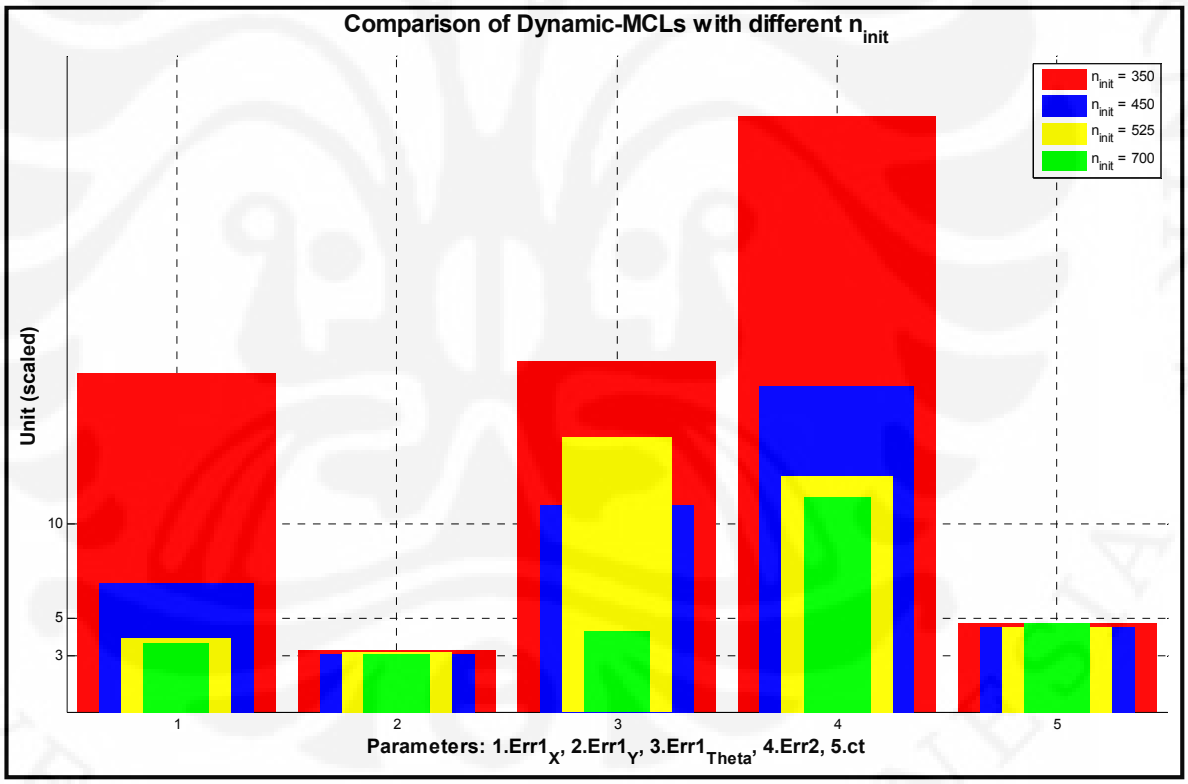
Dalam global-localization, robot tidak tahu tentang pose awalnya. Oleh karena itu, solusi pada Dynamic-MCL adalah dengan menyebar particle-particle secara uniform dan random pada daerah kerjanya saat t = 0 (inisialisasi). Jumlah particle yang disebar pada awal waktu kerja (n_{init}), tentu saja, mempengaruhi kualitas hasil lokalisasi, gambar (8.1). Lagipula, berdasarkan gambar (8.2), (8.3), dan (8.4), standard kualitas akan tercapai lebih cepat dengan menggunakan lebih banyak particle yang disebar pada saat inisialisasi. Tampak bahwa Dynamic-MCL dengan n_{init} = 700 mencapai standard kualitas, $err1_X$ ($\leq$ 3 cm), $err1_Y$ ($\leq$ 3 cm), dan err2 ($\leq$ 10%), lebih cepat daripada dengan n_{init} = 350, 450, 525.

Perlu ditegaskan bahwa, pada gambar (8.2), (8.3), dan (8.4), secara sengaja, tidak semua waktu ditampilkan yaitu hanya sampai t = 15. Langkah tersebut diambil karena Dynamic-MCL, seburuk-buruknya dengan n_{init} = 350, telah berhasil mengatasi global-localization pada waktu t = 10. Dengan demikian, pada waktu-waktu selanjutnya, masalah lokalisasi menjadi local-localization (pose-tracking), dimana kinerja Dynamic-MCL padanya telah ditunjukkan di Bab 7.

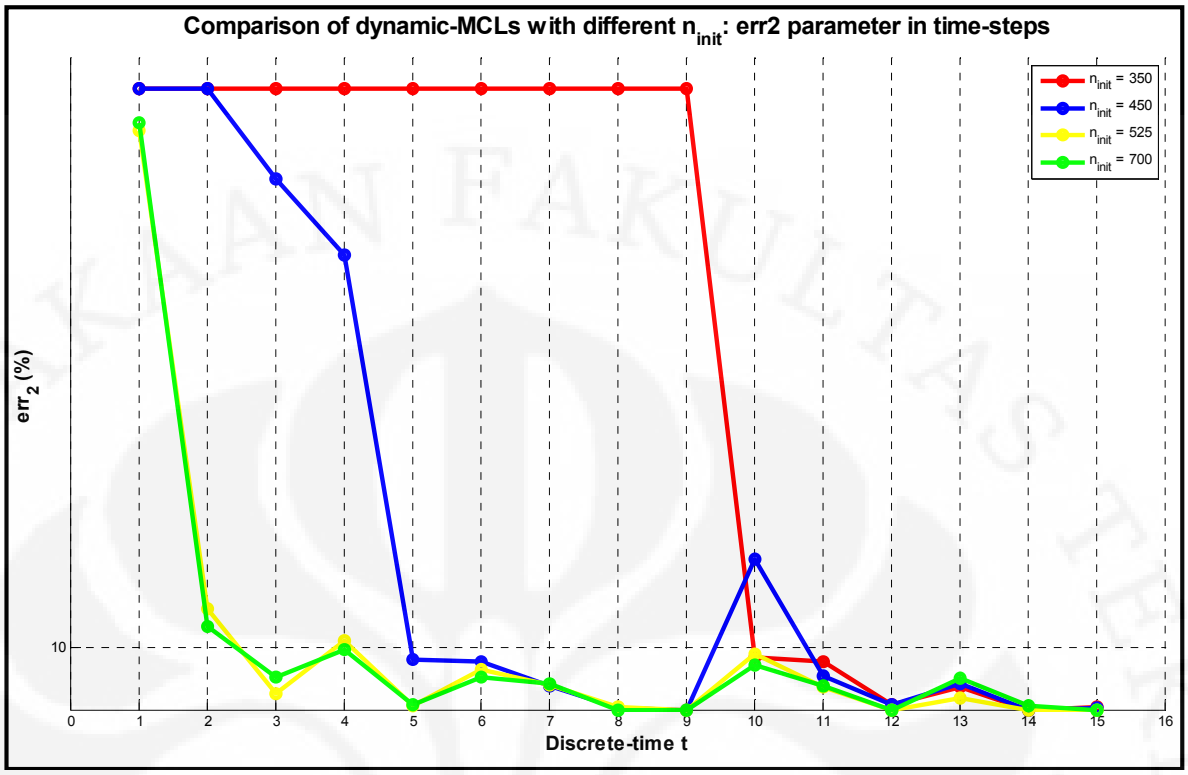
Akhirnya, ditetapkan bahwa untuk global-localization, Dynamic-MCL optimal dengan n_{init} = 700 particles; dua kali n_{init} untuk local-localization. Jumlah tersebut adalah 0.0101% dari semua kemungkinan pose robot di lingkungan eksperimen yaitu bidang lapang berdinding berukuran 160cm x 120 cm dengan resolusi arah-hadap (bearing) sebesar 1˚ sehingga total terdapat 160 x 120 x 360 = 6.912.000 kemungkinan pose. Lebih lanjut, dapat dinyatakan juga bahwa perbedaan n_{init} tidak menyebabkan perbedaan waktu komputasi ct secara signifikan, gambar (8.1)

pada parameter 5. Selain itu, pada t =1, 2, 3, dan seterusnya, Dynamic-MCL tetap menunjukkan ciri khasnya yaitu variasi jumlah particle dari 200 sampai dengan 500 particles, gambar (8.5).

