UNIVERSITAS INDONESIA

STRATEGI LOKALISASI *MOBILE ROBOT* DENGAN MENGGUNAKAN *EXTENDED KALMAN FILTER* PADA LINGKUNGAN TERSTRUKTUR

SKRIPSI

RIZKY PRASETYA ADE NUGROHO
0706163672

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN TEKNIK ELEKTRO
DEPOK
2011

UNIVERSITAS INDONESIA

STRATEGI LOKALISASI *MOBILE ROBOT* DENGAN MENGGUNAKAN *EXTENDED KALMAN FILTER* PADA LINGKUNGAN TERSTRUKTUR

Diajukan sebagai salah satu syarat memperoleh gelar sarjana

SKRIPSI

RIZKY PRASETYA ADE NUGROHO
0706163672

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN TEKNIK ELEKTRO
DEPOK
2011

HALAMAN PERNYATAAN ORISINALITAS

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

Nama : Rizky Prasetya Ade Nugroho

NPM : 0706163672

Tanda Tangan : 

Tanggal : 13 Juni 2011

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	:	Rizky Prasetya Ade Nugroho
NPM	:	0706163672
Program Studi	:	Teknik Elektro
Judul Skripsi	:	Strategi Lokalisasi *Mobile Robot* dengan Menggunakan *Extended Kalman Filter* pada Lingkungan Terstruktur

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 : Ir. Wahidin Wahab, M.Sc, PhD. (.....)

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

Ditetapkan di : Depok

Tanggal : 27 Juni 2011

KATA PENGANTAR

Alhamdulillah. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan ridho-Nya penulis dapat menyelesaikan skripsi ini. Penulis menyadari bahwa skripsi ini tidak akan terselesaikan tanpa bantuan dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada Bapak Dr. Abdul Muis, S.T. M.Eng selaku pembimbing yang memberikan arahan, nasihat, dan pinjaman sensor-sensor serta peralatan sehingga penulis dapat menyelesaikan skripsi ini. Terima kasih spesial penulis sampaikan pada Vektor Dewanto atas nasihat-nasihatnya yang sangat berarti dan atas kesediaannya berbagi ilmu dan pengalaman, juga kepada Nur Hidayat atas bantuannya dalam memahami penggunaan mikrokontroler Freescale.

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada Tim Robot UI dan para anggotanya atas ilmu, skill, pengalaman, kebersamaan, dan persaudaraan yang erat. Selanjutnya, penulis mengucapkan terima kasih banyak kepada orang tua dan keluarga penulis yang telah memberikan dorongan dan semangat pada penulis. Penulis juga ingin mengucapkan terima kasih kepada Chairy Wahyu, Ari Nugraheni, Daryanto, Azlul Fadhly Oka, Ade Yurianto dan sahabat-sahabat penulis yang telah memberikan dukungan dan banyak bantuan pada penulis. Khususnya kepada Ade Yurianto dan Azlul Fadhly Oka yang telah membantu dalam pemasangan kamera dan atas kesediaannya menemani penulis saat stress mendera. Ucapan terima kasih juga penulis sampaikan pada teman-teman satu bimbingan, Yudo Dewanto, Faiz, dan Agung Wibisono, atas motivasi dan kebersamaannya dalam menghadapi seminar dan skripsi.

Terima kasih juga penulis sampaikan kepada teman-teman asisten lab kendali atas segala bantuan dan motivasi yang telah diberikan, khususnya kepada Yuddy dan Wicak yang telah membantu dalam banyak perhitungan. Kemudian, penulis mengucapkan terima kasih kepada teman-teman mahasiswa Departemen Teknik Elektro atas persahabatan dan kebersamaan yang telah diberikan, khususnya angkatan 2007. Terakhir, penulis juga ingin berterima kasih pada semua pihak yang telah membantu penulis, baik secara langsung maupun tak langsung yang

belum disebutkan di sini. Penulis berharap semoga Allah SWT membalas semua kebaikan yang telah mereka berikan.

Akhirnya, penulis menyadari bahwa masih banyak kekurangan yang terdapat dalam seminar ini. Penulis mengharapkan kritik dan saran terhadap seminar ini sehingga pada kesempatan mendatang lebih baik.

Depok 13 Juni 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

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

Nama	: Rizky Prasetya Ade Nugroho
NPM	: 0706163672
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 :

STRATEGI LOKALISASI *MOBILE ROBOT* DENGAN MENGGUNAKAN *EXTENDED KALMAN FILTER* PADA LINGKUNGAN TERSTRUKTUR

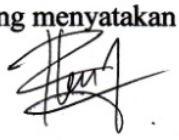
Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini Universitas Indonesia berhak menyimpan, mengalih media/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan seminar saya selama tetap mencantumkan nama saya sebagai penulis/pencipta sebagai pemegang Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 13 Juni 2011

Yang menyatakan



Rizky Prasetya Ade Nugroho

ABSTRAK

Nama : Rizky Prasetya Ade Nugroho
Program Studi : Teknik Elektro
Judul : Strategi Lokalisasi *Mobile robot* Menggunakan *Extended Kalman Filter* pada Lingkungan Terstruktur

Tantangan yang dihadapi *mobile robot* pada operasi *search and rescue* adalah otomatisasi. Dalam mewujudkan *mobile robot* yang benar-benar otomatis, terdapat 3 permasalahan yang perlu dipecahkan. Permasalahan tersebut adalah lokalisasi, pemetaan, dan perencanaan rute. Di antara ketiga permasalahan tersebut, permasalahan paling fundamental yang harus dipecahkan adalah lokalisasi.

Salah satu algoritma yang dapat digunakan untuk melakukan lokalisasi adalah *Extended Kalman Filter* (EKF). Kelebihan algoritma ini antara lain dapat diterapkan pada sistem mikrokontroler 8 bit sekalipun. Pada beberapa penelitian, implementasi algoritma ini membutuhkan banyak sensor. Implementasi algoritma ini pada sistem dengan sumber daya sensor minimal membutuhkan strategi khusus.

Penelitian ini akan menguji performa dua metode yang digunakan untuk implementasi lokalisasi berbasis *Extended Kalman Filter*, yaitu *landmark detection* dan *line extraction*. Implementasi dilakukan dengan menggunakan strategi khusus untuk menyesuaikan dengan keadaan robot yang memiliki sumber daya sensor minimal. Untuk *landmark detection*, strategi yang dilakukan adalah mempartisi dinding area uji, kemudian hasil partisi tersebut dianggap sebagai landmark. Untuk *line extraction*, proses ekstraksi baru dilakukan setelah robot bergerak maju tiga kali dan mendapat tiga titik. Hasil yang didapat menunjukkan bahwa strategi *landmark detection* memiliki performa yang lebih baik daripada strategi *line extraction*, dengan error posisi x dan y dibawah 3 cm dan error orientasi dibawah 5 derajat.

Kata kunci : lokalisasi, *mobile robot*, *Extended Kalman Filter*, *landmark detection*, *line extraction*

ABSTRACT

Name : Rizky Prasetya Ade Nugroho
Study Program : Electrical Engineer

Title : Mobile Robot Localization Strategy Based on Extended Kalman Filter on Structured Environment

A challenge that must be overcome by a mobile robot used in a search-and-rescue operation is automation. To realize truly autonomous mobile robot, there are three problems that need to be solved. Those problems are localization, mapping, and path- planning. Among those three problems, the problem of localization is the most fundamental problem that need to be solved.

One of the algorithm that can be used to localize a mobile robot is Extended Kalman Filter. The advantage of applying Extended Kalman Filter (EKF) for localization is that this algorithm can be implemented even on 8-bit microcontroller based system. On some research, implementation of the EKF needs many sensors. Implementation of this algorithm on a system with minimum sensor resource needs a special strategy.

This research will test the performance of two methods used to implement EKF-based localization, namely landmark detection and line extraction. The method is implemented using a special strategy to cope with the minimal sensor resource provided. To implement landmark detection method, the wall of testing environment is partitioned and then each partition is treated as an individual landmark. To implement the line extraction method, the extraction process is done after the robot moves forward three times and detect three points. The result gotten shows that landmark detection strategy gives better performance than line extraction strategy with the error of x and y position below 3 cm and orientation error below 5 degrees.

Keywords : localization, mobile robot, Extended Kalman Filter, landmark detection, line extraction

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Robotika adalah sesuatu yang telah menjadi impian manusia sejak lama. Sejarah mencatat bidang ilmu robotika telah menjadi perhatian manusia sejak abad ke 3 sebelum masehi. Pada masa tersebut, robot pertama muncul dalam bentuk *automata* yang memiliki bentuk dan ukuran yang menyerupai manusia [1]. Seiring dengan berjalannya waktu, teknologi robot semakin berkembang dan semakin umum ditemui.

Dengan berkembangnya komputer di era modern, bidang robotika pun ikut berkembang. Robot yang dikendalikan oleh komputer menjadi semakin umum ditemui. Salah satu bentuk yang paling umum adalah *manipulator*, suatu jenis robot yang mengimitasi bentuk dan cara kerja tangan manusia. Berkembangnya *manipulator* memicu perkembangan industri menjadi lebih maju, karena *manipulator* dapat diperintah untuk mengerjakan pekerjaan yang repetitif dengan akurasi yang sangat baik. Dengan demikian, produk yang membutuhkan ketelitian tinggi dan melibatkan bagian-bagian yang sangat kecil menjadi mungkin untuk diproduksi. Akan tetapi, *manipulator* ini memiliki kelemahan yang fatal, yaitu kurangnya mobilitas. Pergerakan *manipulator* terbatas pada area di sekitar tempat *manipulator* tersebut diletakkan [2].

Berbeda dengan *manipulator* yang memiliki gerakan terbatas, *mobile robot* memiliki ruang gerak yang lebih bebas. Meskipun demikian, jenis ini membutuhkan mekanisme tambahan agar dapat bergerak dengan bebas. Mekanisme yang dapat diterapkan sangat beragam, mulai dari mekanisme berjalan, melompat, bergeser, sampai dengan mekanisme yang menggunakan roda. Di antara mekanisme tersebut, mekanisme yang menggunakan roda dapat dikatakan sebagai mekanisme yang paling efisien [2] dan memiliki istilah *wheeled mobile robot*.

Karena pergerakannya yang lebih leluasa, *mobile robot* ini memiliki aplikasi yang lebih luas. Aplikasi tersebut antara lain *service robot* (membantu manusia melakukan berbagai pekerjaan domestik) dan *telepresence* (menggantikan seseorang untuk berada pada suatu tempat, misalnya kantor, namun tetap dikendalikan oleh orang yang bersangkutan) [3]. Selain itu, *mobile robot* juga digunakan dalam operasi *search and rescue*.

Karakteristik operasi *search and rescue* yang melelahkan memunculkan kebutuhan akan robot pada operasi tersebut. Robot yang tidak mungkin lelah dapat beroperasi dengan lebih baik dibandingkan manusia. Alasan lainnya adalah, setelah adanya bencana seperti gempa bumi dan gedung runtuh, area pencarian korban akan sangat berbahaya dan tidak memungkinkan untuk dilewati oleh manusia [4]. Pada area semacam inilah, robot diperlukan. Selain itu, pada robot juga dapat dipasang sensor visual seperti kamera sehingga keadaan pada area dapat dipantau meskipun tidak secara langsung.

Tantangan *mobile robot* yang digunakan dalam operasi *search and rescue* adalah otomatisasi. Agar otomatisasi dapat tercapai dalam operasi *search and rescue*, sebuah *mobile robot* dituntut untuk dapat menjawab tiga pertanyaan yang juga merupakan masalah dasar *mobile robot*, yaitu (1) "Where am I?" (2) "Where is my goal?" dan (3) "How do I get there?". Pertanyaan-pertanyaan tersebut merepresentasikan problem (1) Lokalisasi (penentuan posisi robot pada saat tertentu) (2) Pemetaan (*mapping*) dan (3) Perencanaan rute (*path planning*) untuk mencapai sasaran [5].

Di antara tiga permasalahan *mobile robot* untuk dapat mencapai goal, permasalahan lokalisasi adalah permasalahan paling mendasar yang harus dipecahkan. Jika robot mengetahui posisi dan orientasinya pada suatu lingkungan, robot tersebut dapat dikatakan benar-benar bersifat *autonomous* dalam mengerjakan tugasnya [6].

Pemecahan masalah lokalisasi telah banyak dilakukan dengan menggunakan algoritma *Extended Kalman Filter*. Namun, implementasi *Extended Kalman Filter* yang telah dilakukan kebanyakan menggunakan sensor yang berlebih (*redundant*). Dengan demikian, penerapan algoritma

Extended Kalman Filter dengan menggunakan sumber daya sensor yang minim adalah sebuah tantangan yang memerlukan strategi khusus.

1.2 Perumusan Masalah

Masalah pokok dalam riset ini adalah mengimplementasikan algoritma *Extended Kalman Filter* untuk memecahkan problem lokalisasi. Lebih lanjut lagi, riset ini juga bertujuan untuk menguji strategi lokalisasi berbasis *Extended Kalman Filter* yang diterapkan pada *mobile robot* dengan sumber daya sensor yang minim.

1.3 Tujuan Penelitian

Mendesain strategi lokalisasi untuk *mobile robot* dengan menggunakan metode *Extended Kalman Filter* yang dapat diterapkan pada robot dengan sumber daya sensor yang minim

1.4 Pembatasan Masalah

Pada penelitian ini, lingkungan yang digunakan adalah lingkungan terstruktur dengan ketinggian sama di setiap posisi. Lingkungan ini berupa suatu area persegi panjang berukuran 1.5x1.3 meter yang dibatasi dinding di keempat sisinya. Adapun sensor yang digunakan adalah *shaft encoder*, sensor sonar dan kompas digital.

1.5 Metodologi Penelitian

Pada penelitian ini digunakan beberapa metode

a. Studi Literatur, yaitu dengan mencari referensi tentang unsur-unsur pembentuk *mobile robot*, metode-metode lokalisasi *mobile robot*, dan sistem pengujian algoritma lokalisasi
b. Eksperimen, yaitu dengan menerapkan algoritma yang telah didapat pada sistem fisik

1.6 Sistematika Penulisan

a. Bab I akan menjelaskan tentang pendahuluan, yaitu berupa latar belakang, perumusan masalah, tujuan penelitian, pembatasan masalah, metodologi penelitian dan sistematika penulisan
b. Bab II akan menjelaskan tentang komponen *mobile robot*. Bagian ini mencakup penjelasan tentang sensor, aktuator, mikrokontroller, *platform*, *power supply* dan perangkat komunikasi
c. Bab III akan menjelaskan algoritma lokalisasi yang digunakan. Bagian ini mencakup penjelasan tentang prinsip lokalisasi dan algoritma yang sudah diterapkan, dasar teori tentang *Kalman Filter*, dan algoritma lokalisasi yang akan digunakan dalam penelitian
d. Bab IV akan menjelaskan tentang perancangan sistem *mobile robot* yang digunakan pada penelitian
e. Bab V akan menjelaskan perancangan sistem yang digunakan untuk melakukan pengujian, meliputi lingkungan yang digunakan dan sistem validasinya
f. Pada Bab VI akan diberikan skema pengujian, hasil pengujian dan analisanya
g. Pada Bab VII akan diberikan kesimpulan dan saran penelitian serta pengembangan labih lanjut

BAB 2
KOMPONEN *MOBILE ROBOT*

2.1 Sensor

Sebuah *mobile robot*, dan robot pada umumnya, memiliki beberapa unsur pembangun. Pada penelitian ini, penulis membagi-bagi unsur pembangun tersebut menjadi aktuator, sensor, kontroler, *platform*, dan perangkat komunikasi. Interaksi antar unsur inilah yang kemudian mewujudkan *mobile robot* secara utuh.

Tujuan akhir dari penelitian ini adalah merancang sebuah *mobile robot* dengan kemampuan lokalisasi. Oleh karena itu, pada bab ini akan dijelaskan unsur-unsur yang diperlukan untuk merancang sebuah *mobile robot* yang sesuai dengan tujuan tersebut dan cara kerjanya secara umum. Berikutnya akan dijelaskan prinsip kerja masing-masing unsur. Terakhir akan dijelaskan komponen yang digunakan oleh penulis dalam membangun *mobile robot* yang digunakan dalam penelitian.

Sensor adalah unsur *mobile robot* yang berfungsi sebagai sarana bagi *mobile robot* untuk mengetahui keadaan lingkungan [7]. Jenis-jenis sensor ini antara lain sensor jarak, sensor orientasi dan *encoder*.

a. Sensor Jarak

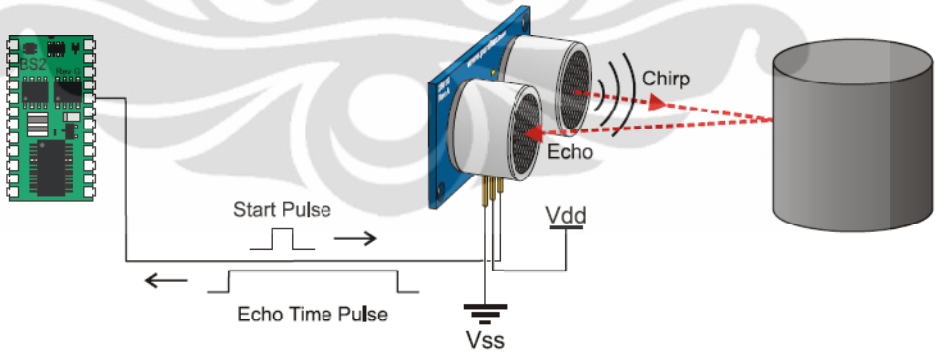
Salah satu prinsip kerja sensor jarak yang umum digunakan adalah prinsip penghitungan *time of flight* [8]. Bagian *transmitter* akan memancarkan suatu bentuk energi ke suatu arah hingga mencapai suatu bidang. Ketika mengenai bidang tersebut, bentuk energi yang dipancarkan sensor akan dipantulkan. Bagian *receiver* akan menerima pantulan tersebut, kemudian sensor akan menghitung jarak antara sensor dengan bidang pantul berdasarkan waktu tempuh (*time of flight*/TOF) energi yang dipancarkan.