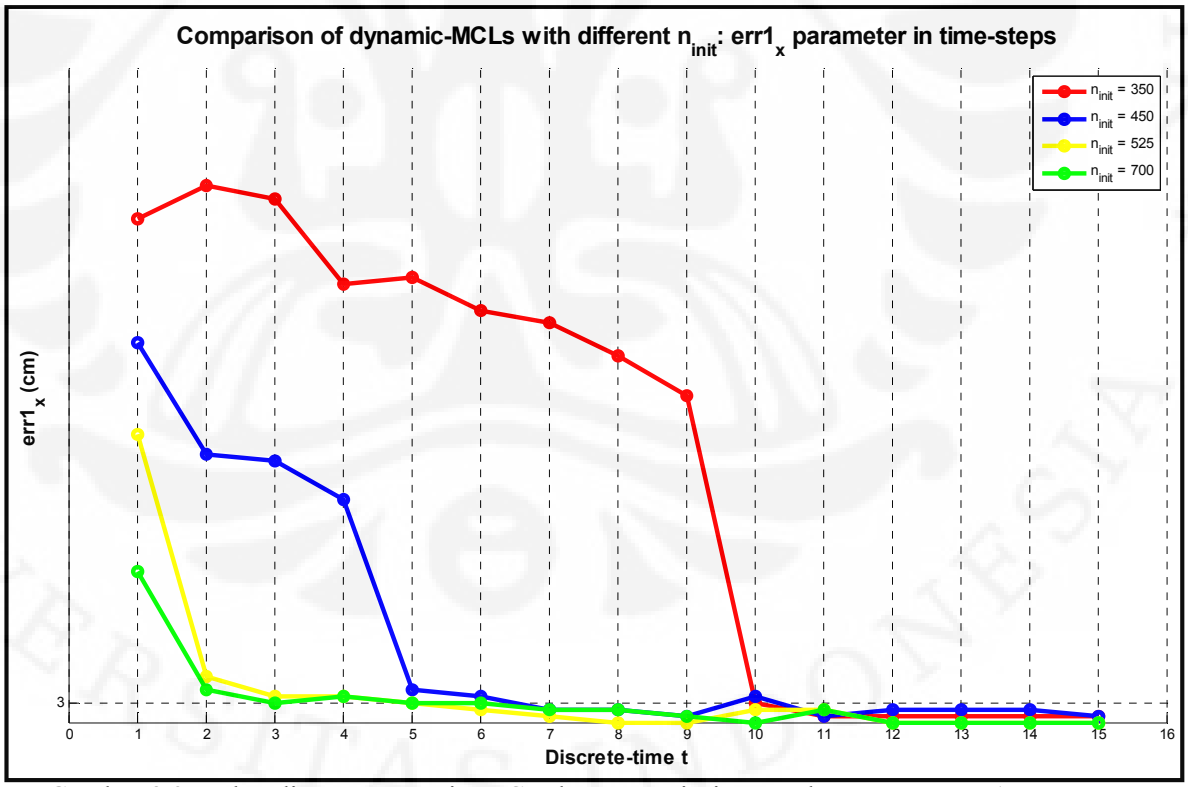
Visualisasi kerja Dynamic-MCL untuk global-localization pada lingkungan-B, seperti pada subbab 7.5, ditunjukkan pada gambar (8.6). Pada gambar itu, lingkaran hijau menandakan pose-absolut robot, lingkaran merah adalah pose simpulan hasil lokalisasi, dan titik-titik kecil berarah adalah particle-particle.



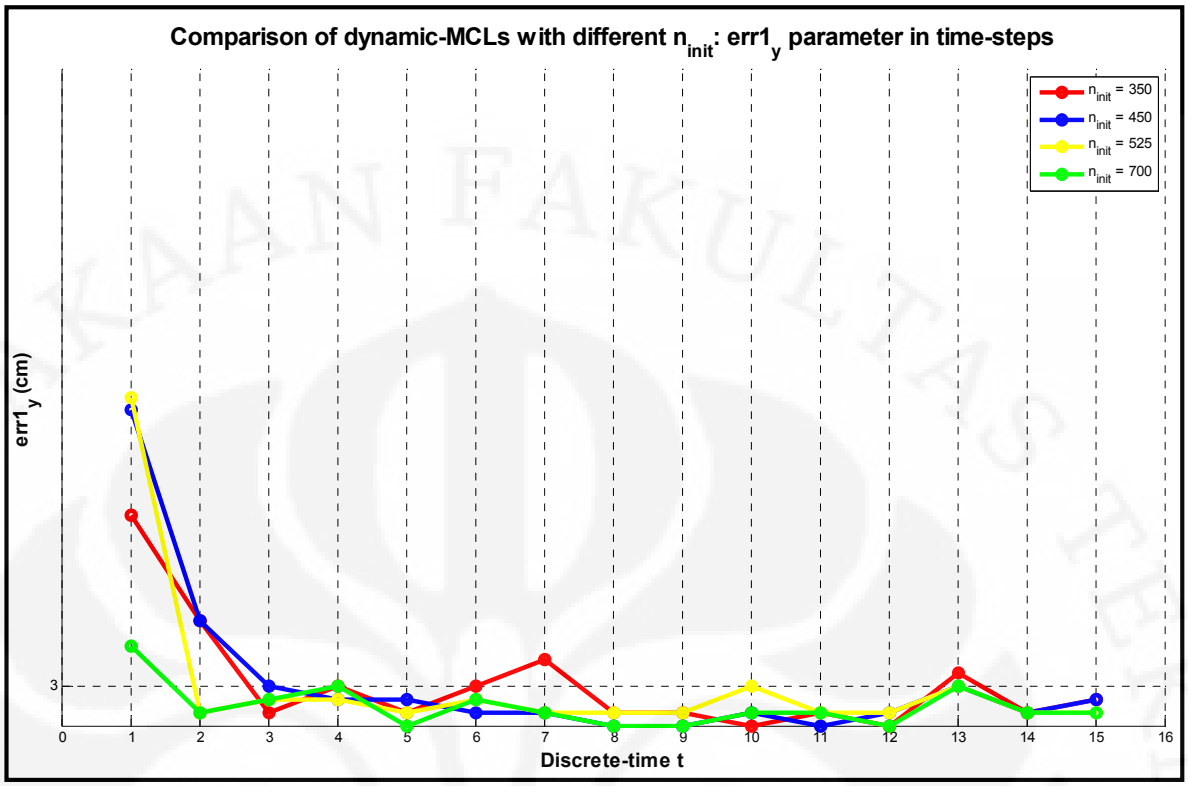
Gambar 8.1 Perbandingan Dynamic-MCL dengan variasi n_{init}



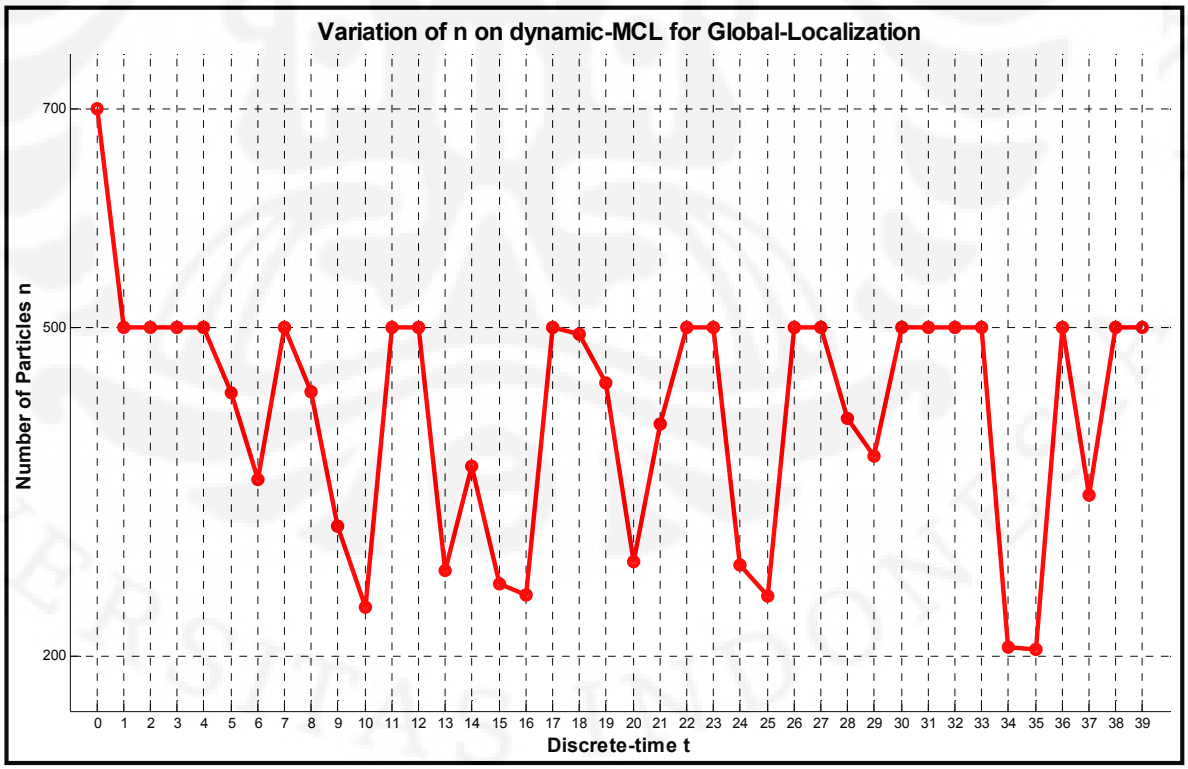
Gambar 8.2 Perbandingan Dynamic-MCL dengan variasi n_{init} pada parameter err_2