Ada beberapa jenis bentuk energi yang dipancarkan sensor untuk mendapatkan data berupa jarak, yaitu gelombang ultrasonik, laser dan sinar infra merah. Pada penelitian ini digunakan sensor berjenis PING produksi Parallax yang memanfaatkan pantulan gelombang ultrasonik

untuk mendeteksi jarak. Bentuk fisik sensor ini dapat dilihat pada gambar 2.1

Gambar 2.1 Bentuk fisik PING [9]

Untuk mengaktifkan dan membaca data jarak dari sensor ini, diperlukan suatu metode khusus [10]. Pertama, untuk mengaktifkan sensor dan memberinya perintah untuk mendeteksi jarak, perlu diberikan pulsa dengan lebar 2 mikrosekon pada pin *signal*. Berikutnya, tunggu hingga sensor memberikan suatu pulsa dengan lebar sinyal tertentu pada pin yang sama. Lebar pulsa ini merepresentasikan jarak yang dideteksi oleh sensor dengan ketentuan lebar pulsa minimum adalah 115 mikrosekon dan lebar pulsa maksimal adalah 18.5 milisekon. Lebar pulsa ini bersesuaian dengan jarak pembacaan 3cm dan 3m [10]. Dengan demikian dapat disimpulkan bahwa lebar pulsa 29.04 merepresentasikan jarak 1 cm. Ilustrasi metode ini dapat dilihat pada gambar 2.2

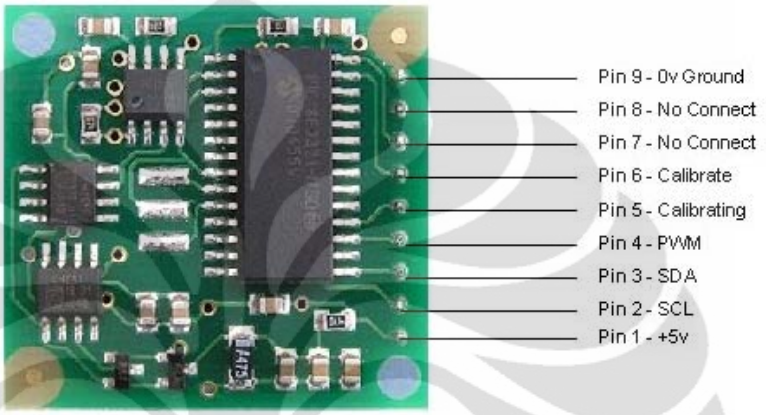


Gambar 2.2 Ilustrasi Metode pembacaan PING [10]

b. Sensor Orientasi

Sensor orientasi yang biasanya digunakan pada *mobile robot* adalah kompas digital. Sensor ini memberikan data berupa orientasi absolut robot dengan cara mendeteksi medan magnet bumi. Pada penelitian ini,

kompas digital yang digunakan adalah modul CMPS03 produksi Devantech [11]. Sensor ini menggunakan IC Philips KMZ51 untuk mendeteksi medan magnet bumi. Sensor ini mendukung pembacaan melalui protokol I2C. Bentuk fisik sensor ini dan konfigurasi pinnya dapat dilihat pada gambar 2.3

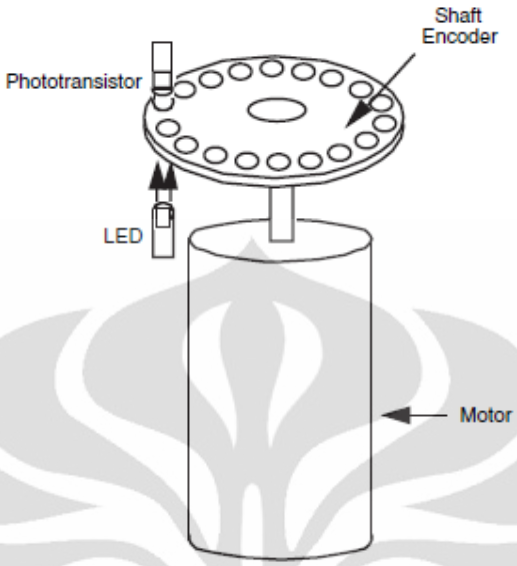


Gambar 2.3 Bentuk fisik dan konfigurasi pin CMPS03 [11]

c. *Encoder*

Encoder adalah sensor yang digunakan untuk menghitung putaran suatu roda. Pada *mobile robot*, data *encoder* ini dapat juga digunakan untuk mengetahui jarak yang telah ditempuh oleh robot dan arah gerak robot.

Encoder ini terdiri atas dua bagian. Bagian pertama adalah sebuah piringan dengan sejumlah lubang di sepanjang kelilingnya. Bagian kedua adalah pasangan *transmitter* dan *receiver* berupa LED atau sumber cahaya lain dan sensor yang peka cahaya. Pasangan *transmitter* dan *receiver* ini ditempatkan sedemikian rupa pada piringan sehingga ada waktu ketika cahaya dari *transmitter* tidak dapat ditangkap oleh *receiver* dan ada waktu ketika cahaya tersebut dapat ditangkap melalui lubang yang ada pada piringan [12]. Ilustrasi susunan *encoder* ini dapat dilihat pada gambar 2.4

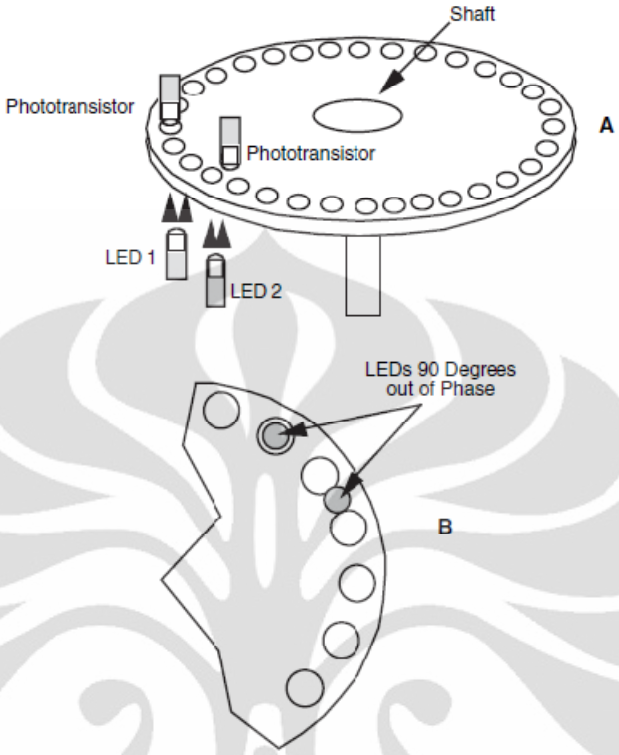


Gambar 2.4 Ilustrasi susunan *encoder* [12]

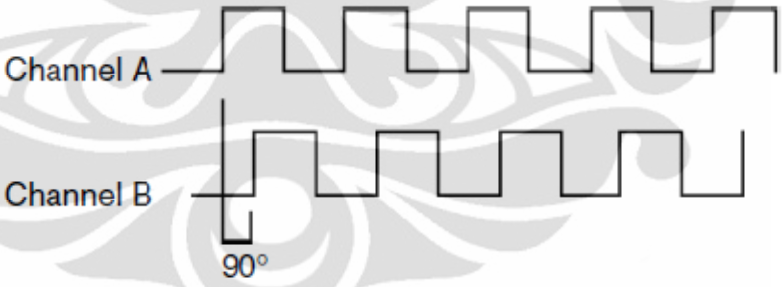
Ketika *receiver* menerima cahaya, maka *receiver* tersebut akan mengeluarkan sinyal dengan logika "1" atau "*high*". Sebaliknya, ketika tidak mendapat cahaya, maka *receiver* tersebut akan mengeluarkan sinyal dengan logika "0" atau "*low*". Dengan demikian, dalam satu putaran, akan diperoleh pulsa dari *encoder* yang berupa sinyal dengan logika "*high*" dan "*low*" yang muncul bergantian. Penghitungan terhadap pulsa yang dihasilkan akan menghasilkan jumlah putaran yang telah dilakukan. Ilustrasi pulsa yang dihasilkan *encoder* ini dapat dilihat pada gambar 2.5

Gambar 2.5 Ilustrasi pulsa yang dihasilkan *encoder* [12]

Jenis lain dari *encoder* ini disebut *quadrature encoder*. Pada *encoder* jenis ini terdapat dua pasang *transmitter-receiver* yang dipasang dengan konfigurasi seperti digambarkan pada gambar 2.6 Dengan konfigurasi ini, *encoder* akan memiliki dua keluaran pulsa yang memiliki beda fasa antara keduanya, seperti dapat dilihat pada gambar 2.7 Dua data ini dapat digunakan untuk menentukan arah pergerakan *mobile robot*.



Gambar 2.6 Konfigurasi Quadrature *Encoder* [12]



Gambar 2.7 Pulsa keluaran quadrature *encoder* [13]

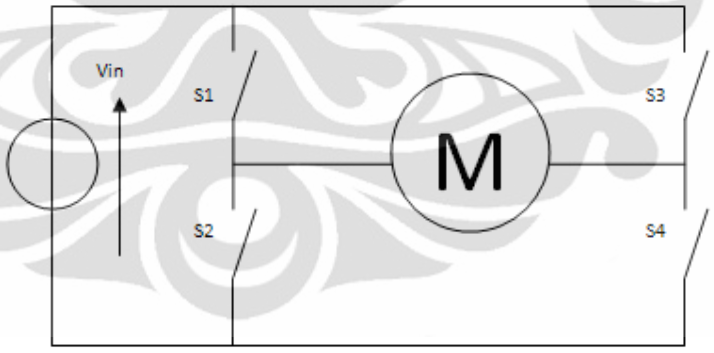
2.2 Aktuator

Aktuator pada *mobile robot* dapat dipandang sebagai kebalikan dari sensor. Jika sensor mendeteksi keadaan lingkungan di sekitar robot dan memberi masukan data ada robot, aktuator memberikan keluaran dari robot bagi lingkungan. Pada penelitian ini, bagian *mobile robot* yang dapat digolongkan sebagai aktuator antara lain motor DC *brushed* sebagai penggerak, dan LCD.

a. Motor DC Brushed

Pada *mobile robot*, motor DC brushed digunakan sebagai penggerak utama. Seperti terlihat pada namanya, motor DC ini adalah jenis motor yang digerakkan dengan menggunakan arus searah/*direct current*. Motor ini memiliki dua komponen yang dapat dikendalikan, yaitu arah gerakan dan kecepatan. Arah gerakan dikendalikan dengan cara mengubah arah arus yang melewati motor ini, sedangkan kecepatan dikendalikan dengan mengubah nilai tegangan yang diaplikasikan pada motor.

Untuk mengendalikan komponen-komponen motor DC secara otomatis, diperlukan adanya suatu cara khusus. Arah gerakan dikendalikan dengan suatu rangkaian yang dinamakan *H-bridge*. Secara sederhana, *H-bridge* dibuat dengan merangkai *switch* dengan konfigurasi yang mirip dengan huruf "H" di sekitar motor, seperti terlihat pada gambar 1.8 Motor bergerak ketika kombinasi *switch* dengan posisi tertentu diaktifkan. Kombinasi *switch* dan pergerakan yang dihasilkan dapat dilihat pada tabel 2.1



Gambar 2.8 Ilustrasi *H-Bridge*

Tabel 2.1 Tabel Kebenaran *H-Bridge*

S1	S2	S3	S4	
1	0	0	1	Maju
0	1	1	0	Mundur
1	1	0	0	Rem
0	0	1	1	Rem

Untuk mengendalikan kecepatan, digunakan metode PWM. Pada metode ini, sinyal persegi (*square wave*) dengan periode dan lebar pulsa tertentu diaplikasikan pada motor. Nilai tegangan rata-rata yang diterima pada motor ditentukan oleh rasio lebar pulsa waktu bernilai "*High*" terhadap periode. Dengan kata lain, mengubah rasio lebar pulsa terhadap periode akan mengubah tegangan yang diberikan pada motor, yang pada akhirnya akan mempengaruhi kecepatan motor.

b. LCD

Pada sistem *mobile robot*, LCD digunakan untuk keperluan *debugging*. Robot dapat diperintahkan untuk menampilkan sesuatu pada LCD, sehingga apa yang dilaukan robot dapat dilacak dan kesalahan program dapat lebih cepat ditemukan.

2.3 Mikrokontroller

Mikrokontroller adalah bagian yang berfungsi untuk mengatur bagian-bagian lain pada *mobile robot*. Pada penelitian ini, ada dua jenis mikrokontroller yang digunakan.

a. ColdFireV1

ColdFireV1 merupakan microcontroller 32 bit produksi Freescale [14]. Mikrokontroller ini memiliki fitur-fitur antara lain:

- *Clock* internal hingga 50MHz
- Protokol komunikasi UART, SPI dan I2C
- *Timer* dengan 8 *channel*
- *General purpose* I/O

b. ATmega16

ATmega16 merupakan mikrokontroller 8 bit produksi Atmel [15]. Microcontroller ini memiliki fitur-fitur antara lain:

- Dua *timer* 8 bit dan 1 *timer* 16 bit
- Pin eksternal untuk *clock timer*
- *External interrupt*
- Protokol komunikasi UART, SPI dan I2C
- *General purpose* I/O

2.4 *Platform*

Unsur *platform* pada *mobile robot* dapat disebut juga *chassis*. Unsur ini mencakup badan robot dan mekanisme roda. Pada penelitian ini digunakan *platform* TraxsterII produksi RoboticsConnection [16] yang memiliki spesifikasi sebagai berikut :

- Panjang : 9 inci = 229 mm
- Lebar : 8 inci = 203 mm
- Tinggi : 3 inci = 76 mm
- Berat : 2.0 lbs = 907.1860 gram
- Ground clearance : 0.75 inci = 19 mm

2.5 *Power supply*

Untuk mewujudkan *mobile robot* dengan pergerakan yang bebas, maka dibutuhkan *power supply* yang compact. Untuk itu, biasanya *power supply mobile robot* menggunakan baterai. Pada penelitian ini digunakan baterai LiPo (Lithium Polymer) dengan tegangan 7.4 V DC.

2.6 Perangkat Komunikasi

Perangkat komunikasi pada *mobile robot* dibutuhkan untuk melakukan pengendalian robot dan akuisisi data. Pada penelitian ini digunakan perangkat wireless USART YS1020U RF Transceiver produksi Hong Kong HuaWei International Electronic Limited. Spesifikasi perangkat ini dapat dilihat pada gambar 2.9.

Power supply: DC 5v atau 3.3V;
Daya RF $\leq$ 10mW;
Arus penerima $\leq$ 25mA;
Arus pengirim $\leq$ 40mA;
Arus keadaan *sleep* $<$ 20uA;
Jarak pengiriman mencapai 500m (BER=10-3@9600bps);
Dimensi: 47mm x 26mm x 10mm (tanpa port antena)

Gambar 2.9 Spesifikasi YS1020U RF Transceiver [17]

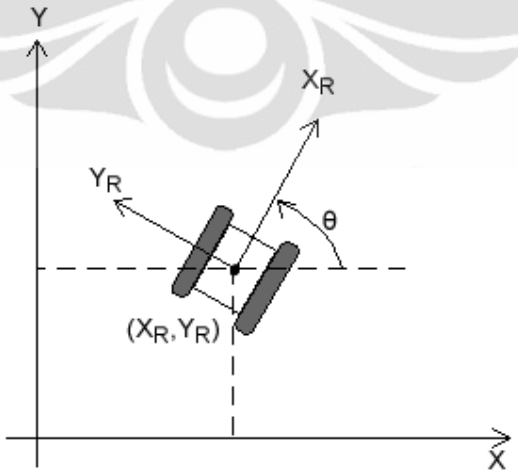
BAB 3
LOKALISASI DENGAN MENGGUNAKAN METODE *EXTENDED KALMAN FILTER*

3.1 Prinsip Lokalisasi *Mobile Robot*

Untuk dapat menentukan posisinya terhadap suatu kerangka referensi tertentu, sebuah *mobile robot* bergantung pada data dari sensor yang merupakan gambaran lingkungan disekitar robot. Data sensor ini kemudian diolah dengan menggunakan suatu algoritma untuk mendapatkan posisi robot di lingkungan tersebut.

Bab ini akan menjelaskan tentang prinsip lokalisasi *mobile robot* secara umum. Selain itu juga akan diberikan contoh beberapa penelitian terdahulu yang telah dilakukan. Pada bab ini juga akan dijelaskan tentang algoritma *Extended Kalman Filter* dan algoritma lokalisasi menggunakan *Extended Kalman Filter*.

Pada dasarnya, lokalisasi adalah permasalahan menentukan posisi dan orientasi robot terhadap suatu kerangka referensi tertentu. Kerangka referensi ini biasanya dibagi menjadi dua jenis, kerangka referensi lokal terhadap titik *origin* robot, dan kerangka referensi global. Dua kerangka referensi ini dapat diilustrasikan dengan gambar 3.1



Gambar 3.1 Ilustrasi kerangka referensi *mobile robot*

Parameter posisi robot lebih lanjut lagi dapat dijabarkan sebagai koordinat robot (x,y) berdasarkan kerangka referensi global, dimana posisi robot ditentukan berdasar titik pusatnya. Parameter posisi dalam koordinat (x,y) ini bersama dengan parameter orientasi lazim didefinisikan sebagai *pose*, p. *Pose* ini memiliki representasi matematis seperti dijelaskan pada (3.1)

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3.1}$$

Dalam melakukan lokalisasi, ada dua metode estimasi posisi yang diterapkan, yaitu *relative positioning* dan *absolute positioning*. *Relative positioning* biasanya dilakukan dengan menggunakan odometri, yaitu perhitungan posisi relatif *mobile robot* berdasarkan pengukuran putaran roda dan/atau *steering angle* [18]. Pada kebanyakan *mobile robot*, odometri dilakukan dengan memanfaatkan data dari sensor *encoder* dan posisi awal robot yang diaplikasikan pada model kinematik robot. Odometri ini mudah dilakukan, sederhana, cenderung murah, dan mudah dilakukan dalam aplikasi *real-time* [18]. Akan tetapi, odometri ini memiliki error yang nilainya cenderung terakumulasi seiring dengan bertambahnya *step* yang dilakukan oleh robot [18]. Oleh karena itu, hasil odometri ini tidak dapat langsung digunakan untuk melakukan *absolute positioning*. Perlu dilakukan metode koreksi pada *absolute positioning*. Cara lain yang dapat dilakukan adalah melakukan kalibrasi pada odometri sehingga error odometri dapat dikurangi.