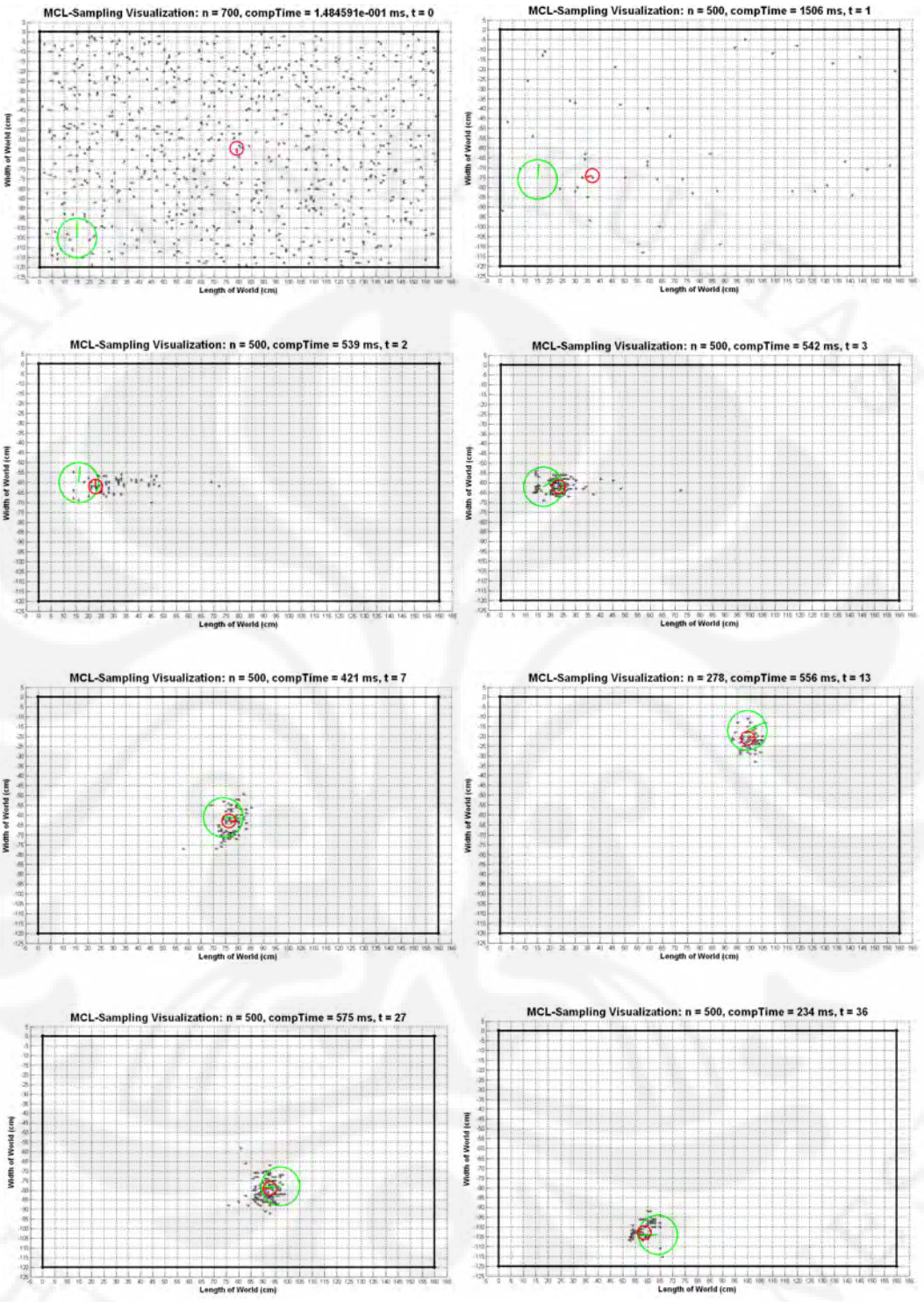
Gambar 8.3 Perbandingan Dynamic-MCL dengan variasi n_{init} pada parameter $err1_X$



Gambar 8.4 Perbandingan Dynamic-MCL dengan variasi n_{init} pada parameter $err1_Y$



Gambar 8.5 Variasi n pada Dynamic-MCL untuk global-localization



Gambar 8.6 Visualisasi kerja Dynamic-MCL untuk global-localization

8.3 Kidnapped-Robot

8.3.1 Ide Dasar

Pada Dynamic-MCL, identifikasi bahwa telah terjadi penculikan dapat dilakukan dengan mengobservasi bobot semua particles yang belum dinormalisasi. Sebagaimana disarankan oleh [23] dan [35], secara garis besar, robot menyatakan dirinya diculik jika nilai maksimal bobot particle w_{max} (unnormalized) lebih kecil daripada suatu batasan tertentu $th_{Kidnapped}$. Solusi tersebut, sekilas, memang tampak sederhana, tetapi penentuan nilai $th_{Kidnapped}$ bukanlah suatu yang trivial. Selain itu, karena sifat ketidakpastian dalam persepsi, timbul resiko terjadinya underestimate maupun overestimate terhadap penculikan. Dengan kata lain, bisa jadi robot menyatakan dirinya diculik padahal tidak jika bobot kecil disebabkan misalnya oleh sensor yang rusak, atau robot belum menyatakan dirinya diculik padahal telah diculik jika nilai bobot masih dalam batas toleransi. Kendati demikan, solusi tersebut akan tetap diambil dengan penentuan nilai $th_{Kidnapped}$ yang hati-hati dengan aturan yang rasional.

Untuk TMR Alfathvrss yang memiliki lima sumber informasi persepsi, nilai $th_{Kidnapped}$ ditentukan dengan aturan sebagai berikut:

- Suatu pembacaan sensor dikatakan gagal (fail) jika nilai probabilitasnya untuk suatu particle lebih kecil dari nilai $p_{CmpsFail}$ untuk kompas-digital dan $p_{SonarFail}$ untuk sonar.
- Probabilitas terkecil yang masih dianggap tidak-gagal untuk kompas-digital adalah nilai probabilitas pada tiga kali standard-deviation-nya sehingga $p_{CmpsFail} = 1.2920e\text{-}004$.
- Probabilitas terkecil yang masih dianggap tidak-gagal untuk sebuah sonar adalah nilai probabilitas pembacaan random sehingga $p_{SonarFail} = 0.0016$.
- Robot dinyatakan diculik (kidnapped) jika minimal terjadi situasi yang dapat direpresentasikan dengan pembacaan sebuah kompas-digital dan dua buah sonar yang nyaris gagal; keadaan tersebut hanyalah salah satu dari

banyak kemungkinan untuk mendeskripsikan situasi minimal di atas. Hal tersebut dapat dinyatakan dengan persamaan (8.1).

- Akhirnya, ditetapkan bahwa $th_{Kidnapped}$ = 9.9999e-11; relatif sedikit lebih kecil daripada hasil persamaan (8.1) untuk menghindari prasangka buruk bahwa telah terjadi penculikan .

$$p_{CmpsFail} \times (p_{SonarFail})^2 = 3.4128e-010 \quad (8.1)$$

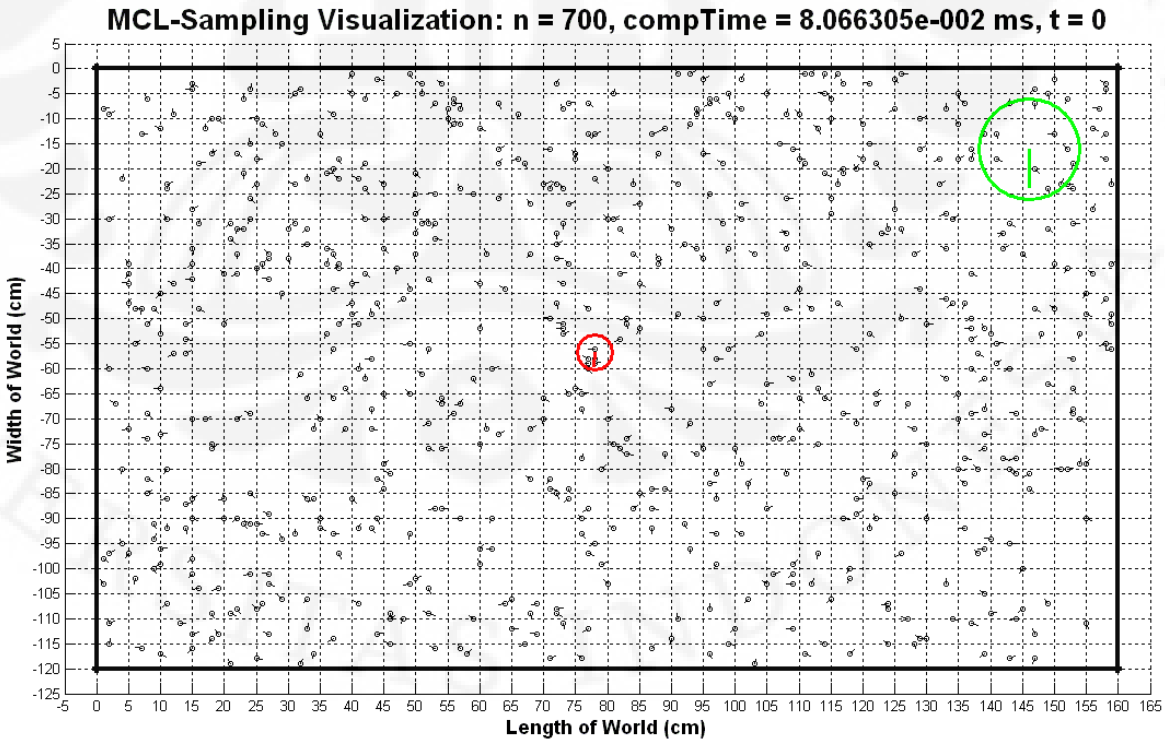
Setelah robot mengetahui bahwa ia telah diculik, aksi yang dilakukan adalah melakukan global-localization. Teknik tersebut diambil karena saat robot telah yakin terhadap posenya, particle-particle terkonsentrasi di sekitar pose-simpulan yang ditunjukkan dengan nilai Spread-factor (S) yang kecil. Dengan demikian, akan susah untuk melokalisasi robot yang pose-absolutnya berada jauh dari konsentrasi particle tanpa menyebarkan kembali particle-particle pada ruang kerja.

8.3.2 Hasil Eksperimen dan Analisis

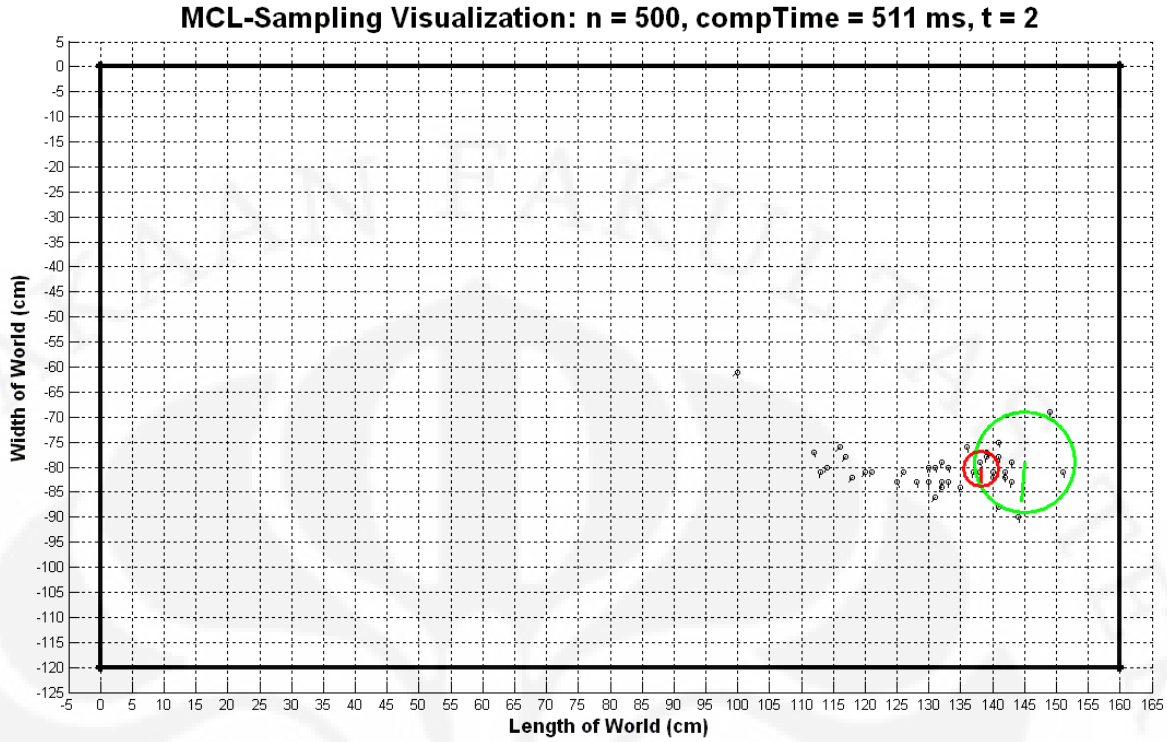
Solusi yang telah diuraikan di atas akan diuji pada sebuah lingkungan kerja robot, diberi nama lingkungan-C, di mana dilakukan skenario penculikan saat robot bekerja. Lebih spesifik, robot memulai dengan global-localization pada t = 0 sehingga pada t = 2, robot mulai yakin terhadap posenya. Selanjutnya, pada t = 11 dilakukan pemindahan pose robot secara diam-diam (penculikan). Pada eksperimen ini terdapat 22 titik-referensi pose. Visualisasi kerja Dynamic-MCL pada beberapa titik-referensi pose ditunjukkan pada gambar (8.4) sampai dengan (8.11), seperti yang lalu, lingkaran hijau adalah pose-absolut, sedangkan yang merah adalah sebuah pose simpulan. Sementara itu, variasi w_{max} (unnormalized) pada t = 1, 2, 3,…, 22 ditunjukkan pada gambar (8.12) dan visualisasi tahap weighting di mana dihasilkan bobot (unnormalized) untuk tiap particle ditunjukkan pada gambar (8.13).

Berdasarkan visualisasi kerja MCL dan tahap weighting padanya, tampak bahwa pada t = 2 robot telah berhasil melokalisasi dirinya. Oleh karena itu, untuk saat-saat selanjutnya, masalah lokalisasi menjadi bersifat lokal (pose-tracking). Pada saat terjadi penculikan, t =11, nilai maksimal bobot (unnormalized) adalah w_{max} = 1.2211e-11 < $th_{kidnaped}$. Oleh karena itu, penculikan terdeteksi oleh robot sehingga ia, sekali lagi, melakukan global-localization. Peristiwa penculikan juga dapat diamati dari variasi nilai w_{max} (unnormalized) pada gambar (8.12), tampak bahwa nilai w_{max} pada t = 11 merupakan nilai terkecil di antara yang lain. Selanjutnya, pada t = 12, terlihat bahwa particle telah kembali mengumpul di sekitar titik pose-absolut. Oleh karena itu, robot melakukan local-localization lagi sampai pada pose akhir, t = 22.

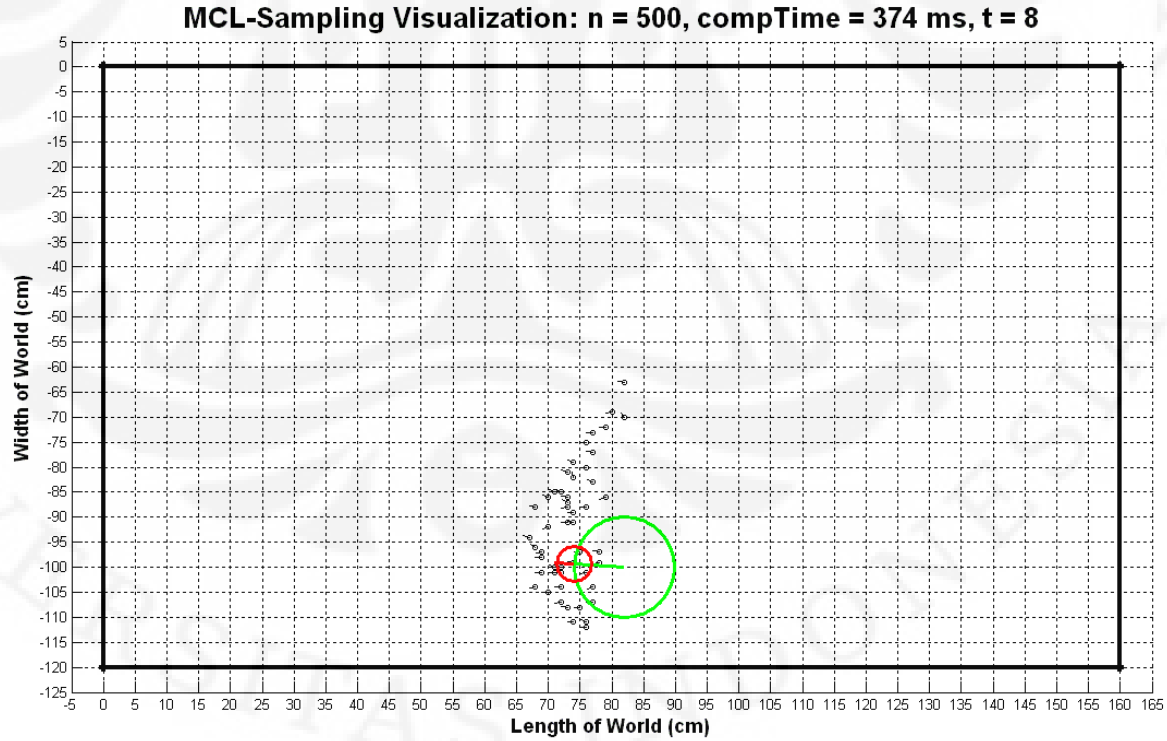
Akhirnya, diklaim bahwa Dynamic-MCL dapat mengatasi tantangan kidnapped-robot, di mana aktivitas pendeteksi penculikan didasarkan pada perbandingan bobot maksimal dengan $th_{Kidnapped}$, dan setelah tahu bahwa diculik robot melakukan global-localization.



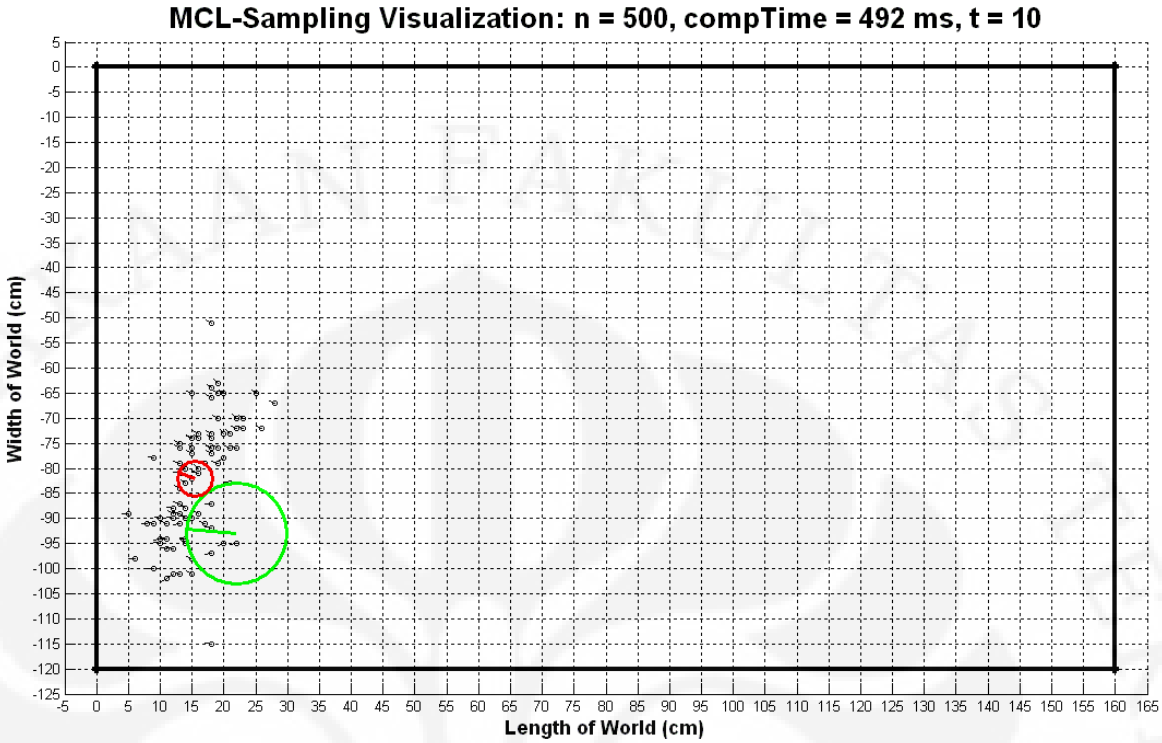
Gambar 8.4 Visualisasi Kerja Dynamic-MCL pada scenario Kidnapped-robot t = 0