Metode *absolute positioning* dapat dipandang sebagai koreksi terhadap *relative positioning*. Pada tahap ini dilakukanlah penggabungan dari beberapa sensor, atau lebih dikenal dengan istilah *sensor fusion*. Penggunaan beberapa sensor ini memiliki kelebihan dibandingkan dengan penggunaan satu sensor, antara lain dapat menoleransi kesalahan sensor dan memperbaiki data error yang salah [19]. Metode yang paling sering digunakan untuk melakukan sensor fusion dalam rangka melakukan *absolute positioning* adalah *Kalman Filter* [6] [20] [21] [22]. Keunggulan *Kalman Filter* adalah kemampuannya untuk mengestimasi *state* pada waktu lampau, sekarang, maupun di waktu mendatang, bahkan ketika karakteristik spesifik dari model yang akan diestimasi tidak diketahui [23] Keunggulan lainnya adalah metode ini dapat

diimplementasikan dengan mudah pada *mobile robot*, bahkan pada *mobile robot* dengan kontroler berbasis mikrokontroler 8 bit sekalipun [24]. Pada [6], lokalisasi dilakukan dengan menggunakan EKF (*Extended Kalman Filter*) dan data sensor jarak yang diolah untuk mendapatkan segmen yang sesuai dengan peta. Pada [20], lokalisasi dilakukan dengan menggunakan Augmented EKF, dimana faktor ketidakidealan *mobile robot* digabungkan dengan *pose* robot menjadi satu vektor *state* untuk diestimasi. Pada [21], lokalisasi dilakukan dengan menggunakan EKF dan peta yang berbasis *occupancy grid*. Ippoliti pada [22] menggunakan EKF untuk mengintegrasikan data dari beberapa sensor untuk memecahkan problem lokalisasi.

Melihat kelebihan algoritma *Kalman Filter* dan banyaknya penelitian sebelumnya yang telah menggunakan metode ini, diputuskan bahwa lokalisasi pada penelitian ini akan dilakukan dengan menggunakan algoritma *Kalman Filter*.

3.2 Algoritma *Kalman Filter*

3.2.1 Kalman Filter

Kalman Filter adalah solusi rekursif untuk permasalahan optimasi *linear filtering* data diskrit yang ditemukan oleh R.E. Kalman pada tahun 1960 [25] [26]. Metode Kalman Filter ini dikembangkan berdasarkan formulasi *state*-space sistem linier dinamis [26].

Algoritma Kalman Filter ini digunakan untuk mengestimasi proses linier dinamis seperti yang terlihat pada (3.2)

$$\boldsymbol{x}_k = A\boldsymbol{x}_{k-1} + B\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \quad (3.2)$$

dengan persamaan *measurement* yang diperlihatkan pada (3.3)

$$\boldsymbol{z}_k = H\boldsymbol{x}_k + \boldsymbol{v}_k \quad (3.3)$$

dimana wk dan vk adalah variabel acak yang masing-masing merepresentasikan *process noise* dan *measurement noise*. *Noise* ini diasumsikan merupakan *white noise* dengan kovarians masing-masing Q dan R yang diasumsikan konstan. A adalah matriks yang menghubungkan *state* waktu sebelumnya dengan *state* waktu sekarang. B adalah matriks yang menghubungkan sinyal kendali atau *input*

dengan *state* waktu sekarang. H adalah matriks yang menghubungkan *state* waktu sekarang dengan hasil pengukuran.

Untuk menurunkan algoritma Kalman Filter, pertama definisikan $\hat{\boldsymbol{x}}_k^-$ sebagai *a-priori state estimate* dan $\hat{\boldsymbol{x}}_k$ sebagai *a-posteriori state estimate*. *A-priori state estimate* disini berarti estimasi *state* pada *step* ke- k yang diperoleh dengan menggunakan pengetahuan yang ada sampai sebelum *step* ke- k. *A-posteriori state estimate* adalah koreksi dari *a-priori state estimate* setelah data hasil pengukuran diperoleh. Definisikan juga vektor *state-error a-priori* (e_k^-) dan *a-posteriori* (e_k). Persamaan untuk dua parameter ini ditunjukkan oleh (3.4) dan (3.5)

$$e_k^- = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^- \tag{3.4}$$

$$e_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \tag{3.5}$$

dengan kovarians *error* masing-masing P_k^- dan P_k

Tujuan dari algoritma Kalman Filter adalah mengupdate estimasi *state* yang tidak diketahui dengan menggunakan informasi yang terkandung pada hasil pengukuran, $\mathbf{z}_k$ yang didapat setiap time *step* k. Disini, *estimator* yang diinginkan berjenis *linear estimator*. Dengan demikian, *a-posteriori estimate* dapat direpresentasikan sebagai kombinasi linier antara *a-priori estimate* dan hasil pengukuran [27]. Persamaan yang terbentuk kemudian adalah (3.6)

$$\hat{\boldsymbol{x}}_k = \boldsymbol{K}_k^{(1)}\hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\boldsymbol{z}_k \tag{3.6}$$

dengan $\boldsymbol{K}_k^{(1)}$ dan $\boldsymbol{K}_k$ adalah *gain* yang nilainya akan dicari.

Untuk mencari nilai *gain* $\boldsymbol{K}_k^{(1)}$ dan $\boldsymbol{K}_k$, digunakanlah prinsip ortogonalitas. Prinsip ini menyatakan bahwa pada sebuah *estimator*, nilai estimasi $\hat{\boldsymbol{x}}_k$ akan memiliki nilai *mean square error* minimal jika dan hanya jika (3.7) terpenuhi [28].

$$E[\boldsymbol{e}_k\,\boldsymbol{y}_i^T] = \boldsymbol{0} \quad \text{untuk i = 1,2,3,…k-1} \tag{3.7}$$

Dengan menggunakan (3.3), (3.5), (3.6) dan (3.7) didapat (3.8)

$$E\left[(\boldsymbol{x}_k - \boldsymbol{K}_k^{(1)}\hat{\boldsymbol{x}}_k^- - \boldsymbol{K}_k H\boldsymbol{x}_k - \boldsymbol{K}_k\boldsymbol{w}_{k-1})\boldsymbol{z}_i^T\right] = \boldsymbol{0} \tag{3.8}$$

Karena $\boldsymbol{v}_k$ dan $\boldsymbol{w}_{k-1}$ *uncorrelated*, maka

$$E[\boldsymbol{w}_k\,\boldsymbol{y}_i^T] = \boldsymbol{0}$$

Sehingga (3.8) dapat ditulis kembali menjadi (3.9)

$$E\left[(\boldsymbol{I}-\boldsymbol{K}_k H-\boldsymbol{K}_k^{(1)})\boldsymbol{x}_k\boldsymbol{y}_i^{\boldsymbol{T}}+\boldsymbol{K}_k^{(1)}(\boldsymbol{x}_k-\widehat{\boldsymbol{x}}_k^-)\boldsymbol{z}_i^T\right]=\boldsymbol{0} \quad (3.9)$$

Dari prinsip ortogonalitas, dapat diperoleh hubungan

$$E[(\boldsymbol{x}_k-\widehat{\boldsymbol{x}}_k^-)\boldsymbol{z}_i^T]=\boldsymbol{0} \quad (3.10)$$

Dengan demikian, (3.9) dapat disederhanakan menjadi bentuk (3.11)

$$(\boldsymbol{I}-\boldsymbol{K}_k H-\boldsymbol{K}_k^{(1)})E\left[\boldsymbol{x}_k\boldsymbol{z}_i^T\right]=\boldsymbol{0} \quad (3.11)$$

Untuk sembarang nilai $\boldsymbol{x}_k$ dan $\boldsymbol{y_i^T}$, (3.11) hanya dapat terpenuhi jika berlaku hubungan

$$\boldsymbol{I}-\boldsymbol{K}_k H-\boldsymbol{K}_k^{(1)}=\boldsymbol{0}$$

atau dengan kata lain

$$\boldsymbol{K}_k^{(1)}=\boldsymbol{I}-\boldsymbol{K}_k H \quad (3.12)$$

Substitusi (3.12) ke (3.6) akan menghasilkan representasi lain dari *a-posteriori estimate* seperti tergambar pada (3.13)

$$\widehat{\boldsymbol{x}}_k=\widehat{\boldsymbol{x}}_k^-+\boldsymbol{K}_k(\boldsymbol{z}_k-H\widehat{\boldsymbol{x}}_k^-) \quad (3.13)$$

Persamaan (3.13) adalah inti dari proses estimasi *state* pada algoritma Kalman Filter. Parameter $\boldsymbol{K}_k$ memiliki peran sangat penting untuk mengendalikan proses koreksi estimasi *state* berdasarkan data hasil pengukuran [23]. Parameter ini biasa disebut sebagai Kalman *Gain*.

Persamaan eksplisit untuk menghasilkan $\boldsymbol{K}_k$ akan didiskusikan lebih lanjut. Dari prinsip ortogonalitas dapat diperoleh (3.14)

$$E[(\boldsymbol{x}_k-\widehat{\boldsymbol{x}}_k^-)\widehat{\boldsymbol{z}}_i^T]=\boldsymbol{0} \quad (3.14)$$

dengan $\widehat{\boldsymbol{z}}_i^T$ adalah estimasi dari data hasil pengukuran dan memiliki persamaan seperti terlihat pada (3.15)

$$\widehat{\boldsymbol{z}}_k=H\widehat{\boldsymbol{x}}_k^- \quad (3.15)$$

Definisikan residu atau proses inovasi sebagai (3.15)

$$\tilde{\boldsymbol{z}}_k=\boldsymbol{z}_k-\widehat{\boldsymbol{z}}_k \quad (3.16)$$

Dengan mensubstitusikan (3.15) dan (3.3) ke (3.16), maka didapat (3.17) yang merupakan representasi lain dari (3.16)

$$\tilde{\boldsymbol{z}}_k=\boldsymbol{H}\boldsymbol{e}_k^-+\boldsymbol{v}_k \quad (3.17)$$

Mengurangkan (3.10) dengan (3.14) dan menggunakan definisi (3.16) menghasilkan (3.18)

$$E[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^-)\tilde{\boldsymbol{z}}_i^T] = \mathbf{0} \quad (3.18)$$

Dengan menggunakan persamaan *measurement* pada (3.3) dan persamaan *state estimate update* pada (3.13), maka vektor *state-error* e_k dapat ditulis ulang menjadi (3.19)

$$\boldsymbol{e}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{e}_k^- - \boldsymbol{K}_k\boldsymbol{v}_k \quad (3.19)$$

Mensubstitusi (3.19) dan (3.17) ke (3.18) menghasilkan (3.20)

$$E[\{(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{e}_k^- - \boldsymbol{K}_k\boldsymbol{v}_k\}(\boldsymbol{H}\boldsymbol{e}_k^- + \boldsymbol{v}_k)] = \mathbf{0} \quad (3.20)$$

Karena *measurement noise* $\boldsymbol{v}_k$ independen terhadap *state* dan juga terhadap *error* $\boldsymbol{e}_k^-$, maka (3.20) dapat direduksi menjadi (3.21)

$$(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})E[\boldsymbol{e}_k^-\boldsymbol{e}_k^{-T}]\boldsymbol{H}^T - \boldsymbol{K}_k E[\boldsymbol{v}_k\,\boldsymbol{v}_k{}^T] = \mathbf{0} \quad (3.21)$$

Dengan mendefinisikan $E[\boldsymbol{e}_k^-\boldsymbol{e}_k^{-T}]$ sebagai $\boldsymbol{P}_k^-$ dan memperhatikan sifat kovarians, (3.21) dapat direduksi menjadi (3.22)

$$(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-\boldsymbol{H}^T - \boldsymbol{K}_k\boldsymbol{R} = \mathbf{0} \quad (3.22)$$

Menyelesaikan (3.22) untuk $\boldsymbol{K}_k$ akan menghasilkan persamaan eksplisit untuk $\boldsymbol{K}_k$ pada (3.23)

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T[\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T + \boldsymbol{R}]^{-1} \quad (3.23)$$

Tahap terakhir untuk mendapatkan algoritma Kalman Filter adalah menemukan efek waktu terhadap matriks *error* kovarians. Proses ini disebut *error covariance propagation* dan memiliki dua langkah, yaitu

a. Menemukan perhitungan $\boldsymbol{P}_k$ jika diketahui $\boldsymbol{P}_k^-$
b. Menemukan perhitungan$\boldsymbol{P}_k^-$ jika diketahui $\boldsymbol{P}_{k-1}$

Langkah pertama dilakukan dengan mendefinisikan $\boldsymbol{P}_k$ dengan persamaan (3.24)

$$\boldsymbol{P}_k = E[\boldsymbol{e}_k\boldsymbol{e}_k{}^T] \quad (3.24)$$

Dengan menggunakan persamaan *state error* vector pada (3.19) dan memperhatikan bahwa *process noise* $\boldsymbol{v}_k$ independen terhadap *a-priori estimate error* $\boldsymbol{e}_k^-$, maka dari (3.24) didapat (3.25)

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})^T - \boldsymbol{K}_k\boldsymbol{R}\boldsymbol{K}_k{}^T \quad (3.25)$$

Menggunakan hubungan pada (3.23), (3.25) dapat disederhanakan menjadi (3.26)

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^- \quad (3.26)$$

Persamaan (3.26) merupakan persamaan untuk menghitung $\boldsymbol{P}_k$ yang akan digunakan dalam langkah-langkah algoritma Kalman Filter.

Langkah kedua dilakukan dengan mendefinisikan *a-priori estimate* sebagai fungsi *a-posteriori estimate* pada waktu sebelumnya. Persamaan yang digunakan untuk merepresentasikan hal tersebut dijelaskan pada (3.27)

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} \tag{3.27}$$

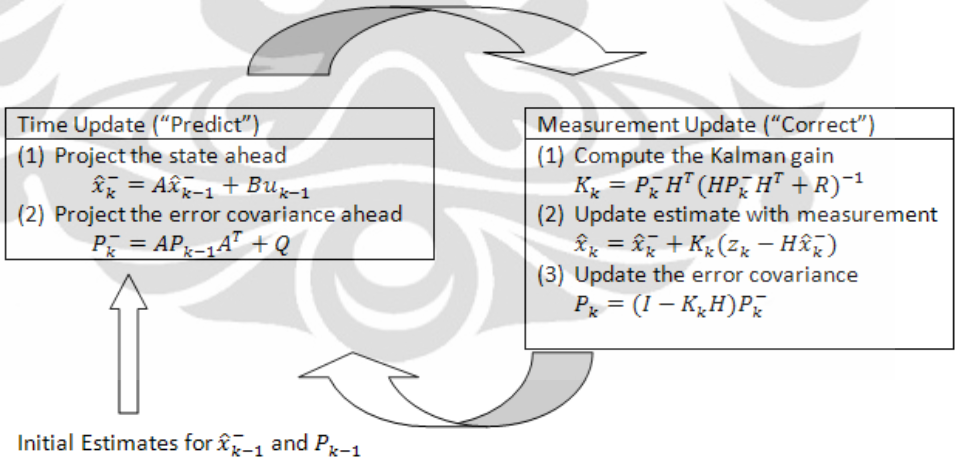
Dengan menggunakan (3.2) dan (3.27), dapat dibuat definisi baru dari *state error vector* yang dijelaskan pada (3.28)

$$\boldsymbol{e}_k^- = \boldsymbol{A}\boldsymbol{e}_k^- + \boldsymbol{w}_{k-1} \tag{3.28}$$

Berikutnya, dengan menggunakan (3.28) pada definisi $\boldsymbol{P}_k^-$, didapatlah (3.29)

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{A}^T + \boldsymbol{Q} \tag{3.29}$$

Setelah persamaan-persamaan yang dibutuhkan telah didapatkan, maka langkah-langkah dalam algoritma Kalman Filter dapat dirumuskan. Langkah-langkah algoritma ini dijelaskan pada gambar 3.1



Gambar 3.2 Representasi visual algoritma Kalman Filter

Pada algoritma Kalman Filter, terutama pada bagian perhitungan Kalman *Gain*, terdapat karakteristik yang perlu diperhatikan. Untuk nilai $\boldsymbol{R}$ yang semakin kecil mendekati nol, maka $\boldsymbol{K}_k$ akan memboboti residu dengan lebih berat. Dapat dikatakan, untuk nilai *measurement error covariance* yang semakin kecil, maka data hasil pengukuran akan semakin dipercaya. Sebaliknya, ketika nilai *a-priori estimate error*

covariance $\boldsymbol{P}_k^-$ semakin kecil, bobot residu akan semakin kecil. Dapat dikatakan, untuk nilai *estimate error covariance* yang semakin kecil, *state* hasil estimasi akan semakin dipercaya [18].

Kekurangan Kalman Filter terletak pada persamaan yang digunakan untuk mendapatkan *a-priori state estimate*. Persamaan tersebut merupakan melibatkan matriks A dan B yang dikalikan dengan *state estimate* waktu sebelumnya dan *input* (*control signal*). Dengan demikian, persamaan ini menjadi linier, sehingga Kalman Filter tidak dapat digunakan untuk mengestimasi sistem nonlinier. Untuk mengestimasi sistem nonlinier diperlukan varian dari Kalman Filter yang disebut dengan *Extended Kalman Filter*.