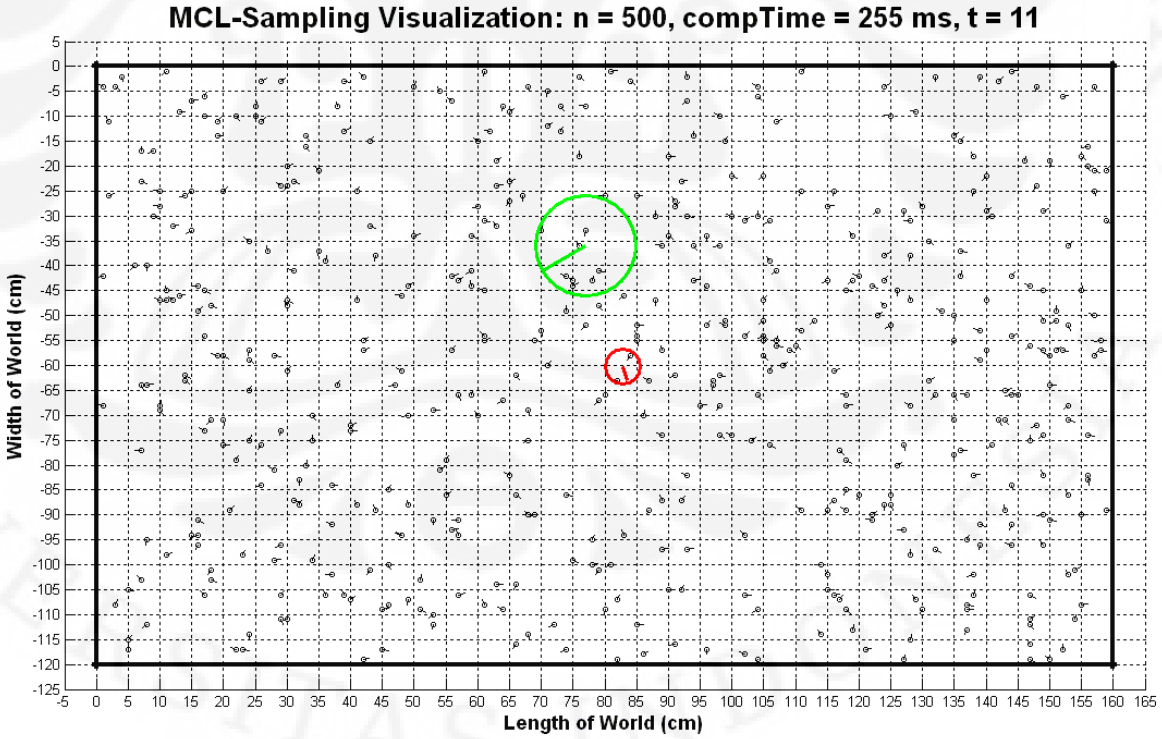
Gambar 8.5 Visualisasi Kerja Dynamic-MCL pada scenario Kidnapped-robot t = 2



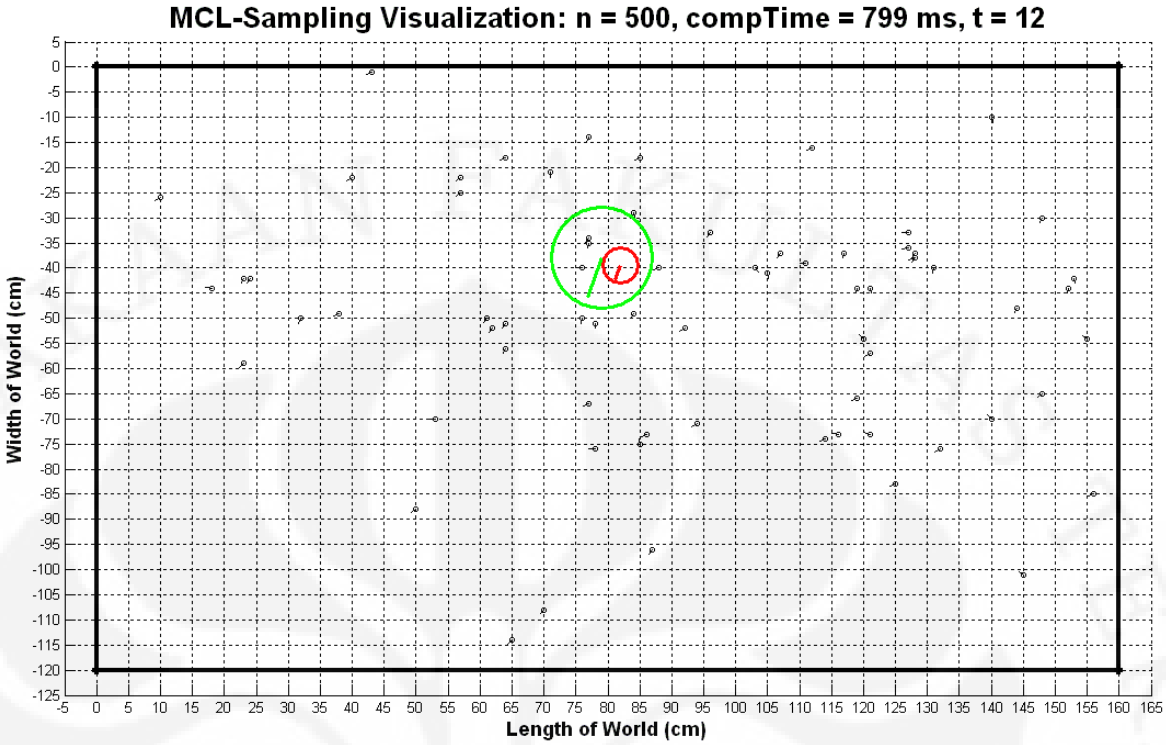
Gambar 8.6 Visualisasi Kerja Dynamic-MCL pada scenario Kidnapped-robot t = 8



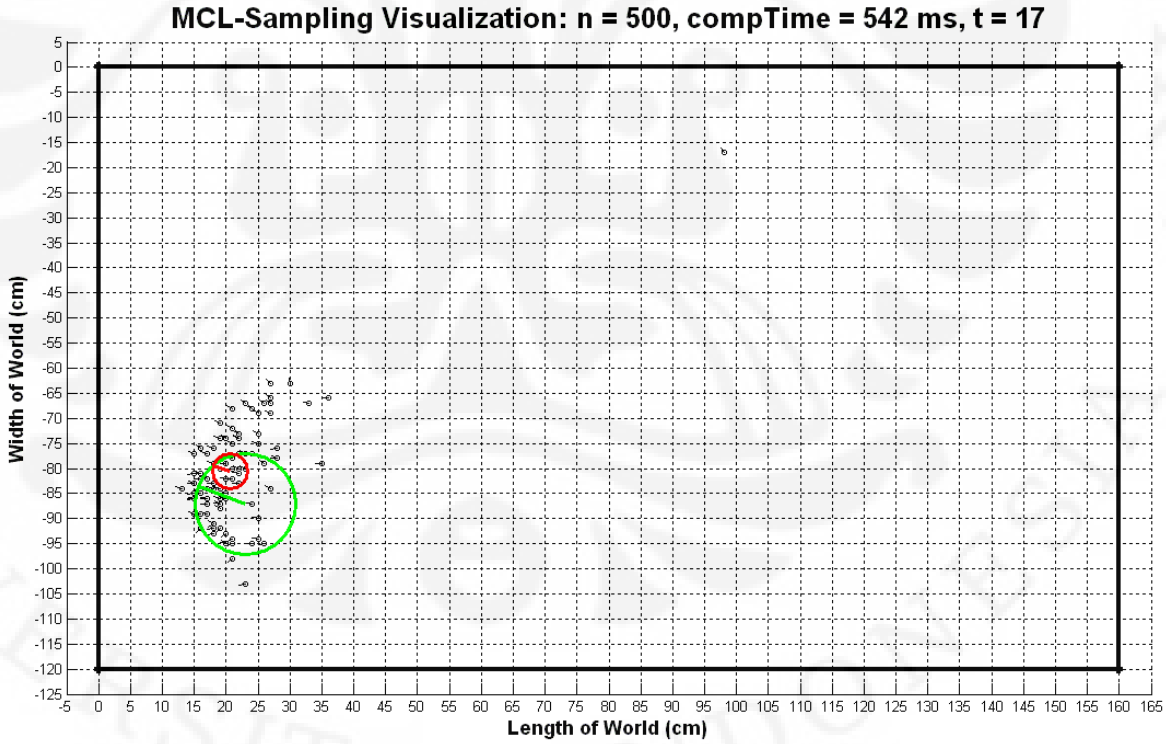
Gambar 8.7 Visualisasi Kerja Dynamic-MCL pada scenario Kidnapped-robot t = 10



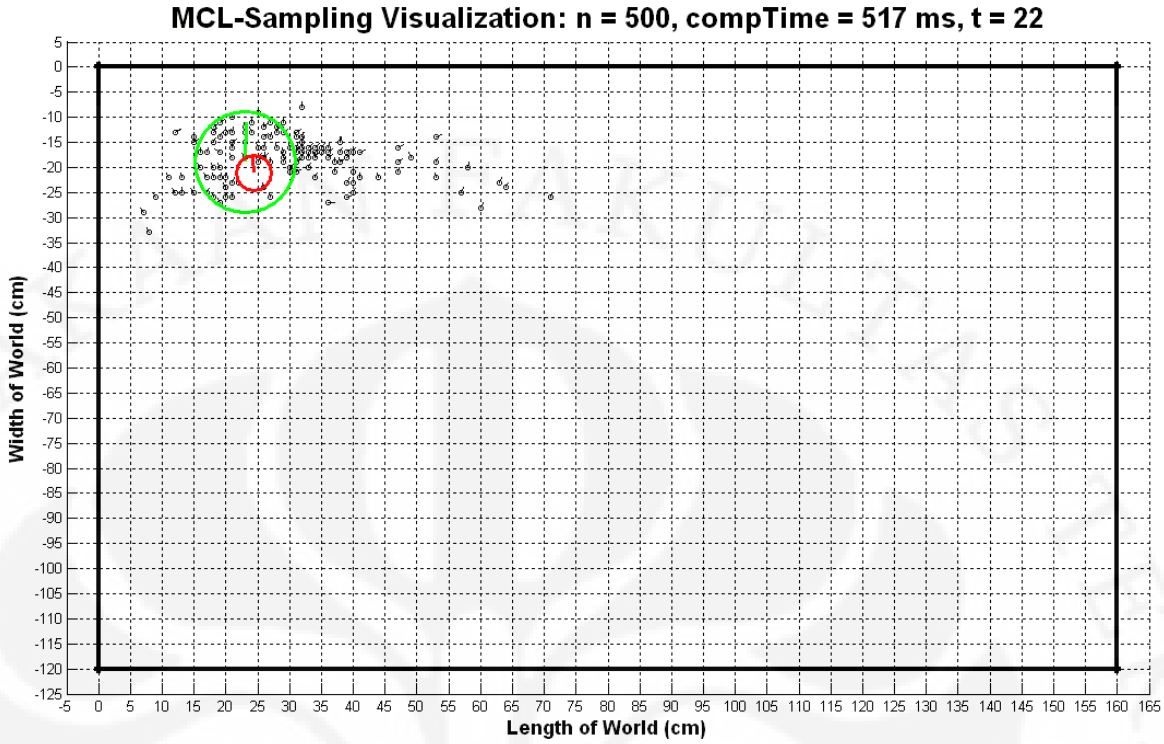
Gambar 8.8 Visualisasi Kerja Dynamic-MCL pada scenario Kidnapped-robot t = 11



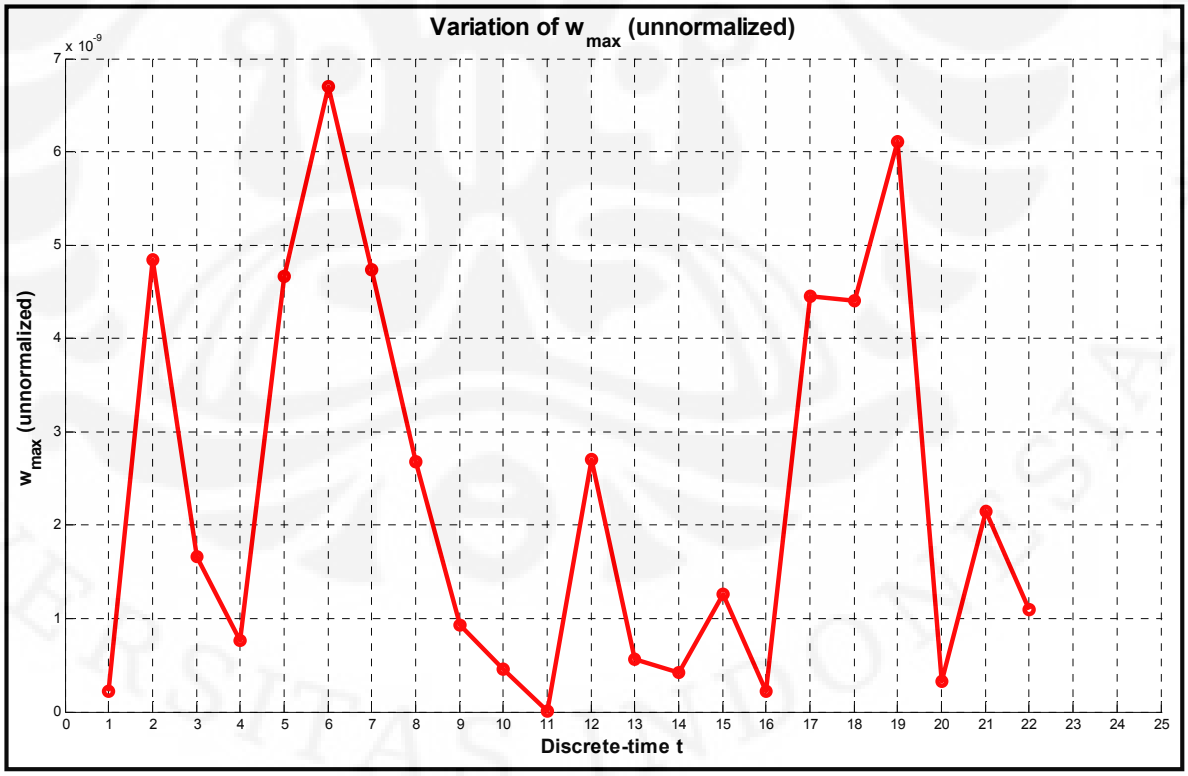
Gambar 8.9 Visualisasi Kerja Dynamic-MCL pada scenario Kidnapped-robot t = 12



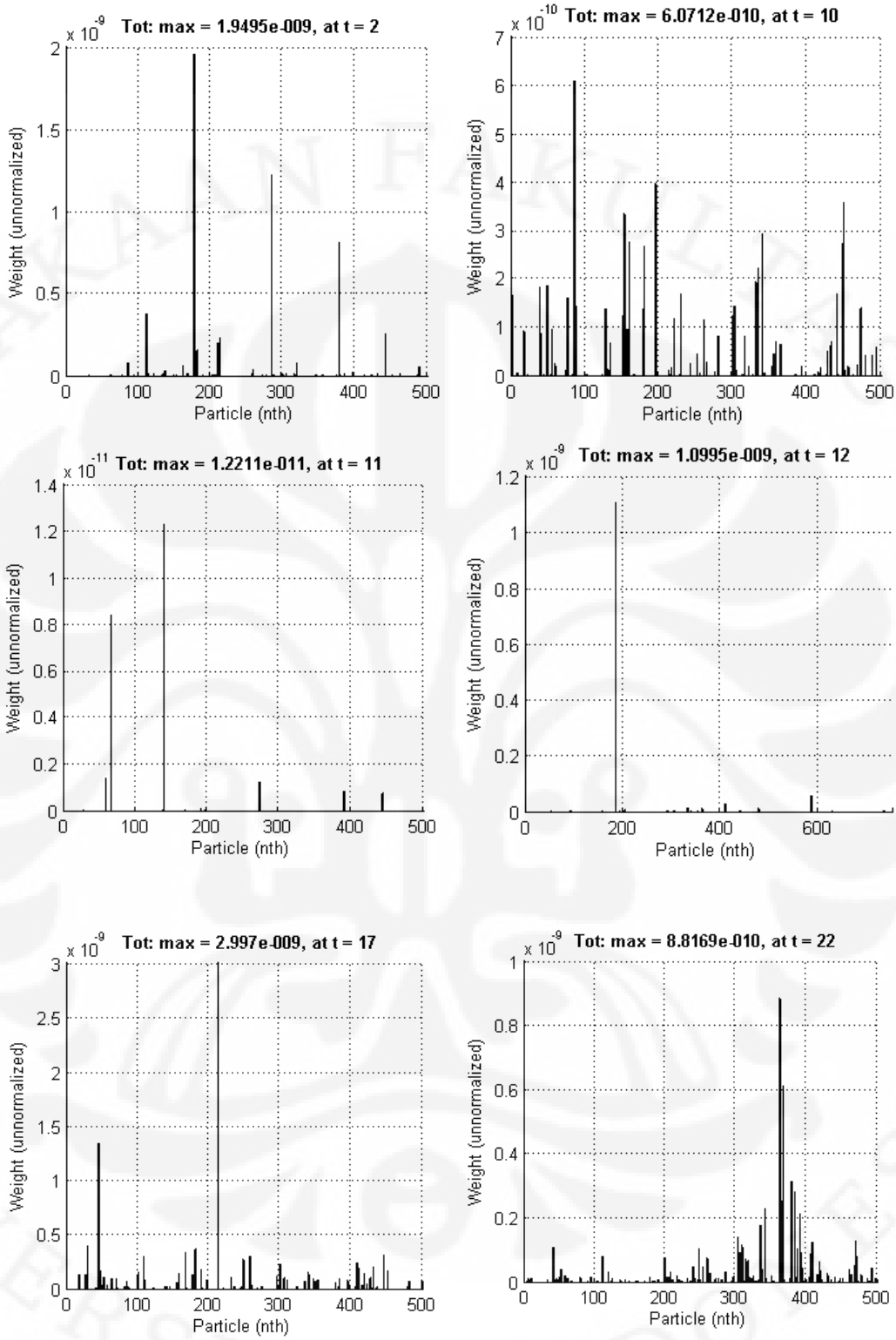
Gambar 8.10 Visualisasi Kerja Dynamic-MCL pada scenario Kidnapped-robot t = 17



Gambar 8.11 Visualisasi Kerja Dynamic-MCL pada scenario Kidnapped-robot t = 22



Gambar 8.12 Variasi nilai w_{max} (unnomarlized)



Gambar 8.13 Visualisasi tahap Weighting Dynamic-MCL pada lingkungan-C

BAB 9

PENUTUP

9.1 Kesimpulan

Pada riset ini, berhasil dikembangkan suatu varian MCL yang dinamakan Dynamic-MCL. Karakteristik khasnya adalah adanya variasi jumlah particle yang dilibatkan berdasarkan Spread-factor (S) yaitu parameter yang mengindikasikan persebaran particle-particle. Lebih lanjut, integrasinya dengan algoritma Plain-MCL dilakukan pada tahap resampling. Berdasarkan eksperimen, terbukti bahwa Dynamic-MCL dapat memecahkan tantangan lokalisasi pada TMR, mulai dari local-localization, global-localization, sampai kidnapped-robot, di mana nilai error posisi: $err1_X \leq 3$ cm dan $err1_Y \leq 3$ cm, nilai error orientasi: $err1_\theta \leq 5^\circ$, serta rasio antara jumlah particle yang berada di luar badan robot terhadap jumlah particle total: $err2 \leq 10$ %. Sementara itu, jumlah particle yang dibutuhkan saat inisialisasi Dynamic-MCL untuk local-localization dan global-localization adalah $n_{init} = 350$ dan $n_{init} = 700$, secara berurutan. Pada kidnapped-robot, identifikasi tentang terjadinya penculikan dilakukan dengan cara membandingkan nilai w_{max} (unnormalized) dengan $th_{kidnapped}$, lalu solusi untuk global-localization dijalankan kembali.