3.2.2 *Extended Kalman Filter*

Metode *Extended Kalman Filter* digunakan untuk mengestimasi sistem dengan persamaan *difference* yang diperlihatkan pada (3.30)

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{3.30}$$

dengan persamaan *measurement* yang diperlihatkan pada (3.31)

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k, \boldsymbol{v}_k) \tag{3.31}$$

dimana $f(.)$ adalah persamaan nonlinier yang menghubungkan *state* waktu sebelumnya, *process noise* dan *input* dengan *state* waktu sekarang. Pada persamaan *measurement*, $h(.)$ adalah persamaan nonlinier yang menghubungkan *state* sekarang dan *measurement noise* dengan hasil pengukuran. Estimasi untuk mendapatkan *state* $\boldsymbol{x}_k$, dilakukan dengan menggunakan (3.32) dan (3.33)

$$\hat{\boldsymbol{x}}_k^- = f(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{3.32}$$

$$\boldsymbol{z}_{\boldsymbol{k}}^- = h(\hat{\boldsymbol{x}}_k^-, \boldsymbol{v}_k) \tag{3.33}$$

Untuk persamaan eksplisit estimasi *state* pada (3.30), pertama persamaan (3.32) dan (3.33) harus dilinierisasi. Linierisasi ini dilakukan menggunakan deret Taylor di sekitar titik kerja yaitu $\boldsymbol{w}_k = \boldsymbol{0}$ dan $\boldsymbol{v}_k = \boldsymbol{0}$. Asumsi ini digunakan karena nilai *noise* yang terjadi pada proses dan pada pengukuran tidak dapat dihitung. Estimasi dilakukan dengan menganggap *noise-noise* tersebut bernilai nol. Dengan demikian, persamaan sistem berubah menjadi (3.34) dan (3.35)

$$\boldsymbol{x}_k \approx f(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0) + \boldsymbol{F}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0) + \boldsymbol{W}\boldsymbol{w}_k \quad (3.34)$$

$$\boldsymbol{z}_k \approx h(\hat{\boldsymbol{x}}_k^-, 0) + \boldsymbol{H}(\hat{\boldsymbol{x}}_k^-, 0) + \boldsymbol{V}\boldsymbol{v}_k \quad (3.35)$$

dimana

$$\boldsymbol{F}_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0)$$

$$\boldsymbol{W}_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0)$$

$$\boldsymbol{H}_{[i,j]} = \frac{\partial H_{[i]}}{\partial w_{[j]}}(\hat{\boldsymbol{x}}_k^-, 0)$$

$$\boldsymbol{V}_{[i,j]} = \frac{\partial H_{[i]}}{\partial v_{[j]}}(\hat{\boldsymbol{x}}_k^-, 0)$$

Berikutnya dengan mendefinisikan *measurement residual* sebagai (3.36) dan memperhatikan persamaan *a-priori state error*, didapat persamaan *error* proess seperti terlihat pada (3.37) dan (3.38)

$$\hat{e}_{z_k}^- = \boldsymbol{z}_k - \hat{\boldsymbol{z}}_k^- \quad (3.36)$$

$$\hat{e}_{x_k}^- \approx A(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}^-) + \varepsilon_k \quad (3.37)$$

$$e_{z_k}^- \approx \boldsymbol{H}\hat{e}_{x_k}^- + \eta_k \quad (3.38)$$

Pada (3.37) dan (3.38), terdapat dua variabel baru, yaitu ε_k dan η_k. Variabel-variabel ini adalah variabel acak baru yang memiliki *mean* nol dan matriks kovarians masing-masing WQW^T dan VRV^T. Variabel-variabel acak pada (3.36)-(3.38) memiliki karakteristik :

$$p(\hat{e}_{x_k}^-) \sim N(0, E[\hat{e}_{x_k}^- \hat{e}_{x_k}^{-T}])$$

$$p(\varepsilon_k) \sim N(0, E[\boldsymbol{W}\boldsymbol{Q}_k\boldsymbol{W}^T])$$

$$p(\eta_k) \sim N(0, E[\boldsymbol{V}\boldsymbol{R}_k\boldsymbol{V}^T])$$

Selanjutnya, akan dicari estimasi dari $\hat{e}_{x_k}^-$ dengan menggunakan Kalman Filter hipotetis [23]. Hasil estimasi ini dinamakan $\hat{\boldsymbol{e}}_k$ dan akan digunakan untuk mendapatkan *a-posteriori state estimate* berdasarkan (3.39)

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \hat{\boldsymbol{e}}_k \quad (3.39)$$

Dengan memperhatikan karakteristik $\hat{e}_{x_k}^-$, ε_k, dan η_k, men-set nilai prediksi $\hat{\boldsymbol{e}}_k$ menjadi nol, dan mempertimbangkan data $\hat{e}_{z_k}^-$ didapatlah persamaan Kalman Filter hipotetis untuk memperoleh nilai $\hat{\boldsymbol{e}}_k$ pada (3.40)

$$\hat{\boldsymbol{e}}_k = \boldsymbol{K}_k \hat{e}_{z_k}^- \tag{3.40}$$

Mensubstitusikan (3.40) ke (3.48) dan mempertimbangkan persamaan *measurement residual*, didapat (3.41)

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k^-) \tag{3.41}$$

Persamaan (3.45) adalah persamaan *state estimate update* untuk sistem nonlinier.

Persamaan-persamaan yang digunakan dalam algoritma *Extended Kalman Filter* kemudian adalah sesuai dengan yang tertera pada tabel 3.1. Persamaan-persamaan ini diperoleh dari persamaan Kalman Filter dengan beberapa penyesuaian.

Tabel 3.1 Persamaan-persamaan *Extended Kalman Filter*

Time Update ("Prediction") Equations	
$\hat{\boldsymbol{x}}_k^- = f(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1})$	(3.42)
$\boldsymbol{P}_k^- = \boldsymbol{F}\boldsymbol{P}_{k-1}\boldsymbol{F}^T + \boldsymbol{W}\boldsymbol{Q}\boldsymbol{W}^T$	(3.43)
Measurement Update ("Correction") Equations	
$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^T [\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T]^{-1}$	(3.44)
$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_k^-, 0))$	(3.45)
$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-$	(3.46)

3.3 Algoritma Lokalisasi

Untuk membahas algoritma lokalisasi yang digunakan, pertama perlu didefinisikan terlebih dahulu *state* yang akan diestimasi. Sesuai dengan tujuan lokalisasi, maka *state* yang akan diestimasi dengan *Extended Kalman Filter* di sini adalah *pose* dari robot, seperti dapat dilihat pada (3.47)

$$\boldsymbol{x}_{r,k} = p = \begin{bmatrix} x_{r,k} \\ y_{r,k} \\ \theta_{r,k} \end{bmatrix} \tag{3.47}$$

Langkah-langkah yang dilakukan untuk mendapatkan posisi menggunakan langkah yang hampir sama dengan langkah algoritma *Extended Kalman Filter*. Terdapat perubahan kecil terhadap langkah yang digunakan. Persamaan-persamaan yang digunakan dapat dilihat pada (3.48)-(3.52)

$$\hat{\boldsymbol{x}}_{r,k}^- = f(\hat{\boldsymbol{x}}_{r,k-1}, \boldsymbol{u}_{r,k-1}) \tag{3.48}$$

$$\boldsymbol{P}_k^- = \boldsymbol{F}\boldsymbol{P}_k\boldsymbol{F}^T + \boldsymbol{Q} \quad (3.49)$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T[\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T + \boldsymbol{R}]^{-1} \quad (3.50)$$

$$\hat{\boldsymbol{x}}_{r,k} = \hat{\boldsymbol{x}}_{r,k}^- + \boldsymbol{K}_k(\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{r,k}^-, 0)) \quad (3.51)$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^- \quad (3.52)$$

Persamaan *f*(.) yang digunakan pada algoritma lokalisasi ini menggunakan model kinematika robot dan persamaan odometri. Persamaan ini akan dijelaskan lebih lanjut pada bab berikutnya.

Dalam kasus ini, nilai noise $\boldsymbol{w}_k$ dan $\boldsymbol{v}_k$ tidak dapat dihitung, sehingga diabaikan pada proses *state* projection dan *measurement*. Dengan demikian, nilai $\boldsymbol{V}$ dan $\boldsymbol{W}$ juga tidak dapat dihitung, sehingga nilainya diabaikan.

3.4 Correction Step

Correction step pada algoritma lokalisasi berbasis *Extended Kalman Filter* memanfaatkan hasil pembacaan sensor terhadap lingkungan di sekitar robot. Akan tetapi, hasil pembacaan sensor tersebut tidak dapat digunakan begitu saja. Diperlukan langkah pengolahan terhadap data sensor agar dapat digunakan dalam *correction step*. Langkah pengolahan yang dapat dilakukan adalah sebagai berikut

a. *Line Extraction*

Pengolahan menggunakan *line extraction* mentransformasi hasil pembacaan dari sensor jarak menjadi fitur geometris, dalam hal ini garis. Dalam hal ini garis dipilih karena garis merupakan fitur geometris paling sederhana [29]. Pada metode ini, hasil pembacaan jarak yang dilakukan *mobile robot* digunakan untuk menentukan koordinat poin-poin yang terdeteksi dalam kerangka lokal (kerangka referensi *mobile robot*). Dari poin-poin tersebut, kemudian dicarilah parameter-paramter garis seperti jarak dan sudut garis terhadap *mobile robot*. Berikutnya segmen garis yang terdeteksi dicocokkan dengan garis pada peta, untuk menentukan garis mana yang terdeteksi pada kerangka global.

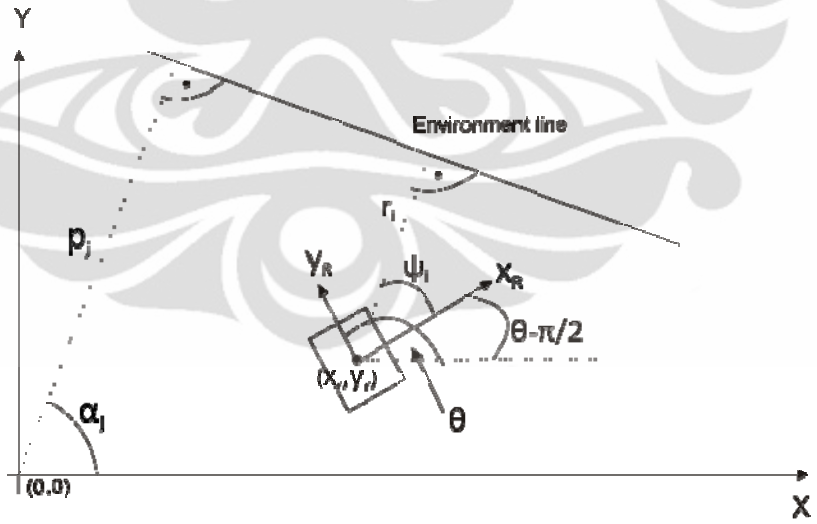
Segmen garis pada kerangka lokal kemudian digunakan sebagai vektor *measurement*, $\boldsymbol{z}$, sedangkan segmen garis pada kerangka global kemudian

ditransformasikan ke kerangka lokal robot dengan memanfaatkan *pose* robot yang telah dideteksi sebelumnya. Persamaan yang digunakan dapat dilihat pada (3.53) [6]

$$\begin{bmatrix} \hat{r}_i \\ \hat{\psi}_i \end{bmatrix} = \begin{bmatrix} |C_j| \\ \alpha_j - \left(\hat{\theta}^-_{r,k} - \frac{\pi}{2}\right) + \left(-0.5 sign(C_j) + 0.5\right)\pi \end{bmatrix} \tag{3.53}$$

$$C_j = p_j - \hat{x}^-_{r,k}\cos\hat{\theta}^-_{r,k} - \hat{y}^-_{r,k}\sin\hat{\theta}^-_{r,k}$$

Dimana $\hat{r}_i$ dan $\hat{\psi}_i$ masing-masing adalah jarak dan sudut segmen garis yang terdeteksi terhadap *mobile robot*, sedangkan α_j dan p_j adalah jarak dan sudut segmen garis pada map global terhadap *origin* (0,0). Ilustrasi posisi garis dan robot serta parameter-parameter yang menyertainya dapat dilihat pada gambar 3.3 Pada gambar terlihat bahwa untuk strategi lokalisasi menggunakan *line extraction* ini, sumbu y pada kerangka referensi global didefinisikan berhimpit dengan arah gerak maju robot. Di sisi lain, sumbu x pada kerangka referensi lokal didefinisikan tegak lurus terhadap arah gerak maju robot.



Gambar 3.3 Ilustrasi sistem koordinat dan parameter garis

Persamaan transformasi yang digunakan juga memanfaatkan informasi *pose* robot yang didapat dari tahap *state* projection, sehingga persamaan tersebut dapat dikatakan merupakan persamaan *measurement prediction*, *h*(.) pada tahap *correction*.

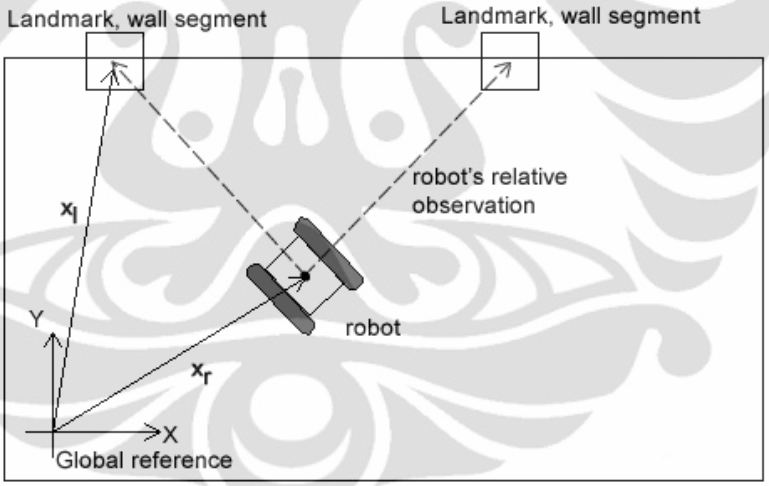
b. *Landmark Detection*

Istilah *landmark* pada metode ini memiliki arti poin pada lingkungan yang dijadikan patokan oleh *mobile robot* dalam menentukan posisi. Pengolahan dengan metode ini dilakukan dengan cara mentransformasi jarak *mobile robot* dengan suatu poin menjadi suatu koordinat pada kerangka global dengan menggunakan (3.54) dan (3.55)

$$\hat{x}_v = \hat{x}_{r,k}^- + r\cos\hat{\theta}_{r,k}^- \tag{3.54}$$

$$\hat{y}_v = \hat{y}_{r,k}^- + r\sin\hat{\theta}_{r,k}^- \tag{3.55}$$

Dimana $(\hat{x}_v, \hat{y}_v)$ adalah koordinat poin yang terdeteksi dan r adalah jarak hasil pembacaan sensor. Poin-poin ini kemudian dicocokkan dengan *landmark* pada map untuk mendapatkan koordinat dari *landmark* yang terdeteksi (x_l, y_l). Ilustrasi posisi robot dan *landmark* dapat dilihat pada gambar 3.4



Gambar 3.4 Ilustrasi posisi robot dan *landmark*

Berikutnya, jarak (r) dan sudut (ϕ) poin *landmark* terhadap robot yang dideteksi oleh sensor digunakan sebagai vektor *measurement*, $\boldsymbol{z}$. Untuk persamaan *measurement prediction*, $h(.)$ digunakanlah (3.56) [20]

$$h\left(\hat{\boldsymbol{x}}_{r,k}^-\right) = \begin{bmatrix}\hat{r}_i \\ \hat{\phi}_i\end{bmatrix} = \begin{bmatrix}\sqrt{(x_{l,i}-\hat{x}_{r,k}^-)^2 + (y_{l,i}-\hat{y}_{r,k}^-)^2} \\ \tan^{-1}\frac{(y_l-\hat{y}_{r,k}^-)}{(x_l-\hat{x}_{r,k}^-)} - \hat{\theta}_{r,k}^-\end{bmatrix} \tag{3.56}$$

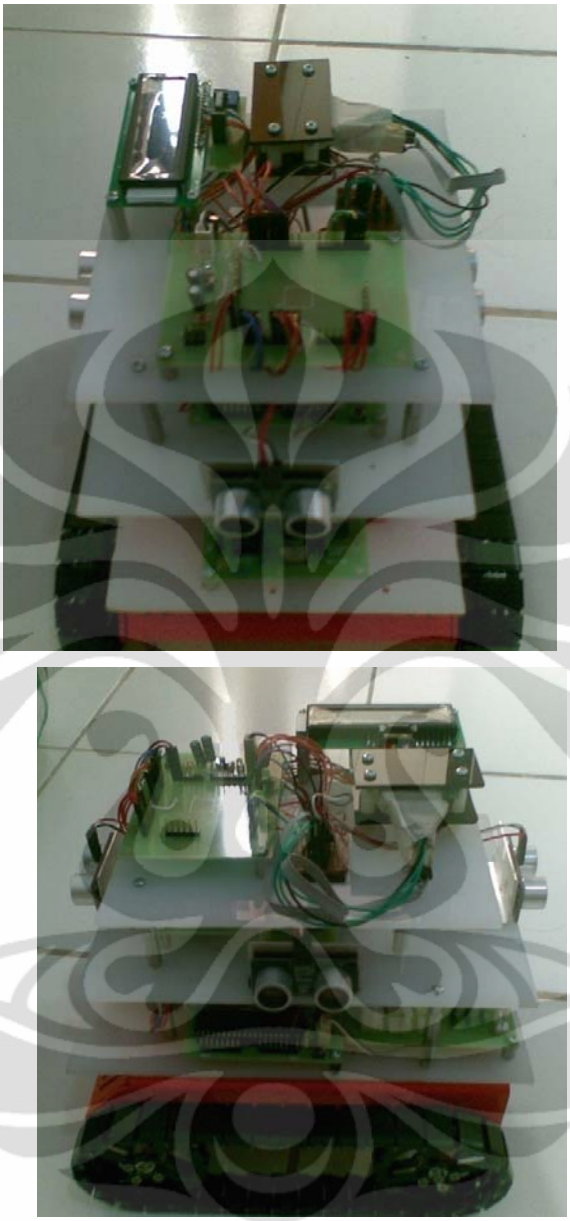
BAB 4
PERANCANGAN SISTEM *MOBILE ROBOT*

4.1 Desain *Hardware*

Tujuan akhir dari penelitian ini adalah merancang *mobile robot* yang memiliki kemampuan untuk melakukan lokalisasi pada area terstruktur dengan sumber daya sensor minim. Untuk itu diperlukan konstruksi *mobile robot* yang khusus, baik dari segi *hardware* maupun dari segi algoritma lokalisasi yang digunakan.

Komponen-komponen yang diperlukan dalam sebuah *mobile robot* telah dijelaskan pada bab II. Metode lokalisasi yang akan digunakan telah dijelaskan pada bab III beserta dasar teori tentang *Kalman Filter*. Berikutnya, berdasarkan penjelasan pada bab-bab sebelumnya, bab ini akan menjelaskan tentang perancangan sistem *mobile robot* yang akan digunakan dalam penelitian. Pertama akan dijelaskan tentang desain hardware *mobile robot* yang akan digunakan, selain itu juga akan dijelaskan juga mengenai model kinematik *mobile robot*, koreksi odometri dan terakhir adalah implementasi algoritma lokalisasi.