Keberhasilan Dynamic-MCL tidak terlepas dari peranan model gerakan dan persepsi probabilistik dalam tahap sampling dan weighting. Model gerakan probabilistik yang dipakai berjenis Odometry Motion Model dengan pendekatan distribusi Normal with-exclusion. Sementara itu, model persepsi probabilistik dirumuskan dengan asumsi bahwa semua exteroceptive sensor bersifat independent satu sama lain, di mana untuk sonar ada tiga tipe error yang diperhitungkan secara eksplisit dalam model probabilistiknya, yaitu local noise, failures, dan random measurements.

Berkaitan dengan kehandalan model gerakan probabilistik yang terbukti dalam riset ini, dapat disimpulkan pula bahwa odometry dalam TMR yang buruk tetap mempunyai fungsi yang vital dalam melakukan lokalisasi yaitu sebagai pendeskripsi informasi kontrol. Dengan demikian, Dynamic-MCL adalah pelengkap (complement) bagi lokalisasi dengan odometry; bukan pengganti (substitute). Akhirnya, dapat diklaim bahwa implementasi Dynamic-MCL pada jenis robot lain, seperti DMR atau legged-robot, akan mencapai kesuksesan yang serupa dan bahkan lebih bagus.

9.2 Diskusi tentang Implementasi Dynamic-MCL untuk Robot KRCI

Perlu dikabarkan bahwa kemampuan robotika penulis lahir di Kontes Robot Cerdas Indonesia (KRCI). Oleh karena itu, sesungguhnya, motivasi untuk riset ini bermula dari keinginan penulis untuk memecahkan tantangan-tantangan yang sengaja dimunculkan dalam kontes robot paling bergengsi di Indonesia itu. Pada KRCI, tepatnya divisi wheeled, legged, dan expert, tugas utama robot adalah memadamkan api lilin. Lingkungan robot diset menyerupai rumah (home-setting) sehingga mempunyai beberapa ruang berpintu, hanging-object, dan uneven-floor. Tampak bahwa lingkungan kerja robot dirancang untuk benar-benar menyerupai rumah yang lengkap dengan properti-properti yang digantung di dinding oleh pemiliknya serta lantai rumah yang memang didesain untuk tidak selalu datar.

Dari sekian banyak tantangan untuk dapat berkompetisi dengan baik di KRCI, satu yang relevan dengan riset ini adalah navigasi, yaitu bagaimana robot dari pose awalnya bergerak menuju ke lilin (goal). Dalam [2] disebutkan bahwa navigasi tersusun dari empat komponen, yaitu perception, localization, cognition, dan motion control. Lebih lanjut, banyak roboticists yang berpendapat bahwa masalah dalam navigasi bersumber pada kesulitan untuk mempertahankan keyakinan tentang pose robot itu sendiri. Dengan demikian, lokalisasi adalah yang paling fundamental.

Untuk itu, Dynamic-MCL yang dipaparkan dalam riset ini dapat diaplikasikan pada robot yang berlaga di KRCI. Perbedaan mengenai lingkungan kerja robot, dapat diatasi dengan usaha merancang ulang map. Perlu disampaikan kembali, bahwa peta lingkungan yang diberikan ke robot berfungsi dalam tahap weighting, lebih tepatnya pada fungsi ray-casting untuk mendapatkan pembacaan sonar ideal dengan masukan pose robot. Dengan metode ray-casting yang diajukan di riset ini, maka dinding-dinding di arena KRCI perlu dinyatakan dengan persamaan garis dalam map. Objek dalam ruang (misal furniture) bisa saja tidak perlu dipetakan karena model persepsi probabilistik akan mengatasi pembacaan sensor yang kemungkingan bersifat erroneous karenanya. Tentu saja, untuk tidak menabrak objek tadi, robot telah dilengkapi dengan kecerdasan obstacle-avoidance.

Kemudian, berkaitan dengan kemungkinan konfigurasi lapangan yang lebih dari satu, maka untuk pemakaian Dynamic-MCL, semua kemungkinan itu perlu diberikan ke robot (dalam bentuk map). Jadi, dalam running-time-nya robot melakukan banyak Dynamic-MCL; satu untuk tiap map. Langkah itu ditempuh jika belum ditemukan solusi SLAM (Simultaneous Localization And Mapping) yang mumpuni dan praktis (handy). Sesungguhnya, memang algoritma SLAM-lah yang ditujukan untuk robot yang bekerja di lingkungan yang belum diketahui sama sekali (banyak kemungkinan lingkungan adalah salah satu irisan keadaan itu) karena localization mempunyai syarat bahwa peta lingkungan diberikan.

Tentang sumber daya komputasi pada robot KRCI yang biasanya sudah terlalu sibuk karena telah ditujukan untuk berbagai kecerdasan yang lain, dapat disarankan untuk mendedikasikan sebuah microcontroller bagi komputasi Dynamic-MCL, misal sebuah Atmega8 pada 16 MHz. Berdasarkan observasi selama riset ini, dapat diklaim bahwa microcontroller yang MCL-dedicated itu akan dapat melakukan lokalisasi secara periodik selama running-time dengan periode minimal sekitar 300 ms (frekuensi maksimal sekitar 3 Hz). Lagipula, berdasarkan pengamatan penulis sampai dengan KRCI 2010, untuk

menyelesaikan tugas di kontes itu algoritma wall-following sudah cukup sebagai kecerdasan kendali robot dan akhirnya kemampuan lokalisasi hanya diperlukan sesekali saja, misal pada saat wall-following gagal atau kondisi lain yang menyebabkan robot stuck atau hang.

Diakui pula bahwa untuk menggondol predikat juara di KRCI, pada kenyataannya, kemampuan lokalisasi yang handal bisa dikatakan tidak selalu dibutuhkan. Ambil contoh, sebuah robot yang mempunyai kecerdasan kontrol wall-following cukup bagus di mana hanya satu dari sepuluh kejadian robot itu akan menyangkut di dinding (wall-following gagal). Sekalipun kegagalan tersebut terjadi, masih ada peluang untuk retry dan akhirnya robot berhasil memadamkan api. Seandainya robot itu mempunyai kemampuan lokalisasi maka pada saat wall-following gagal, kecerdasan kontrol dapat meminta modul kecerdasan lokalisasi untuk tahu tentang posenya. Kemudian, robot menyimpulkan bagaimana bertindak berdasarkan pengetahuan tentang pose terbaru itu. Namun demikian, supaya robot bernavigasi kembali mencari api setelah error, skenario yang barusan tidak dapat dijamin lebih cepat daripada skenario retry. Selain itu, tentunya, kecerdasan buatan robot menjadi lebih kompleks. Akan tetapi, tetap saja, solusi dengan algoritma self-localization terasa lebih elegant dan berkelas.

9.3 Riset Lebih Lanjut

Sebagaimana ternyatakan dalam uraian sebelumnya, tantangan riset terdekat adalah SLAM. Selain itu, jika masih dalam lingkup localization, ada peluang untuk mengembangkan solusi terhadap active localization, multi-robot localization, sampai pada lokalisasi di lingkungan yang bersifat dinamis dan outdoor. Lebih lanjut, dalam rangka mendukung realisasi service-robot (seperti yang digambarkan dengan baik dalam film “I, Robot”), kecerdasan-kecerdasan lain dalam mobile robotics (the technology of mobility) dan integrasi maupun implementasinya juga menantang untuk diriset, misal sub-problem yang lain dari navigation yaitu path-planning, path-execution, dan motion control.

Daftar Referensi

[1] J. Blankenship, S. Mishal. "Robot Programmer Bonanza". New York: McGrawHill. 2008

[2] R.Siegwart, I.R.Nourbakhsh."Introduction to Autonomous Mobile Robot". Massachusetts: A Bradford Book. 2004

[3] S.Thrun, W.Burgard, and D.Fox. "Probabilistic Robotics". Cambridge, MA: MIT Press. 2005

[4] G.Cen, H.Nakamoto, N.Matsuhira, I.Hagiwara."Effective Application of Monte Carlo Localization for Service Robot". In Proceedings of the 2007 IEEE IROS. San Diego, USA, 2007

[5] Y.Liu, B.Wang, Z.Ding."Fundamental Principles and Applications of Particle Filters". In Proceedings of the 6th World Congress on Intelligent Control and Automation, June 21 - 23, 2006, Dalian, China

[6] J.Gu, M.Meng, A.Cook, P.X.Liu,"Sensor Fusion in Mobile Robot: Some Perspectives". In Proceedings of the 4th World Congress on Intelligent Control and Automation. June 10-14, 2002, Shanghai, China

[7] P.Zingaretti, E.Frontoni."Vision and Sonar Sensor Fusion for Mobile Robot Localization in Aliased Environment". IEEE Xplore.

[8] W.Chung, C.Moon , K.Kim and J.Song. "Design of a sensor model and semi-global localization of a mobile service robot". SICE-ICASE International Joint Conference, Oct.18-21,2006, Busan, Korea

[9] Teddy N. Yap, Jr. and Christian R. Shelton, "Simultaneous Learning of Motion and Sensor Model Parameters for Mobile Robots". IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, May 19-23, 2008

[10] Nicholas Roy and Sebastian Thrun. "Online Self-Calibration For Mobile Robots".

[11] Thomas Collins, J.J. Collins, Conor Ryan." Occupancy Grid Mapping: An Empirical Evaluation". Mediterranean Conference on Control and Automation. July 27-29, 2007, Athens, Greeks

[12] Cody Kwok, Dieter Fox, Marina Meili. "Adaptive Real-time Particle Filters for Robot Localization". Proceedings of the *2003* IEEE Internatiooal Conference on Robotics & Automation, Taipei, Taiwan, September 14-19, 2003

[13] E. Garcia, M.A.Jimenez, P. Gonzalez De Santos, M. Armada, "The evolution of robotics research," IEEE Robotics & Automation Magazine Issue 99 pp.1-14, 2007

[14] M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte, and M. Csorba." A Solution to the Simultaneous Localization and Map Building SLAM Problem". IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 17, NO. 3, JUNE 2001

[15] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp." A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking". IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 50, NO. 2, FEBRUARY 2002

[16] Anthony R. Cassandra Leslie Pack Kaelbling James A. Kurien." Acting under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation". In Proceedings of the IROS 1996

[17] Stergios I. Roumeliotis, and George A. Bekey." Bayesian estimation and Kalman filtering: A unified framework Mobile Robot Localization". Proceedings of the 2000 IEEE International Conference on Robotics & Automation San Francisco, CA April 2OOO

[18] Michael Montemerlo, Sebastian Thrun, Daphne Koller and Ben Wegbreit." FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem"

[19] Davide Scaramuzza." Lecture Notes on Probabilistic Map Based Robot Localization Markov Localization". ETHZ, April 6, 2009

[20] Sebastian Thrun." Is Robotics Going Statistics? The Field of Probabilistic Robotics".