Pada penelitian ini dirancang sebuah *mobile robot* yang berjenis *track*ed *mobile robot*. *Mobile robot* yang digunakan pada penelitian ini dirancang dengan menggunakan komponen yang telah dijelaskan pada bab I. Utuk *platform*, digunakan *platform* Traxster II produksi RoboticsConnection. Selain itu, digunakan juga layer-layer dari bahan akrilik warna putih untuk tempat meletakkan rangkaian-rangkaian pendukung. Bentuk fisik *mobile robot* ini dapat dilihat pada gambar 4.1 , sedangkan spesifikasi fisiknya dapat dilihat pada tabel 1.1



Gambar 4.1 Bentuk fisik robot yang digunakan pada penelitian

Tabel 4.1 Spesifikasi fisik robot

No	Spesifikasi	Satuan	Nilai
1	Berat	N	2.5
2	Panjang	m	0.23
3	Lebar	m	0.203
4	Tinggi	m	0.21
5	Jenis *Belt*	-	Plastik
6	Lebar *Belt*	m	0.023
7	Luas area kontak	m^2	0.006
8	Kecepatan puncak tiap *track*	m/s	0.586

Pada *mobile robot* dipasang empat buah sensor jarak PING di sisi kiri, kanan, depan dan belakang robot. Ketentuan pemasangan sensor-sensor tersebut adalah sebagai berikut :

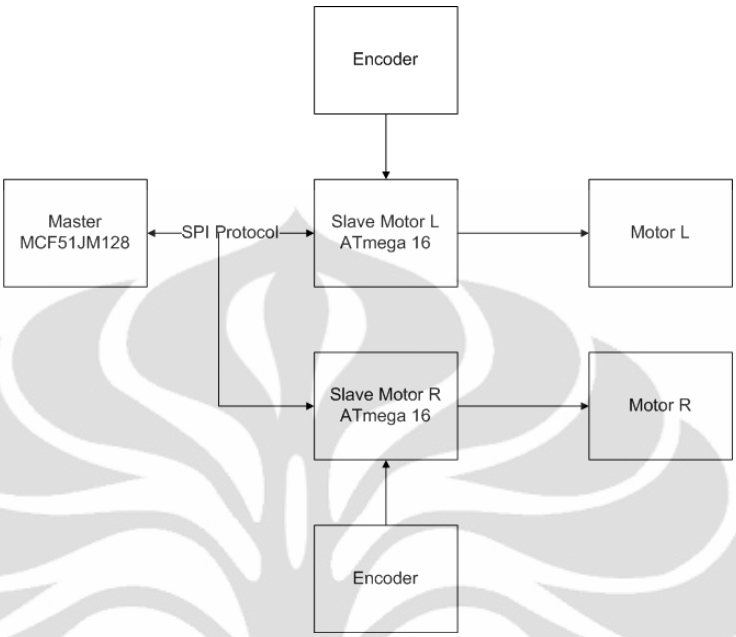
1. Sensor bagian depan berjarak 11 cm dari titik pusat robot
2. Sensor jarak bagian belakang berjarak 12.5 cm dari titik pusat robot
3. Sensor jarak bagian kiri berjarak 9.1 cm dari titik pusat robot
4. Sensor jarak bagian kanan berjarak 8.8 cm dari titik pusat robot

Penerapan algoritma lokalisasi umumnya mengasumsikan sensor jarak menempel pada titik pusat robot. Oleh karena itu, implementasi algoritma lokalisasi pada penelitian ini akan mempertimbangkan *offset* pembacaan sensor seperti yang telah dijelaskan.

Untuk mengendalikan *mobile robot*, digunakanlah mikrokontroler. Ada tiga unit mikrokontroler yang digunakan sebagai pengendali pada *mobile robot* ini. Tugas masing-masing unit mikrokontroler adalah sebagai berikut :

1. Satu unit mikrokontroler sebagai pengendali utama (disebut *master*) yang bertugas memberikan perintah pada *unit* mikrokontroler lain dan menjalankan algoritma lokalisasi. Mikrokontroler yang digunakan sebagai pengendali utama memiliki jenis ColdFire V1 (MCF51JM128) buatan Freescale dengan *internal clock* sebesar 40MHz.
2. Dua unit mikrokontroler sebagai pengendali motor, masing-masing motor kiri dan motor kanan (disebut *slave motor left* dan *slave motor right*). Lebih detail lagi, unit mikrokontroler ini bertugas memberi perintah pada motor melalui *driver* motor dan melakukan pembacaan terhadap sensor *encoder*. Mikrokontroler yang digunakan untuk unit ini adalah ATmega16 produksi Atmel dengan *clock* sebesar 11.0592 MHz, sedangkan *driver* motor yang digunakan adalah *driver H-bridge* berbasiskan IRF9540 dan IRF540.

Tiga mikrokontroler ini berkomunikasi dengan menggunakan protokol SPI. Skema komunikasi antar *unit* mikrokontroler ini dapat dilihat pada gambar 4.2



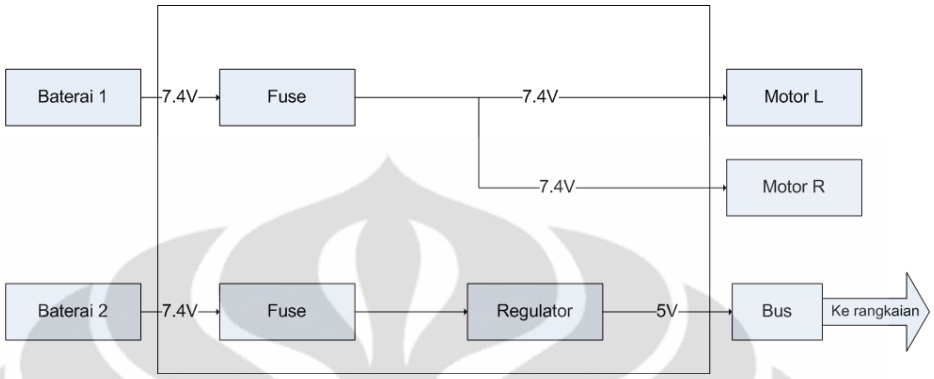
Gambar 4.2 Skema komunikasi antar mikrokontroler

Unit mikrokontroler untuk pengendali motor berkomunikasi dengan *master* melalui satu *channel* SPI yang memiliki kecepatan 155 kHz.

Selain memberikan perintah pada *slave* dan menjalankan algoritma lokalisasi, *master* juga memiliki tugas lain. Tugas yang pertama adalah melakukan pembacaan terhadap sensor jarak. Pembacaan ini dilakukan dengan memanfaatkan fitur mikrokontroler MCF51JM128 berupa *timer* 16 bit. Tugas berikutnya adalah mengendalikan LCD yang digunakan untuk melakukan *debugging*. Tugas lain yang dilakukan oleh *master* adalah melakukan komunikasi dengan PC. Komunikasi ini dilakukan dengan menggunakan wireless USART YS1020U RF Transceiver yang dihubungkan dengan USART *channel* 1.

Seperti telah dijelaskan pada bab II, *mobile robot* yang dirancang ini menggunakan baterai LiPo 7.4V DC. Lebih spesifik lagi digunakan baterai jenis ini sebanyak dua buah. Satu baterai digunakan untuk menyuplai rangkaian dan sensor-sensor pada *mobile robot* dengan tegangan 5V. Baterai lain digunakan untuk memberikan suplai pada motor. Untuk mewujudkan sistem suplai daya tersebut, maka dirancanglah sebuah rangkaian *power*

distribution. Rangkaian ini memiliki skema secara garis besar yang ditunjukkan oleh gambar 4.3



Gambar 4.3 Garis besar skema *power distribution*

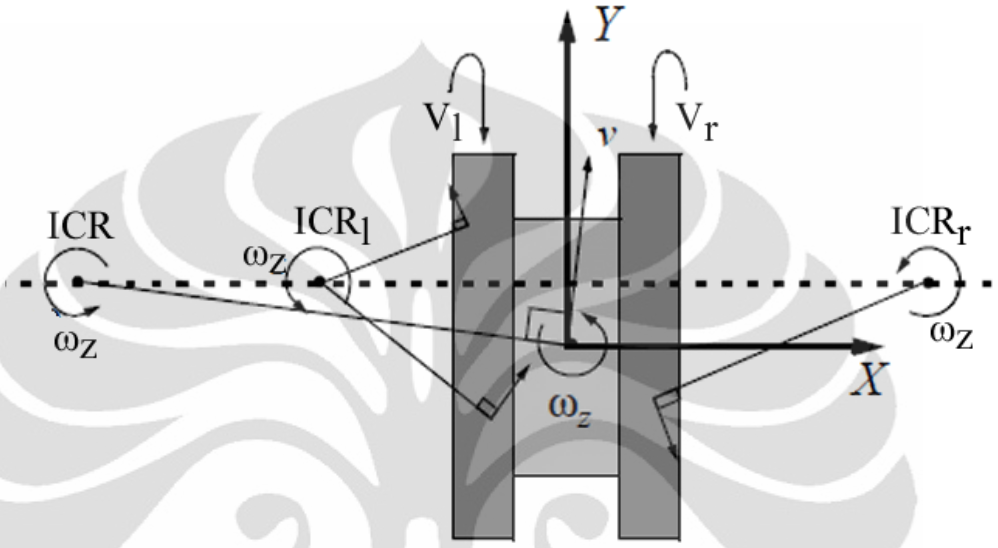
4.2 Model Kinematik

Desain hardware *mobile robot* telah dijelaskan pada bagian sebelumnya. Berikutnya akan dijelaskan mengenai model kinematik *mobile robot* yang dirancang pada penelitian ini. *Mobile robot* yang digunakan pada penelitian ini adalah *mobile robot* berjenis TMR (*track*ed *mobile robot*). Karakteristik khasnya adalah adanya *track* yang meliliti roda depan dan belakang sehingga memiliki bentuk roda seperti tank. Pada [30] disebutkan bahwa TMR ini merupakan *differential mobile robot* yang menggunakan prinsip *skid steering* (variasi dari kecepatan relatif *track*) yang menghasilkan *slippage* dan *soil-shearing* yang pada akhirnya menghasilkan *steering*. Dengan demikian, terdapat kesamaan antara TMR dan DMR dari segi lokomotif. Akan tetapi, kinematika DMR tidak dapat langsung diaplikasikan pada TMR karena pada TMR gaya gesek sangat bepengaruh pada variabel posisi. Diperlukan suatu cara khusus untuk memperoleh model kinematika TMR.

Pada [30], J.L. Martinez melakukan pemodelan kinematik TMR dengan pendekatan analogi terhadap model kinematika DMR. Robot dianggap sebuah *rigid body* dengan kerangka referensi lokal yang memiliki titik *origin* di tengah area antara dua *track*. Pada kerangka referensi ini, sumbu Y didefinisikan searah dengan arah pergerakan maju.

Untuk pergerakan planar, *Instantaneous Center of Rotation* (ICR) didefinisikan sebagai poin pada bidang dimana pergerakan robot dapat direpresentasikan sebagai putaran tanpa gerak translasi. Pada TMR, terdapat

tiga bagian yang memiliki ICR, yaitu masing-masing *track* (kiri dan kanan) dan robot itu sendiri. Ketiga ICR ini terletak pada satu sumbu yang sejajar dengan sumbu x robot, namun memiliki letak yang berbeda. Penggambaran posisi ICR tersebut dapat dilihat pada gambar 4.4



Gambar 4.4 Penggambaran posisi ICR [30]

Koordinat ICR kiri, kanan dan ICR robot secara keseluruhan terhadap kerangka referensi lokal (robot) dapat diperoleh secara geometris melalui (4.1)-(4.4)

$$x_{ICR} = \frac{-v_y}{\omega_z} \tag{4.1}$$

$$y_{ICR} = y_{ICR\,l} = y_{ICR\,r} = \frac{v_x}{\omega_z} \tag{4.2}$$

$$x_{ICR\,l} = \frac{V_l - v_y}{\omega_z} \tag{4.3}$$

$$x_{ICR\,r} = \frac{V_r - v_y}{\omega_z} \tag{4.4}$$

dengan v_x dan v_y adalah kecepatan translasional robot pada kerangka referensi lokal.

Nilai v_x, v_y, dan ω_z dapat diperoleh dengan menggunakan persamaan *direct kinematics* seperti tertera pada (4.5) – (4.7)

$$v_x = \frac{V_r - V_l}{x_{ICR\,r} - x_{ICR\,l}} y_{ICR} \tag{4.5}$$

$$v_y = \frac{V_r+V_l}{2} - \frac{V_r-V_l}{x_{ICR\,r}-x_{ICR\,l}} \frac{x_{ICR\,r}+x_{ICR\,l}}{2} \quad (4.6)$$

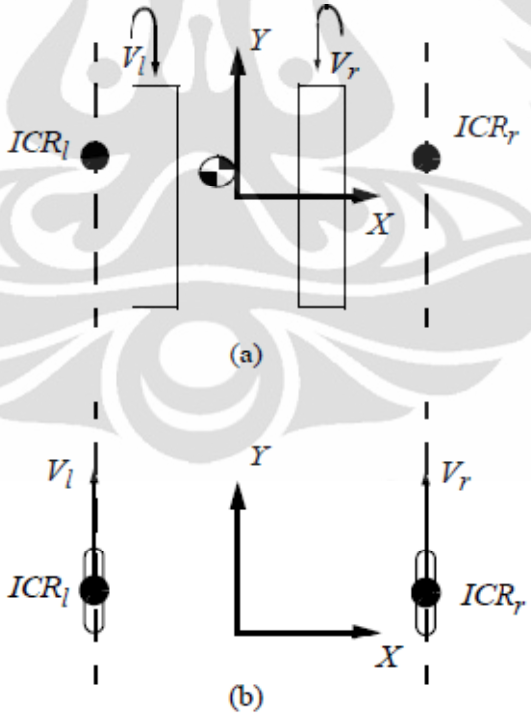
$$\omega_z = \frac{V_r-V_l}{x_{ICR\,r}-x_{ICR\,l}} \quad (4.7)$$

Sedangkan persamaan *inverse kinematics* dapat dilihat pada 4.8 dan 4.9

$$V_l = \sqrt{\|v\|^2 - y_{ICR}^2 \times \omega_z^2} + x_{ICR\,l} \times \omega_z \quad (4.8)$$

$$V_r = \sqrt{\|v\|^2 - y_{ICR}^2 \times \omega_z^2} + x_{ICR\,r} \times \omega_z \quad (4.9)$$

Persamaan-persamaan *direct* dan *inverse kinematics* pada (4.5)-(4.9) berlaku baik pada TMR maupun DMR untuk pergerakan sesaat. Perbedaan antara TMR dan DMR adalah pada DMR poin-poin ICR memiliki letak yang tetap dan berada pada titik kontak roda dengan bidang. Di sisi lain poin-poin ICR pada TMR bergantung pada efek dinamika dan selalu berada di luar *track*. Ilustrasi dari fakta ini dapat dilihat pada gambar 4.5. Dengan demikian, dapat dikatakan bahwa efek dinamik *mobile robot* dapat disertakan dalam model kinematik dengan adanya dua poin ICR r dan ICR l.



Gambar 4.5 Penggambaran posisi ICR pada (a) TMR, dan (b) DMR [30]

Pada [30] didefinisikan indeks-indeks sebagai berikut

1. *Steering efficiency index*

Steering efficiency index, c, merepresentasikan ada tidaknya *slippage*, dan didefinisikan sebagai jarak antara ICR yang ternormalisasi. Secara matematis, c didefinisikan oleh (4.10)

$$c = \frac{x_{ICR\,r} - x_{ICR\,l}}{L};\ (c \geq 1) \qquad (4.10)$$

Dimana L adalah jarak antara titik tengah *track*. Pada DMR ideal, dimana tidak ada *slippage*, indeks ini bernilai 1

2. *Eccentricity index*

   Eccentricity index merepresentasikan kesimetrisan poin ICR *track* kiri dan kanan terhadap sumbu Y lokal. Secara matematis indeks ini didefinisikan oleh (4.11)

$$e = \frac{x_{ICR\,r} + x_{ICR\,l}}{x_{ICR\,r} - x_{ICR\,l}} \qquad (4.11)$$

   Indeks ini bernilai 0 pada DMR ideal, dimana ICR untuk *track* kiri dan kanan memiliki posisi yang simetris terhadap sumbu Y lokal.

Berikutnya, J.L. Martinez, dkk melakukan penalaan terhadap parameter ICR menggunakan *genetic algorithm*. Penelitian dilakukan pada TMR (*Track*ed *Mobile Robot*) Auriga-α. Didapatlah parameter-parameter sebagai berikut (semua posisi berdasarkan koordinat lokal) :

1. y_{ICR} bernilai 0.0343 m
2. $x_{ICR\,r}$ bernilai 0.4202 m
3. $x_{ICR\,l}$ bernilai -0.3558 m
4. *Eccentricity index*, c, bernilai 1.847
5. *Steering efficiency index*, e, bernilai 0.083

Dari parameter yang telah didapatkan tersebut, dapat dicari parameter yang sama untuk *mobile robot* yang digunakan pada penelitian ini. Parameter ini dicari menggunakan data rasio berat [5]. Pada karakteristik berat ini telah terkandung karakteristik dinamika yang juga berhubungan dengan parameter ICR.

Pertama didefinisikan parameter ΔICR_l dan ΔICR_r yang merupakan selisih antara nilai koordinat ICR yang didapat dari percobaan dengan nilai koordinat base ICR, yaitu nilai ICR jika *mobile robot* adalah DMR ideal. Pada Auriga-α, nilai-nilai tersebut adalah:

$$ICR_{l,Auriga} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.3558 \\ 0.0343 \end{bmatrix} \quad (4.12)$$

$$ICR_{r,Auriga} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.4204 \\ 0.0343 \end{bmatrix} \quad (4.13)$$

$$base\ ICR_{l,Auriga} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.3750 \\ 0 \end{bmatrix} \quad (4.14)$$

$$base\ ICR_{r,Auriga} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.3750 \\ 0 \end{bmatrix} \quad (4.15)$$

Selanjutnya didapatlah nilai ΔICR_l dan ΔICR_r

$$\Delta ICR_{l,Auriga} = ICR_{l,Auriga} - base\ ICR_{l,Auriga} = \begin{bmatrix} 0.0192 \\ 0.0343 \end{bmatrix} \quad (4.16)$$

$$\Delta ICR_{r,Auriga} = ICR_{r,Auriga} - base\ ICR_{r,Auriga} = \begin{bmatrix} 0.0452 \\ 0.0343 \end{bmatrix} \quad (4.17)$$

Berikutnya dicari nilai r_w yang merupakan nilai rasio berat *mobile robot* yang digunakan pada penelitian ini dengan Auriga-α. Berdasarkan tabel 4.1, berat *mobile robot* yang digunakan dalam penelitian ini adalah 2.5N, sedangkan berat Auriga-α adalah 258N. Dengan demikian didapat nilai r_w sebesar 0.00969.

Dengan diperolehnya nilai r_w, maka nilai ΔICR_l dan ΔICR_r dapat dicari, yaitu dengan menggunakan perhitungan pada (4.18) dan (4.19) [5]:

$$\Delta ICR_{l,TMR} = \Delta ICR_{l,Auriga} \times r_w = \begin{bmatrix} 0.00019 \\ 0.00033 \end{bmatrix} \quad (4.18)$$

$$\Delta ICR_{r,TMR} = \Delta ICR_{r,Auriga} \times r_w = \begin{bmatrix} 0.00044 \\ 0.00033 \end{bmatrix} \quad (4.19)$$

dimana nilai yang diperoleh adalah dalam satuan meter. Selanjutnya, dengan mempertimbangkan nilai base ICR seperti disajikan dalam (4.20)-(4.21)

$$base\ ICR_{l,TMR} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.1015 \\ 0 \end{bmatrix} \quad (4.20)$$

$$base\ ICR_{r,TMR} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.1015 \\ 0 \end{bmatrix} \quad (4.21)$$

dapat diperoleh nilai ICR untuk *track* kiri dan kanan

$$ICR_{l,TMR} = base\ ICR_{l,TMR} + \Delta ICR_{l,TMR} = \begin{bmatrix} -0.10169 \\ 0.33 \end{bmatrix} \quad (4.22)$$

$$ICR_{r,TMR} = base\ ICR_{r,TMR} + \Delta ICR_{r,TMR} = \begin{bmatrix} 0.10169 \\ 0.33 \end{bmatrix} \quad (4.23)$$

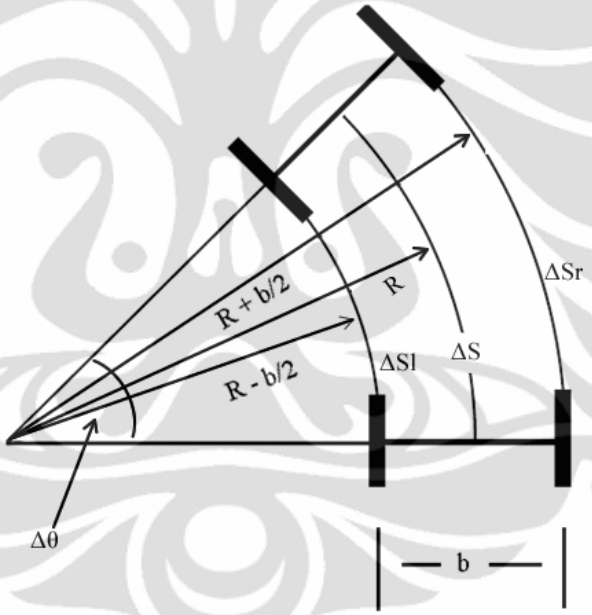
Berikutnya, dengan menggunakan (4.10) dan (4.11), diperoleh nilai steering efficiency index (c) dan eccentricity index (e) sebagai berikut:

$$c = 1.0031$$

$$e = 0.0012$$

Terlihat bahwa nilai steering efficiency index yang diperoleh mendekati 1 dan nilai eccentricity index yang diperoleh mendekati 0. Hal ini berarti, model kinematika *mobile robot* yang digunakan dalam penelitian ini dapat didekati dengan model kinematika DMR ideal.

Jika diasumsikan kecepatan putar roda konstan, perubahan posisi dan orientasi robot dapat diperoleh melalui total jumlah dan selisih perpindahan roda kiri dan kanan [31]. Hubungan ini dapat dilihat pada gambar 4.6.



Gambar 4.6 Hubungan antara perpindahan roda kiri dan kanan dengan perpindahan robot dan perubahan orientasinya [31]

Pada gambar, dua roda bergerak dengan lintasan membentuk lengkungan dengan jari-jari R dan perubahan sudut Δθ. Untuk wheelbase b, yaitu jarak antara titik kontak masing-masing roda dengan lantai, roda kiri bergerak dalam lintasan lengkungan yang memiliki jari-jari R-(b/2) dan roda kanan bergerak dalam lintasan lengkungan dengan jari-jari R + (b/2). Panjang lintasan masing-masing roda dinyatakan dengan simbol ΔS_l dan ΔS_r. Selanjutnya berlaku hubungan seperti tertera pada (4.24)-(4.26)

$$\Delta S = \Delta\theta R \tag{4.24}$$

$$\Delta S_l = \Delta\theta \left(R - \frac{b}{2}\right) \tag{4.25}$$

$$\Delta S_r = \Delta\theta \left(R + \frac{b}{2}\right) \tag{4.26}$$

dimana ΔS adalah panjang lintasan lengkung yang dilalui oleh robot. Dengan mengurangi (4.26) dengan (4.25), didapatlah (4.27) dan (4.28)

$$\Delta S_r - \Delta S_l = \Delta\theta b \tag{4.27}$$

maka

$$\Delta\theta = \frac{\Delta S_r - \Delta S_l}{b} \tag{4.28}$$

Panjang lintasan lengkung yang dilalui poin yang terletak di antara roda dapat dipandang sebagai penjumlahan antara (4.25) dan (4.26). Maka

$$\begin{aligned} \Delta S_r + \Delta S_l &= \Delta\theta \left(R + \frac{b}{2}\right) - \Delta\theta \left(R - \frac{b}{2}\right) \\ &= 2\,\Delta\theta\, R \\ &= 2\Delta S \end{aligned} \tag{4.29}$$

dengan demikian, perpindahan maju dari robot dapat dihitung dengan menggunakan (4.30)

$$\Delta S = \frac{\Delta S_r + \Delta S_l}{2} \tag{4.30}$$

Dengan menggunakan model kinematik yang telah didapat, maka dapat didefinisikan persamaan *pose* untuk waktu pencuplikan berikutnya, t+1, seperti dijelaskan pada (4.31)

$$p_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + \Delta S \cos\theta_{t+1} \\ y_t + \Delta S \sin\theta_{t+1} \\ \theta_t + \Delta\theta \end{bmatrix} \tag{4.31}$$

Persamaan (4.31) dapat pula direpresentasikan dalam bentuk fungsi yang menghubungkan *pose* sekarang dengan waktu sebelumnya, seperti terlihat pada (4.32)

$$p_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + \Delta S \cos\theta_t \\ y_{t-1} + \Delta S \sin\theta_t \\ \theta_{t-1} + \Delta\theta \end{bmatrix} \tag{4.32}$$

4.3 Odometri

Penerapan model kinematik untuk menentukan *pose* robot memerlukan pengetahuan tentang panjang lintasan yang dilalui roda kiri dan roda kanan

seperti terlihat pada (4.30). Panjang lintasan ini dapat diperoleh dengan mengaplikasikan persamaan odometri pada data *encoder*. Persamaan yang digunakan dapat dilihat pada (4.33) [18]

$$\Delta S_{L/R} = c_m N_{L/R} \tag{4.33}$$

dimana c_m adalah faktor konversi yang mentransformasikan pulsa *encoder* menjadi perpindahan linear, sedangkan $N_{L/R}$ adalah jumlah pulsa *encoder*. Faktor c_m dapat dihitung dengan menggunakan (4.34)

$$c_m = \frac{\pi D_n}{n C_e} \tag{4.34}$$

dimana D_n adalah diameter roda nominal dalam mm, C_e adalah resolusi *encoder* dalam ppr (pulse per revolution), dan n adalah gear ratio. Pada *mobile robot* yang digunakan pada penelitian ini, nilai D_n adalah 72 mm, nilai n adalah 1, dan nilai C_e adalah 624 ppr.

Keunggulan penggunaan odometri adalah persamaan yang digunakan sederhana dan mudah diimplementasikan. Hanya saja, proses odometri memiliki error yang cenderung terakumulasi untuk setiap *step*. Error ini dapat digolongkan menjadi dua jenis, yaitu systematic error dan non-systematic errror. Systematic error adalah error yang berhubungan dengan kinematika robot dan terjadi karena diameter roda yang tidak sama dan ketidakpastian mengenai wheelbase (jarak antar roda, b) efektif. Error tersebut masing-masing kemudian didefinisikan sebagai E_d dan E_b. Persamaan yang menjelaskan E_d dan E_b dapat dilihat pada (4.35) dan (4.36)

$$E_d = \frac{D_R}{D_L} \tag{4.35}$$

$$E_b = \frac{b_{actual}}{b_{nominal}} \tag{4.36}$$

Di sisi lain, non-systematic error adalah error yang tidak berhubungan dengan kinematika robot. Error jenis ini biasanya terjadi karena kondisi lantai yang tidak seragam, selip pada roda, dan lain-lain. Error pada odometri ini dapat dikurangi, salah satunya dengan cara mengimplementasikan faktor kalibrasi.

Salah satu metode yang dapat diterapkan untuk mendapatkan faktor kalibrasi adalah UMBmark, yang dikembangkan oleh J. Borenstein dan L.Feng [18]. Metode ini adalah metode kalibrasi offline yang memiliki

keunggulan mudah dilakukan, sistematis, dan tidak memerlukan peralatan yang kompleks [32]. Langkah-langkah yang dilakukan pada metode ini adalah sebagai berikut :

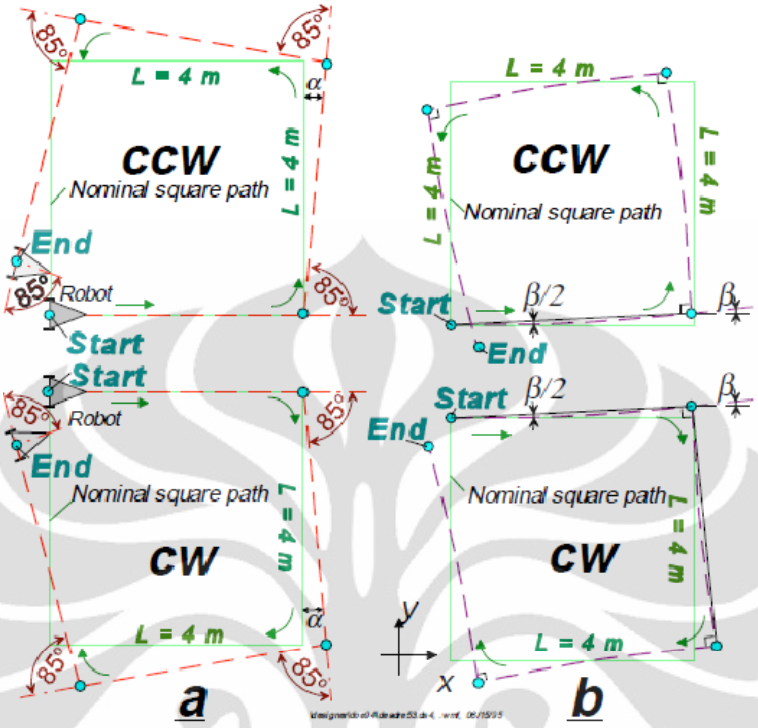
1. Ukur posisi absolut robot pada posisi awal
2. Jalankan robot pada lintasan persegi searah jarum jam (clockwise, cw), dengan ketentuan:
   a. Berhentikan robot setiap selesai melakukan gerakan lurus pada satu sisi lintasan
   b. Lakukan putaran 90º pada satu titik sebanyak total empat kali
   c. Jalankan robot dengan pelan untuk menghindari *slippage*
3. Saat robot kembali ke posisi awal, ukur posisi absolutnya
4. Bandingkan posisi absolut robot dengan posisi yang didapat dari odometri, cari errornya
5. Ulangi langkah 1-4 sebanyak 5 kali
6. Ulangi langkah 1-5 dengan arah berlawanan jarum jam (counter clockwise, ccw)

Setelah langkah 1-6 dilakukan, akan didapatkan 5 data error posisi (x,y) untuk arah berlawanan dan searah jarum jam. Berikutnya dicari titik berat dari data error untuk lintasan searah dan berlawanan arah jarum jam. Persamaan yang digunakan untuk mencari titik berat tersebut dapat dilihat pada (4.37) dan (4.38)

$$x_{c.g,cw/ccw} = \frac{1}{n}\sum_{i=1}^{n} error\ x_{i,cw/ccw} \quad (4.37)$$

$$y_{c.g,cw/ccw} = \frac{1}{n}\sum_{i=1}^{n} error\ y_{i,cw/ccw} \quad (4.38)$$

Langkah selanjutnya adalah menelaah lintasan yang dilalui robot saat melakukan proses UMBmark. Contoh lintasan yang mungkin terbentuk ditunjukkan oleh gambar 4.7



Gambar 4.7 Contoh lintasan yang mungkin terbentuk pada UMBmark yang dipengaruhi (a) error tipe a dan (b) error tipe b [18]

Dari gambar 4.7 dapat didefinisikan dua tipe error dalam konteks tes UMBmark yang berhubungan dengan error orientasi, yaitu error tipe A dan tipe B. Error tipe A adalah error yang menambah atau mengurangi jumlah total sudut rotasi yang terjadi pada arah cw dan ccw. Error ini berkaitan dengan E_b. Penggambaran error tipe A dapat dilihat pada gambar 4.7a. error tipe A ini menyebabkan robot berputar terlalu sedikit atau terlalu banyak pada setiap perputaran 90º. Pada gambar, error sudut yang terjadi pada setiap perputaran 90º disimbolkan dengan α.

Di sisi lain, error tipe B adalah error yang menambah/mengurangi jumlah total sudut rotasi pada satu arah dan mengurangi/menambah jumlah total sudut rotasi pada arah sebaliknya. Error ini berkaitan dengan E_d, dan menyebabkan robot melakukan lintasan yang sedikit melengkung saat diperintahan bergerak lurus. Akibatnya, terjadilah error pada orientasi robot sebesar β. Penggambaran error ini dapat dilihat pada gambar 4.7b.

Nilai α dan β (dalam satuan derajat) ini kemudian dapat dihitung dengan menggunakan (4.39) dan (4.40)

$$\alpha = \frac{x_{c.g,cw} + x_{\text{c.g,ccw}}}{-4\text{L}} \quad (4.39)$$

$$\beta = \frac{x_{c.g,cw} - x_{\text{c.g,ccw}}}{-4\text{L}} \quad (4.40)$$

dimana L adalah panjang sisi lintasan. Berikutnya dihitung pula jari-jari kelengkungan lintasan yang terjadi karena error tipe B menggunakan (4.41)

$$R = \frac{L/2}{\sin(\beta/2)} \quad (4.41)$$

Jika nilai R sudah didapat, maka rasio diameter roda kiri dan kanan dapat diperoleh melalui (4.42)

$$E_d = \frac{D_R}{D_L} = \frac{R + b/2}{R - b/2} \quad (4.42)$$

Di sisi lain, nilai E_b dapat diperoleh menggunakan (4.43)

$$E_b = \frac{90^\circ}{90^\circ - \alpha} \quad (4.43)$$

Dengan demikian berlaku (4.44)

$$b_{actual} = \frac{90^\circ}{90^\circ - \alpha} b_{nominal} \quad (4.44)$$

Dari nilai E_d dapat diperoleh faktor koreksi untuk mengurangi error akibat ketidaksamaan diameter roda. Dengan mendefinisikan D_a sebagai diameter roda rata-rata, didapat (4.45)

$$D_a = \frac{D_R + D_L}{2} \quad (4.45)$$

Berikutnya, dapat diperoleh D_R dan D_L dari (4.46) dan (4.47)

$$D_L = \frac{2}{E_d + 1} D_a \quad (4.46)$$

$$D_R = \frac{2}{(1/E_d) + 1} D_a \quad (4.47)$$

Kemudian dari (4.46) dan (4.47) dapat didefinisikan dua faktor koreksi seperti dijelaskan pada (4.48) dan (4.49)

$$C_L = \frac{2}{E_d + 1} \quad (4.48)$$

$$C_R = \frac{2}{(1/E_d) + 1} \quad (4.49)$$

Dengan demikian, (4.33) mengalami perubahan menjadi (4.50)

$$\Delta S_{L/R} = C_{L/R} C_m N_{L/R} \quad (4.50)$$

Tes UMBmark dengan menggunakan *mobile robot* berbasis platform TraxsterII telah dilakukan pada [5]. Hasil yang diperoleh adalah sebagai berikut:

1. Nilai b_{actual} = 223 mm
2. Nilai C_L = 0.9264
3. Nilai C_R = 1.0736

Penelitian ini dilakukan dengan menggunakan *mobile robot* berbasis *platform* yang sama, yaitu TraxsterII. Lingkungan kerja yang digunakan juga memiliki lantai yang tertutup karpet dengan jenis yang sama. Oleh karena itu, hasil UMBmark yang dilakukan pada [5] dapat diterapkan pula pada penelitian ini.

4.4 Implementasi Algoritma Lokalisasi

4.4.1 Pendahuluan

Algoritma lokalisasi robot telah dijelaskan pada subbab 3.3. Model kinematik robot dan persamaan odometrinya telah dijelaskan masing-masing pada subbab 4.2 dan 4.3. Subbab ini kemudian akan menjelaskan implementasi algoritma lokalisasi yang dilakukan pada penelitian ini. Dimulai dengan penjelasan tentang kerangka referensi yang digunakan oleh robot, penentuan parameter *Extended Kalman Filter*, kemudian dilanjutkan dengan strategi *line extraction* dan *landmark detection*.

Kerangka referensi yang digunakan oleh robot secara umum mengacu pada kerangka referensi yang digambarkan pada gambar 3.1. Sumbu x dan y robot berhimpit dengan sumbu x dan y kerangka referensi global, dengan sumbu x robot berhimpit dengan arah maju robot. Orientasi θ didefinisikan sebagai sudut yang terbentuk antara sumbu x robot dengan sumbu x referensi global. Pengecualian terjadi pada strategi *line extraction*. Pada strategi ini definisi kerangka referensi yang digunakan mengacu pada gambar 3.3.