[21] Sebastian Thrun," Probabilistic Algorithms in Robotics". Carnegie Mellon University, April 2000

[22] Dongsheng Wang, Jianchao Zhao, Wei Wang." Particle Filter Based Robust Mobile Robot Localization". Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation August 9 - 12, Changchun, China

[23] Lei Zhang, Ren´e Zapata and Pascal L´epinay," Self-Adaptive Monte Carlo Localization for Mobile Robots Using Range Sensors". The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems October 11-15, 2009 St. Louis, USA

[24] Sebastian Thrun." Particle Filters in Robotics". In Proceedings of Uncertainty in AI (UAI) 2002

[25] Sebastian Thrun, Dieter Fox, Wolfram Burgardy, and Frank Dellaert.” Robust Monte Carlo Localization for Mobile Robots”. Accepted for publication in Artificial Intelligence

[26] Dieter Fox,Wolfram Burgardy, Frank Dellaert, Sebastian Thrun. “Monte Carlo Localization: Efficient Position Estimation for Mobile Robots”. American Association for Artificial Intelligence. 1999

[27] Frank Dellaerty, Dieter Fox, Wolfram Burgard, Sebastian Thrun.” Monte Carlo Localization forMobile Robots”.

[28] Jens-Steffen Gutmann, Wolfram Burgard, Dieter Fox, Kurt Konolige.” An Experimental Comparison of LocalizationMethods”

[29] Armita Kaboli, Michael Bowling and Petr Musilek. “Bayesian Calibration for Monte Carlo Localization”. American Association for Artificial Intelligence.2006

[30] Adam Milstein, Javier Nicolás Sánchez, Evan Tang Williamson,” Robust Global Localization Using Clustered Particle Filtering”. American Association for Artificial Intelligence.2002

[31] Jorg Muller, Axel Rottmann, Leonhard M. Reindl, Wolfram Burgard.” A Probabilistic Sonar Sensor Model for Robust Localization of a Small-size Blimp in Indoor Environments using a Particle Filter”. 2009 IEEE International Conference on Robotics and Automation Kobe International Conference Center Kobe, Japan, May 12-17, 2009

[32] James Carpenter, Peter Clifford, Paul Fearnhead. “An Improved Particle Filter for Non-linear Problems”

[33] A.F.M. Smith and A.E. Gelfand.”Bayesian Statistics Without Tears: A Sampling-Resampling Perspective”. The American Statistician, May 1992, Vol 46, No.2

[34] Tae-Bum Kwon, Ju-Ho Yang, Jae-Bok Song, Woojin Chung.” Efficiency Improvement in Monte Carlo Localization through Topological Information“.Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems October 9 - 15, 2006, Beijing, China

[35] Guanghui Cen , Nobuto Matsuhira, Junko Hirokawa, Hideki Ogawa and Ichiro Hagiwara.” Mobile Robot Global Localization Using Particle Filters”. International Conference on Control, Automation and Systems 2008 Oct. 14-17, 2008 in COEX, Seoul, Korea

[36] Kanji Tanaka, Nobuhiro Okada, Eiji Kondo, Yoshihiko Kimuro.” Probabilistic Localization for Mobile Robots using Incomplete Maps”. Proceedings of the 17th International Conference on Pattern Recognition (ICPR’04)

[37] Stergios I. Roumeliotis. “Robust Mobile Robot Localization From Single-Robot Uncertainties To Multi-Robot Interdependencies”. A Dissertation Presented To The Faculty Of The Graduate School University Of Southern California. May 2000

[38] Dieter Fox, Wolfram Burgard, Hannes Kruppa, Sebastian Thrun. “A Monte Carlo Algorithm for Multi-Robot Localization”. School of Computer Science Carnegie Mellon University. March 1999

[39] Ionnis M. Rekleitis. “A Particle Filter Tutorial for Mobile Robot Localization”. Center for Intelligent Machines, McGill University

[40] Hua-qing MIN, Huan CHEN, Rong-hua LUO.” Active Particle in MCL: An Evolutionary View”. Proceedings of the 2009 IEEE International Conference on Information and Automation June 22 -25, 2009, Zhuhai/Macau, China

[41] Eryong Wu, Zhiyu Xiang, Jilin Liu.” An Efficient Monte Carlo Method for Mobile Robot Localization”. Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation June 25 - 28, 2006, Luoyang, China

[42] Patric Jensfelt.”Approaches to Mobile Robot Localization in Indoor Environments”.Doctoral Thesis Departement of Signals, Sensors and Systems, Royal Institute of Technology. 2001

[43] Sebastian Thrun. “Learning Maps for Indoor Mobile Robot Navigation”. Accepted for Publication in Artificial Intelligence, 1997

[44] Stefan Richter, Michael Faschinger.” Mobile Robot Localization”. October 2003

[45] Sebastian Thrun, Dieter Fox, Wolfram Burgard.” Monte Carlo Localization With Mixture Proposal Distribution”. American Association for Artificial Intelligence.2002

[46] Adam Milstein, Javier Nicolás Sánchez, Evan Tang Williamson, “Robust Global Localization Using Clustered Particle Filtering”. American Association for Artificial Intelligence.2002

[47] Antoni Burguera, Yolanda Gonz´alez and Gabriel Oliver.” Sonar Sensor Models and Their Application to Mobile Robot Localization”. Sensors 2009

[48] Frank Dellaert, Wolfram Burgard, Dieter Fox, Sebastian Thrun. "Using the Condensation Algorithm for Robust Vision-basedMobile Robot Localization". IEEE Xplore 1999

[49] Bruno Siciliano, Oussama Khatib."Springer Handbook of Robotics". Springer-Verlag Berlin Heidelberg 2008

[50] Scott Lenser, Manuela Veloso," Sensor Resetting Localization for Poorly Modelled Mobile Robot".

[51] John J. Leonard." Directed Sonar Sensing for for Mobile Robot Navigation". Department of Engineering Science University of Oxford

[52] Robin R. Murphy."Introduction to AI Robotics". A Bradford Book.2000

[53] Kevin B. Korb, Ann E. Nicholson."Bayesian Artificial Intelligence". A CRC Press Company. 2004

[54] Ivan T Dimov."Monte Carlo Methods for Scientific Applied". World Scientific Publishing. 2008

[55] Nathaniel Fairfield, George Kantor, David Wettergreen."Towards Particle Filter SLAM with Three Dimensional Evidence Grids in a Flooded Subterranean Environment".

[56] Sebastian Thrun, John Langford, Vandi Verma." Risk Sensitive Particle Filters".

[57] Reuven Y. Rubinstein, Dirk P. Kroese." Simulation And The Monte Carlo Method". John Wiley & Sons. 2008

[58] Craig,J, J. Introduction to Robotics: Mechanics And Control. 2nd Ed. 1989. New York: Addison Willey

[59] Jeroen D. Hol, Thomas B. Schon, Fredrik Gustafsson, "On Resampling Algorithms For Particle Filters". Division of Automatic Control Department of Electrical Engineering Link¨oping University

[60] Miodrag Boli´c, PetarM. Djuri´c, Sangjin Hong."Resampling Algorithms for Particle Filters: A Computational Complexity Perspective". EURASIP Journal on Applied Signal Processing 2004:15, 2267–2277.2004

[61] Shao-Hua Hong, Zhi-Guo Shi, Ji-Ming Chen, Kang-Sheng Chen. "A Low-Power Memory-Efficient Resampling Architecture for Particle Filters". Circuits Syst Signal Process (2010) 29: 155–167.2009

[62] Miodrag Boli´,"Architectures for Efficient Implementation of Particle Filters". A Dissertation Presented to The Graduate School Electrical Engineering Stony Brook University. August 2004

[63] V.Dewanto."Odometry Calibration on Tracked Mobile Robot Alfathvrss". December 2009.

[64] Daniel Lowd. "Monte-Carlo Localization: from Tragedy to Triumph!". Spring 2003. Available: http://www.cs.hmc.edu/~dlowd/robotics

[65] The MathWorks."Statistics Toolbox™ 6 User's Guide". 2008

[66] Devantech SRF08 UltraSonic Ranger DataSheet

[67] J.L. Martínez, A. Mandow, J. Morales, A. García-Cerezo and S. Pedraza . Kinematic Modelling of Tracked Vehicles by Experimental Identification. Universidad de Málaga. Proceedings of the 2004 IEEE/RSJ International Conference on ntelligent Robots and Systems (IROS 2004).

[68] Boreinstein, J & Feng, L. UMBmark – A Method for Measuring, Comparing, and Correcting Dead-reckoning Errors in Mobile Robot. 1994. University of Michigan

[69] Datasheet of TraxsterII Brush DC Motor

[70] YS1020U RF Transceiver MANUAL

[71] Adam Milstein."Dynamic Maps in Monte Carlo Localization". Berlin: Springer.

[72] Braunl, T. "Embedded Robotics". 2006. Berlin: Springer.

[73] Arkin, R, C. "Behaviour-Based Robotics". 1998. Massachusetts: A Bradford Book The MIT Press

[74] Thielscher, M. "Reasoning Robot". 2005. Berlin: Springer.

[75] C.M. Bergren."Anatomy of a Robot". 2003.New York: McGraw-Hill

[76] Dieter Fox."KLD-Sampling: Adaptive Particle Filters and Mobile Robot Localization". Technical Report UW-CSE-01-08-02. September 19, 2001