Pada subbab 3.4 dijelaskan bahwa proses *prediction* pada algoritma lokalisasi yang digunakan memanfaatkan model kinematika robot dan persamaan odometri. Model kinematika ini telah didapatkan pada subbab 4.3. Dengan demikian, fungsi *f*(.) kini dapat didefinisikan sebagai (4.51)

$$f\left(\hat{\boldsymbol{x}}_{r,k-1}^{-}\right) = \begin{bmatrix} \hat{x}_{r,k-1}^{-} + \Delta S \cos \hat{\theta}_{r,k-1}^{-} \\ \hat{y}_{r,k-1}^{-} + \Delta S \sin \hat{\theta}_{r,k-1}^{-} \\ \hat{\theta}_{r,k-1}^{-} + \Delta\theta \end{bmatrix} \quad (4.51)$$

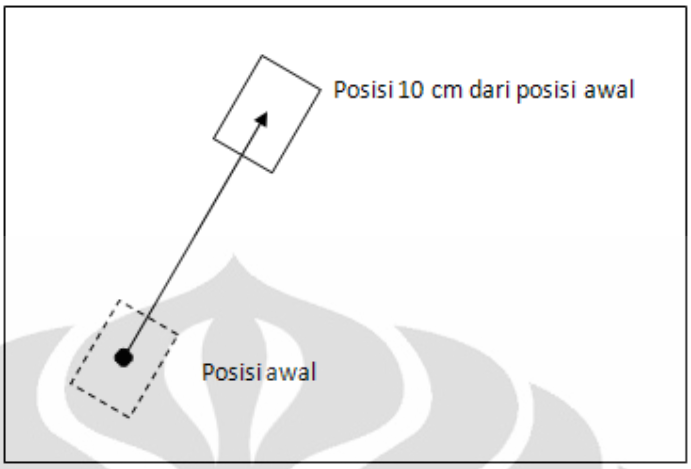
Sehingga matriks ***F***, jacobian dari *f*(.) juga dapat didefinisikan dengan (4.52)

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & \Delta S \sin \hat{\theta}_{r,k-1}^{-} \\ 0 & 1 & \Delta S \cos \hat{\theta}_{r,k}^{-} \\ 0 & 0 & 1 \end{bmatrix} \quad (4.52)$$

Salah satu hal yang perlu diperhatikan pada penggunaan algoritma *Extended Kalman Filter* adalah penentuan nilai awal parameter noise covariance pada matriks P dan nilai matriks Q dan R. Nilai matriks P pada penelitian ini di-set dengan nilai 1000. Hal ini berarti pada awalnya, ketidakpastian pada estimasi *state* cukup besar. Matriks P yang digunakan pada penelitian ini kemudian dapat ditunjukkan dengan (4.53)

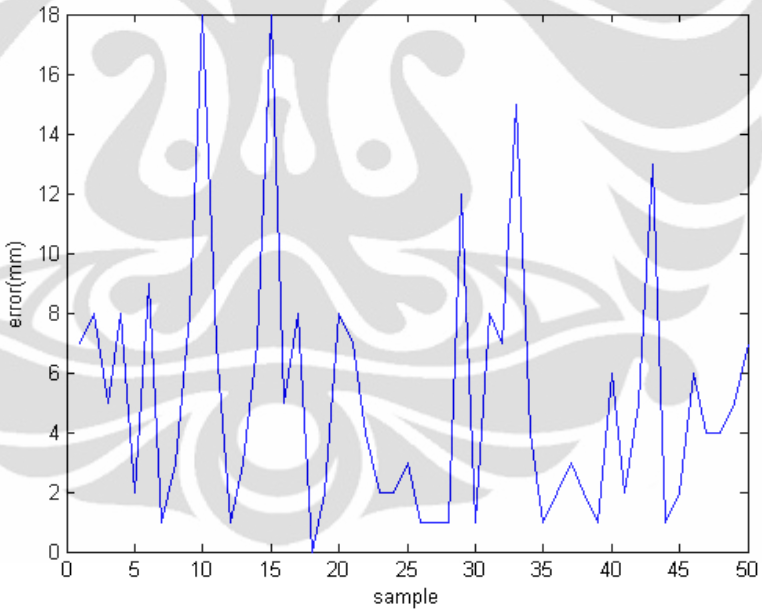
$$\boldsymbol{P} = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix} \quad (4.53)$$

Matriks ***Q*** merupakan matriks process noise covariance. Dalam hal ini, berarti matriks ini mengandung ukuran ketidakpastian dari proses *state* projection, atau dengan kata lain ketidakpastian dari proses odometri dan model kinematika robot. Untuk mengetahui ketidakpastian tersebut, dilakukan percobaan untuk mengetahui nilai error proses odometri. Robot diperintahkan untuk berjalan maju 10 cm, kemudian hasil yang didapat dari perhitungan odometri dibandingkan dengan pengukuran posisi absolutnya. Ilustrasi percobaan yang dilakukan diperlihatkan pada gambar 4.8

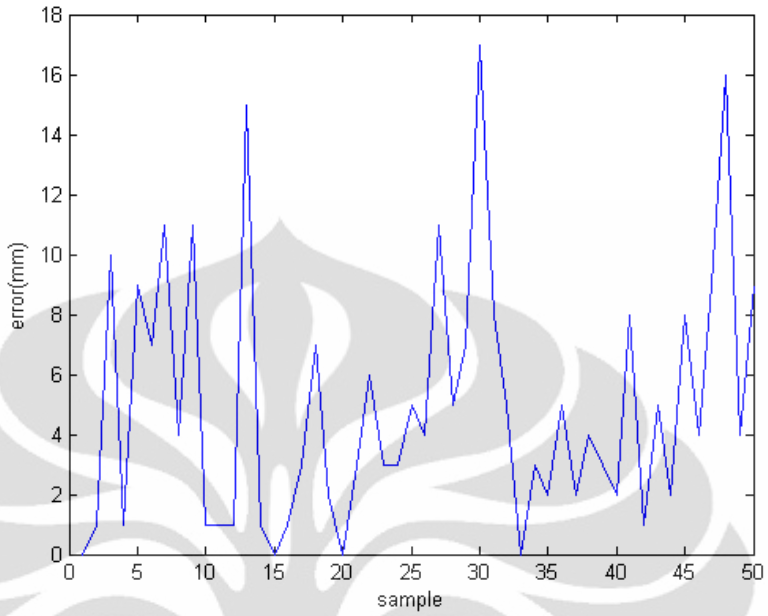


Gambar 4.8 Ilustrasi Percobaan yang dilakukan untuk menentukan ketidakpastian estimasi posisi

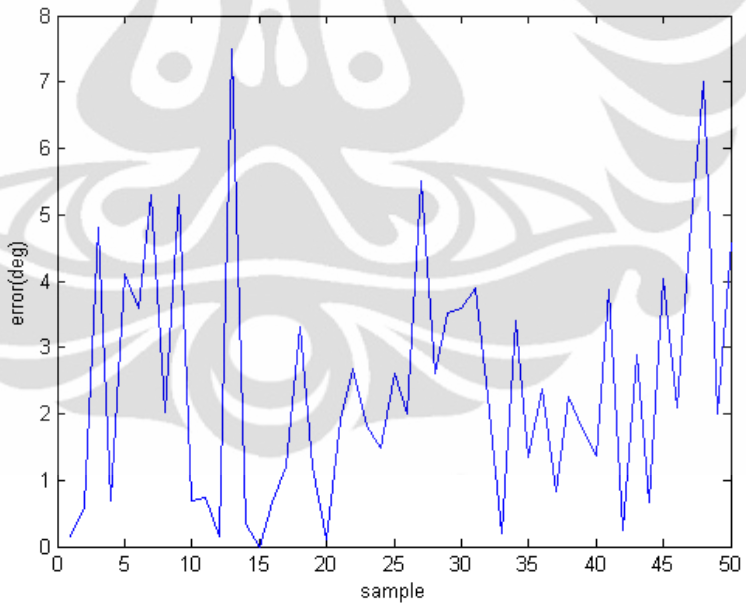
Dari percobaan didapat data error posisi x, y dan orientasi robot seperti terlihat masing-masing pada gambar 4.9, 4.10 dan 4.11



Gambar 4.9 Error posisi x



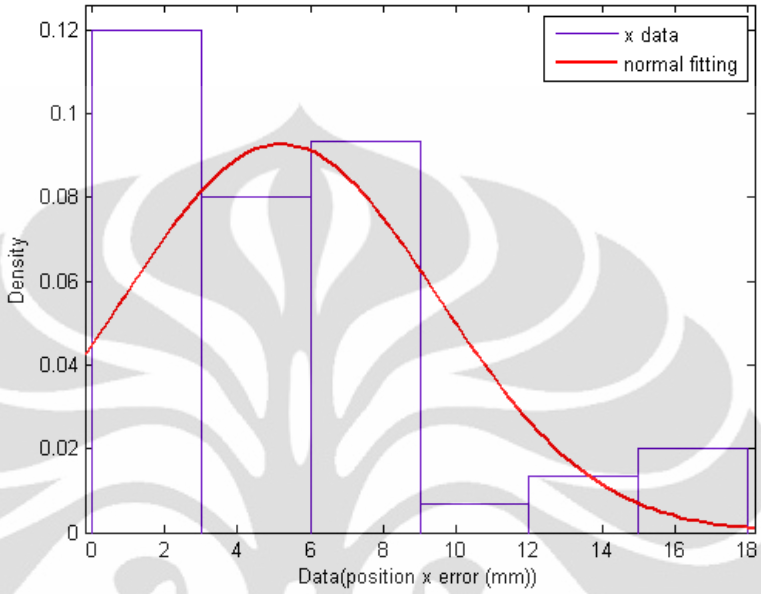
Gambar 4.10 Error posisi y



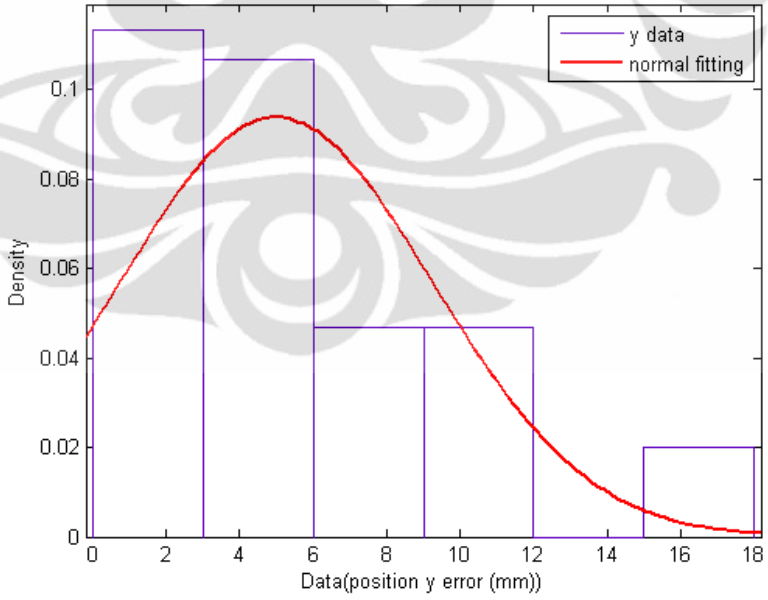
Gambar 4.11 Error orientasi

Selanjutnya dengan menganggap bahwa noise yang mengakibatkan adanya error adalah gaussian noise, dilakukan *normal distribution fitting* pada data error yang telah didapat. Hasil *fitting* untuk error posisi

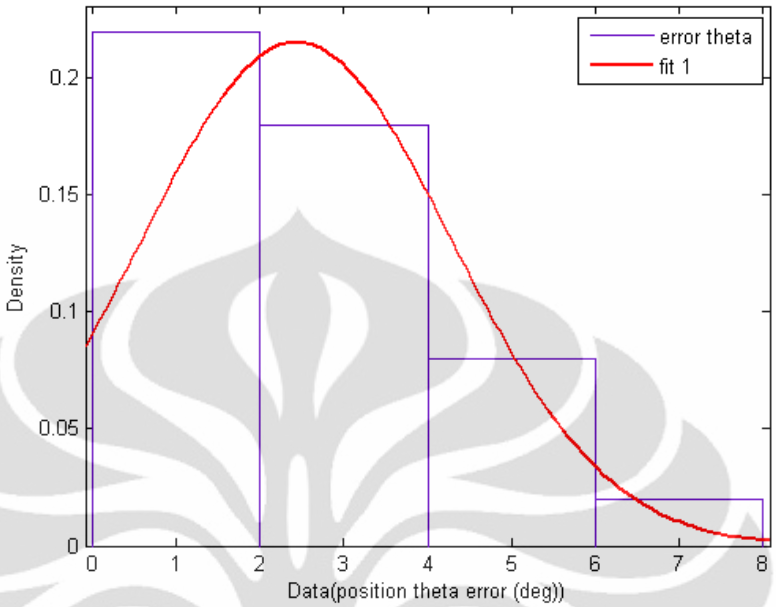
x, y, dan orientasi dapat dilihat masing-masing pada gambar 4.12, 4.13 dan 4.14



Gambar 4.12 *Fitting* terhadap error posisi x



Gambar 4.13 *Fitting* terhadap error posisi y



Gambar 4.14 *Fitting* terhadap error orientasi

Detail dari hasil *fitting* tersebut dapat dilihat pada tabel 4.2

Tabel 4.2 Detail hasil *normal fitting*

Parameter	Error x	Error y	Error theta
Mean	5.2	5	2.43763
Standar deviasi	4.29998	4.25705	1.85123
Variansi	18.4898	18.1224	3.42706

Berdasarkan parameter yang diperoleh dari *fitting*, didapatlah parameter *noise covariance* untuk posisi x, y dan orientasi (θ) sebagai berikut:

1. *Covariance* x = 18.49
2. *Covariance* y = 18.12
3. *Covariance* θ = 3.43

Dengan demikian, matriks Q yang digunakan memiliki bentuk seperti tertera pada (4.54)

$$\boldsymbol{Q} = \begin{bmatrix} 18.49 & 0 & 0 \\ 0 & 18.12 & 0 \\ 0 & 0 & 3.43 \end{bmatrix} \qquad (4.54)$$

Nilai matriks $\boldsymbol{Q}$ ini merupakan asumsi awal. Pada eksperimen, nilai ini akan diubah-ubah untuk mencari nilai yang memberikan hasil optimal.

Matriks ***R*** merupakan matriks *measurement* noise covariance. Matriks ini mengandung ukuran ketidakpastian vektor *measurement*. Penentuan nilai matriks ini akan dibahas pada subbab-subbab berikutnya karena setiap strategi lokalisasi mempunyai vektor *measurement* yang berbeda.

Selain menentukan nilai matriks ***P***, ***Q***, dan ***R***, hal lain yang perlu dilakukan adalah melakukan kalibrasi terhadap sensor, terutama sensor orientasi, CMPS03. Hasil pembacaan sensor ini dipengaruhi oleh lingkungan sekitar, sehingga memberikan hasil yang berbeda untuk setiap titik pengukuran pada lingkungan uji. Untuk itu dicarilah sebuah persamaan yang dapat digunakan untuk memperbesar tingkat akurasi data orientasi. Persamaan ini didapat dengan mengambil data 4 arah utama (90º, 180º, 270º, 360º) di 9 titik pada lingkungan uji. Data yang didapat disajikan dalam tabel 4.3

Tabel 4.3 Data pembacaan kompas di 9 titik untuk empat arah utama

Poin ke	Arah sebenarnya (derajat)				
	90	180	270	360 atau 0	
1	80.5	166.5	258.3	351.5	data sensor
2	82.1	167.8	259.8	534.1	
3	85.8	172.5	262.6	359.7	
4	83.5	170.9	261.5	357.1	
5	88.1	171.9	265.5	2.9	
6	85.1	170.9	261.6	357.8	
7	82.8	168.2	260.8	357.2	
8	78.9	167.2	257.9	352.4	
9	83.1	168.3	260.8	354.9	

Berikutnya dicari rata-rata dari keempat arah dari data yang didapat. Perlu menjadi perhatian di sini, terdapat nilai yang memiliki selisih cukup besar, yaitu pada poin 5 untuk arah 360º. Data yang didapat dari sensor adalah 2.9 º. Nilai ini perlu dikonversi terlebih dahulu, dengan cara dijumlahkan dengan 360º. Hal ini dilakukan untuk menghindari kesalahan dalam pencarian rata-rata. Setelah didapat nilai rata-rata untuk keempat arah, maka dicari persamaan yang menghubungkan nilai

bacaan sensor dengan nilai orientasi asli. Persamaan yang diperoleh dapat dilihat pada (4.55)

$$\theta_{true} = -0.0002795\theta_{sense}^2 + 1.111\theta_{sense} - 0.4712 \quad (4.55)$$

4.4.2 Strategi *Line Extraction*

Skema *correction step* menggunakan *line extraction* biasanya diterapkan menggunakan sensor laser scanner [21] [33] [29] [34]. Sensor ini dapat mengukur jarak dari beberapa titik di sekitar robot terhadap robot sekaligus. Dengan demikian, akan tersedia banyak poin yang dapat diproses menjadi garis. Garis hasil proses ini akan memiliki parameter jarak dan sudut pada referensi lokal robot.

Penerapan skema *line extraction* untuk *mobile robot* dengan sumber daya sensor minimal membutuhkan suatu penanganan khusus. Pada penelitian ini diputuskan untuk menggunakan strategi sebagai berikut untuk mendapatkan poin-poin yang dapat dioleh menjadi garis:

1. Hasil pembacaan sensor jarak diolah menjadi poin-poin koordinat pada kerangka referensi global, bukan kerangka referensi lokal seperti yang dilakukan pada skema line etraction konvensional. Persamaan yang digunakan ditunjukkan oleh (4.56) dan (4.57)

$$x_f = \hat{x}_{r,k}^- + d\cos(\hat{\theta}_{r,k}^- + \alpha) \quad (4.56)$$

$$y_f = \hat{y}_{r,k}^- + d\sin(\hat{\theta}_{r,k}^- + \alpha) \quad (4.57)$$

   dimana d adalah jarak dari *mobile robot* ke suatu titik; hasil pembacaan sensor jarak, sedangkan α adalah sudut pembacaan jarak

2. Untuk mendapatkan poin yang cukup untuk diolah menjadi garis, maka *mobile robot* akan berjalan maju tiga kali dan mengambil data jarak setiap selesai bergerak maju. Jarak maju *mobile robot* ditentukan 10 cm.
3. Poin yang telah didapat kemudian dianalisa. Jika terdapat poin yang memiliki jarak melebihi threshold dengan poin lain, maka poin tersebut diabaikan. Threshold yang dipilih adalah 20 cm.

Alasan pemilihan nilai ini adalah jarak ideal antar tiap poin adalah sama dengan jarak maju *mobile robot*, yaitu 10 cm. Dari nilai tersebut kemudian ditambahkan nilai *offset* 10 cm untuk mengantisipasi kemungkinan pembacaan yang kurang akurat

Poin yang telah didapat kemudian diolah menjadi garis menggunakan linear regression [34]. Pertama dihitung parameter regresi dengan menggunakan (4.58)

$$R_x = \sum_{i=1}^{n} x_{fi},\ R_y = \sum_{i=1}^{n} y_{fi},\ R_{xx} = \sum_{i=1}^{n} x_{fi}^2,$$
$$R_{yy} = \sum_{i=1}^{n} y_{fi}^2,\ R_{xy} = \sum_{i=1}^{n} x_{fi} y_{fi} \quad (4.58)$$

Berikutnya dihitung parameter N1 dan N2 menggunakan (4.59) dan (4.60)

$$N1 = nR_{xx} - R_x^2 \quad (4.59)$$
$$N2 = nR_{yy} - R_y^2 \quad (4.60)$$

Parameter N1 dan N2 digunakan untuk menentukan apakah poin-poin yang didapat cenderung berkumpul pada sumbu x atau y. Jika N1 lebih besar daripada N2, maka poin-poin yang didapat cenderung berkumpul pada sumbu x, sehingga dilakukan regresi dari y terhadap x ($y = mx + q$). Sebaliknya, jika N2 lebih besar daripada N1, maka dilakukan regresi x terhadap y ($x = sy + t$). Parameter m, q, s, t kemudian dihitung menggunakan (4.61) – (4.62)

$$m = \frac{T}{N1},\ q = \frac{R_y - mR_x}{n} \quad (4.61)$$
$$s = \frac{T}{N2},\ t = \frac{R_x - sR_y}{n} \quad (4.62)$$

Dari parameter m, q, s, t yang didapat, dapat dihitung nilai parameter jarak (p) dan sudut (α) dari garis dengan menggunakan (4.63) dan (4.64)

$$p = \left|\frac{q}{\sqrt{m^2+1}}\right| \ atau\ p = \left|\frac{t}{\sqrt{s^2+1}}\right| \quad (4.63)$$
$$\alpha = \tan^{-1}\left(\frac{q}{-qm}\right) \ atau\ \alpha = \tan^{-1}\left(\frac{-ts}{t}\right) \quad (4.64)$$

Parameter garis yang didapat ini adalah parameter garis pada kerangka referensi global. Parameter ini kemudian digunakan untuk melakukan matching dengan garis yang berada dalam map internal robot. Segmen

garis yang terdeteksi dikatakan match jika memenuhi kriteria sebagai berikut:

1. $|p_r - p_{map}| < threshold$, dimana p_r adalah parameter jarak garis yang didapat dari proses *line extraction* dan p_{map}adalah parameter jarak garis yang tersimpan dalam map internal robot. Pada penelitian ini, threshold yang digunakan adalah 200 mm
2. $|\alpha_r - \alpha_{map}| < threshold$, dimana α_r adalah parameter sudut garis yang didapat dari proses *line extraction* dan α_{map}adalah parameter sudut garis yang tersimpan dalam map internal robot. Pada penelitian ini, threshold yang digunakan adalah 25º.

Segmen garis yang terdeteksi dan bagian garis pada map yang match dengan segmen tersebut kemudian ditransformasikan ke kerangka referensi lokal menggunakan (4.65) dan

$$\begin{bmatrix} \hat{r}_i \\ \hat{\psi}_i \end{bmatrix} = \begin{bmatrix} |C_j| \\ \alpha_j - \left(\hat{\theta}_{r,k}^- - \frac{\pi}{2}\right) + \left(-0.5 sign(C_j) + 0.5\right)\pi \end{bmatrix} \quad (4.65)$$

$$C_j = p_j - \hat{x}_{r,k}^- \cos\hat{\theta}_{r,k}^- - \hat{y}_{r,k}^- \sin\hat{\theta}_{r,k}^-$$

Dari parameter yang telah ditransformasi tersebut dapat dibentuk vektor *measurement* seperti dijelaskan pada (4.66)

$$\boldsymbol{z} = \begin{bmatrix} \hat{r}_{r,i} \\ \theta_{\boldsymbol{c}} \end{bmatrix} \quad (4.66)$$

dimana $\hat{r}_{r,i}$ adalah parameter jarak garis ke i yang terdeteksi oleh robot, sedangkan $\theta_{\boldsymbol{c}}$ adalah data orientasi yang dihasilkan oleh sensor CMPS03

Vektor *measurement* yang terlihat pada (4.66) memiliki sedikit perbedaan dibandingkan dengan yang dijelaskan pada [6]. Pada (4.66) digunakan data CMPS03 sebagai salah satu elemen vektor *measurement*. Selain itu, parameter garis berupa sudut (α) tidak dilibatkan dalam vektor *measurement*. Dengan demikian fungsi *measurement prediction*, *h*(.) yang digunakan pun mengalami perubahan menjadi seperti ditunjukkan oleh (4.67)

$$h(\hat{\boldsymbol{x}}_{r,k}^-) = \begin{bmatrix} |C_j| \\ \hat{\theta}_{r,k}^- \end{bmatrix} \quad (4.67)$$

$$C_j = p_{m,j} - \hat{x}_{r,k}^{-} \cos\hat{\theta}_{r,k}^{-} - \hat{y}_{r,k}^{-} \sin\hat{\theta}_{r,k}^{-}$$

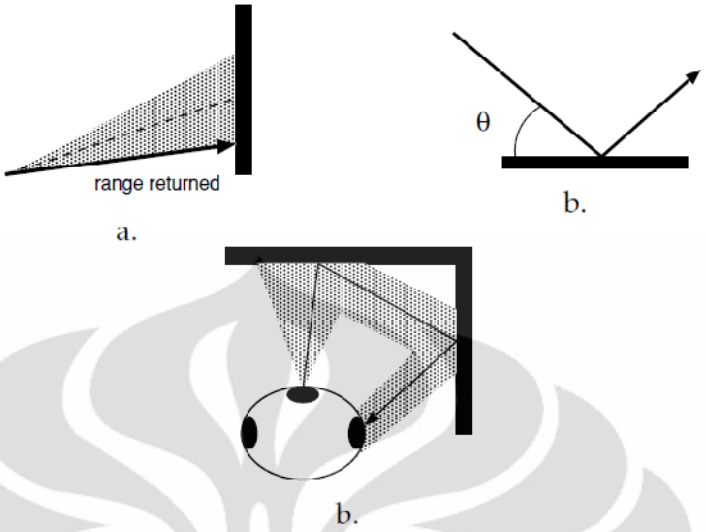
dimana $p_{m,j}$ adalah parameter jarak garis ke j yang terdapat dalam map internal robot. Berdasarkan (4.67) kemudian dapat ditentukan matriks $\boldsymbol{H}$ yaitu seperti terlihat pada (4.68)

$$\boldsymbol{H} = \begin{bmatrix} m\cos\alpha_j & m\sin\alpha_j & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$m = \begin{cases} -1, jika\ C_j > 0 \\ 1, jika\ C_j < 0 \end{cases} \quad (4.68)$$

Matriks $\boldsymbol{R}$ yang digunakan untuk skenario *line extraction* ini memiliki bentuk seperti terlihat pada (4.69)

$$\boldsymbol{R} = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 0.000002 \end{bmatrix} \quad (4.69)$$

Matriks $\boldsymbol{R}$ memiliki ukuran 3x3 karena garis yang didapat pada proses *line extraction* berjumlah dua, atau dengan kata lain data yang digunakan untuk melakukan *line extraction* berasal dari sensor jarak sebelah kiri dan sebelah kanan. Elemen R_{11} dan R_{22}, elemen yang bersesuaian dengan parameter jarak garis pada vektor *measurement* bernilai 200. Hal ini dikarenakan proses *line extraction* memanfaatkan poin yang didapat dari data sensor PING. Data sensor ini memiliki ketidakpastian yang cukup besar karena adanya fenomena foreshortening, specular reflection dan cross-talk [35]. Fenomena-fenomena tersebut menyebabkan jarak yang terbaca oleh sensor PING menjadi lebih besar atau lebih kecil. Ilustrasi fenomena-fenomena tersebut dapat diihat pada gambar 4.15. Untuk elemen R_{33}, elemen ini bernilai 0.000002 karena elemen ini bersesuaian dengan data CMPS03 dan berdasarkan percobaan, ketidakpastian sensor ini kecil.



Gambar 4.15 Ilustrasi permasalahan yang dihadapi sensor sonar [35]

Akhirnya, persamaan-persamaan yang digunakan dalam strategi lokalisasi menggunakan *line extraction* dapat dirangkum dalam tabel 4.4

Tabel 4.4 Persamaan-persamaan yang digunakan dalam strategi *line extraction*

Time Update ("Prediction") Equations	
$\hat{\boldsymbol{x}}_{r,k}^{-} = f(\hat{\boldsymbol{x}}_{r,k-1}, \boldsymbol{u}_{r,k-1})$	(3.48)
$f(\hat{\boldsymbol{x}}_{r,k-1}^{-}) = \begin{bmatrix} \hat{x}_{r,k-1}^{-} + \Delta S \cos\hat{\theta}_{r,k-1}^{-} \\ \hat{y}_{r,k-1}^{-} + \Delta S \sin\hat{\theta}_{r,k-1}^{-} \\ \hat{\theta}_{r,k-1}^{-} + \Delta\theta \end{bmatrix}$	(4.51)
$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & \Delta S \sin\hat{\theta}_{r,k-1}^{-} \\ 0 & 1 & \Delta S \cos\hat{\theta}_{r,k}^{-} \\ 0 & 0 & 1 \end{bmatrix}$	(4.52)
$\boldsymbol{P}_k^{-} = \boldsymbol{F}\boldsymbol{P}_k\boldsymbol{F}^T + \boldsymbol{Q}$	(3.49)
Measurement Update ("Correction") Equations	
$\boldsymbol{K}_k = \boldsymbol{P}_k^{-}\boldsymbol{H}^T[\boldsymbol{H}\boldsymbol{P}_k^{-}\boldsymbol{H}^T + \boldsymbol{R}]^{-1}$	(3.50)
$\hat{\boldsymbol{x}}_{r,k} = \hat{\boldsymbol{x}}_{r,k}^{-} + \boldsymbol{K}_k(\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{r,k}^{-}, 0))$	(3.51)
$\boldsymbol{z}_k = \begin{bmatrix} \hat{r}_{r,i} \\ \theta_c \end{bmatrix}$	(4.66)

$$h(\hat{\boldsymbol{x}}_{r,k}^{-}) = \begin{bmatrix} |C_j| \\ \hat{\theta}_{r,k}^{-} \end{bmatrix} \quad (4.67)$$

$$C_j = p_{m,j} - \hat{x}_{r,k}^{-} \cos\hat{\theta}_{r,k}^{-} - \hat{y}_{r,k}^{-} \sin\hat{\theta}_{r,k}^{-}$$

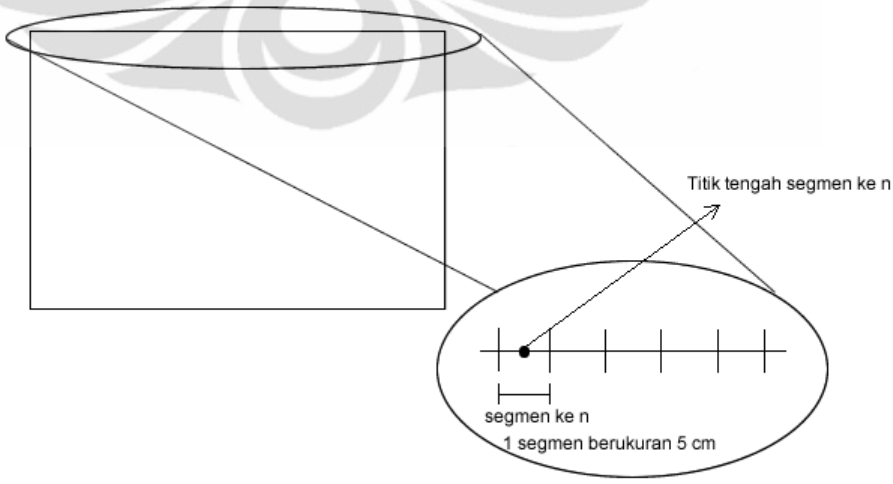
$$\boldsymbol{H} = \begin{bmatrix} m\cos\alpha_j & m\sin\alpha_j & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$m = \begin{cases} -1, jika\ C_j > 0 \\ 1, jika\ C_j < 0 \end{cases} \quad (4.68)$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^{-} \quad (3.52)$$

4.4.3 Strategi *Landmark Detection*

Seperti telah dijelaskan pada subbab 3.5, yang dimaksud *landmark* pada lokalisasi *mobile robot* adalah poin pada lingkungan sekitar robot yang dapat digunakan untuk menentukan posisi. Untuk strategi lokalisasi menggunakan *landmark detection* ini, yang digunakan sebagai *landmark* adalah poin-poin pada dinding lingkungan uji. Dinding pada lingkungan uji dianggap sebagai garis, kemudian garis tersebut dipartisi menjadi segmen-segmen dengan ukuran tertentu. Segmen inilah yang dianggap *landmark* dan digunakan untuk mengoreksi *pose* robot. Posisi suatu *landmark* dianggap berada pada titik tengah segmen yang bersangkutan. Ilustrasi dari partisi dinding yang menghasilkan *landmark* ini dapat dilihat pada gambar 4.16



Gambar 4.16 Ilustrasi partisi dinding

Pada penelitian ini ukuran segmen yang digunakan adalah 5 cm. Dengan demikian berarti total terdapat 112 *landmark* dari empat dinding yang mengelilingi lingkungan uji.

Langkah awal strategi *landmark detection* ini adalah mentransformasi data jarak yang didapat dari sensor sonar PING menjadi poin dengan koordinat tertentu pada kerangka referensi global. Transformasi data jarak tersebut dilakukan dengan menggunakan (4.70) dan (4.71)

$$x_f = \hat{x}_{r,k}^- + d_i \cos(\hat{\theta}_{r,k}^- + \alpha) \quad (4.70)$$

$$y_f = \hat{y}_{r,k}^- + d_i \sin(\hat{\theta}_{r,k}^- + \alpha) \quad (4.71)$$

Selain ditransformasikan menjadi poin-poin untuk dicocokkan dengan koordinat *landmark*, data dari sensor PING juga digunakan untuk membentuk vektor *measurement* bersama data dari kompas. Vektor *measurement* ini kemudian memiliki bentuk yang dapat dilihat pada (4.72)

$$\boldsymbol{z} = \begin{bmatrix} d_i \\ \theta_c \end{bmatrix} \quad (4.72)$$

dimana d_i adalah hasil pembacaan PING ke i dan θ_c adalah data orientasi hasil pembacaan kompas.

Poin-poin yang didapat dari (4.70) dan (4.71) kemudian dicocokkan dengan koordinat *landmark* yang tersimpan dalam map internal robot. Dari pencocokan tersebut kemudian diketahui *landmark* mana yang terdeteksi oleh robot dan koordinatnya. Kriteria kecocokan yang digunakan adalah sebagai berikut:

1. $|x_{land} - x_f| \leq 25\ dan\ \ |y_{land} - y_f| \leq 100$ untuk *landmark-landmark* yang didapat dari segmentasi dinding yang sejajar dengan sumbu x
2. $|y_{land} - y_f| \leq 25\ dan\ \ |x_{land} - x_f| \leq 100$ untuk *landmark-landmark* yang didapat dari segmentasi dinding yang sejajar dengan sumbu y

Pada saat koordinat poin tidak cocok dengan koordinat *landmark* manapun, maka koordinat *landmark* di set sesuai dengan koordinat poin yang didapat. Dengan kata lain, pada kondisi ini berlaku (4.73)

$$x_{land} = x_f$$
$$y_{land} = y_f \tag{4.73}$$

Koordinat poin yang memenuhi kriteria kemudian digunakan dalam persamaan *measurement prediction*, *h*(.), seperti terlihat pada (4.74)

$$h(\hat{\boldsymbol{x}}_{r,k}^-) = \begin{bmatrix} \sqrt{(x_{land,i} - \hat{x}_{r,k}^-)^2 + (y_{land,i} - \hat{y}_{r,k}^-)^2} \\ \hat{\theta}_{r,k}^- \end{bmatrix} \tag{4.74}$$

Matriks jacobian dari *h(.)*, $\boldsymbol{H}$ kemudian dapat diperoleh. Matriks jacobian ini memiliki bentuk seperti terlihat pada (4.75)

$$\boldsymbol{H} = \begin{bmatrix} \frac{-D_{x,i}}{r_i} & \frac{-D_{y,i}}{r_i} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.75}$$
$$D_{x,i} = x_{land,i} - \hat{x}_{r,k}^-$$
$$D_{y,i} = y_{land,i} - \hat{y}_{r,k}^-$$
$$r_i = \sqrt{(x_{land,i} - \hat{x}_{r,k}^-)^2 + (y_{land,i} - \hat{y}_{r,k}^-)^2}$$

Konsekuensi dari penerapan (4.73) jika meninjau (4.74) adalah ketika koordinat landmark yang cocok dengan koordinat poin yang dideteksi tidak ditemukan, maka (4.76) akan berlaku

$$h(\hat{\boldsymbol{x}}_{r,k}^-) = \begin{bmatrix} \sqrt{(x_{land,i} - \hat{x}_{r,k}^-)^2 + (y_{land,i} - \hat{y}_{r,k}^-)^2} \\ \hat{\theta}_{r,k}^- \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{(\hat{x}_{r,k}^- + d\cos(\hat{\theta}_{r,k}^- + \alpha) - \hat{x}_{r,k}^-)^2 + (\hat{y}_{r,k}^- + d\sin(\hat{\theta}_{r,k}^- + \alpha) - \hat{y}_{r,k}^-)^2} \\ \hat{\theta}_{r,k}^- \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{(d\cos(\hat{\theta}_{r,k}^- + \alpha))^2 + (d\sin(\hat{\theta}_{r,k}^- + \alpha))^2} \\ \hat{\theta}_{r,k}^- \end{bmatrix}$$

$$= \begin{bmatrix} d_i \\ \hat{\theta}_{r,k}^- \end{bmatrix} \tag{4.76}$$

Dengan demikian, nilai residual ($\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{r,k}^-, 0)$) untuk elemen d_i akan bernilai 0, sehingga *state* x dan y tidak akan mengalami koreksi. Maka dapat dikatakan, bahwa ketika terjadi keadaan dimana *landmark* tidak dapat diperoleh, maka tidak akan terjadi koreksi posisi (x,y).

Parameter matriks $\boldsymbol{R}$ yang digunakan untuk strategi *landmark detection* ini ditunjukkan oleh (4.76)

$$\boldsymbol{R} = \begin{bmatrix} 200 & 0 & 0 & 0 & 0 \\ 0 & 200 & 0 & 0 & 0 \\ 0 & 0 & 200 & 0 & 0 \\ 0 & 0 & 0 & 200 & 0 \\ 0 & 0 & 0 & 0 & 0.000002 \end{bmatrix} \quad (4.77)$$

Matriks yang digunakan berukuran 5x5 karena ada 4 *landmark* yang dapat dideteksi pada satu waktu. Elemen R_{11}, R_{22}, R_{33}, dan R_{44} bersesuaian dengan elemen vektor *measurement* yang berasal dari data PING. Elemen-elemen ini diberi nilai sebesar 200 karena ketidakpastian pembacaan PING yang besar. Elemen R_{55} diberi nilai 0.000002 karena elemen ini bersesuaian dengan elemen vektor *measurement* yang berisikan data kompas, dan data kompas ini memiliki ketidakpastian yang kecil.

Akhirnya, persamaan-persamaan yang digunakan dalam strategi lokalisasi menggunakan *landmark detection* dapat dirangkum dalam tabel 4.5

Tabel 4.5 Persamaan–persamaan yang digunakan pada strategi *landmark detection*

Time Update ("Prediction") Equations	
$\hat{\boldsymbol{x}}_{r,k}^- = f(\hat{\boldsymbol{x}}_{r,k-1}, \boldsymbol{u}_{r,k-1})$	(3.48)
$f(\hat{\boldsymbol{x}}_{r,k-1}^-) = \begin{bmatrix} \hat{x}_{r,k-1}^- + \Delta S \cos\hat{\theta}_{r,k-1}^- \\ \hat{y}_{r,k-1}^- + \Delta S \sin\hat{\theta}_{r,k-1}^- \\ \hat{\theta}_{r,k-1}^- + \Delta\theta \end{bmatrix}$	(4.51)
$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & \Delta S \sin\hat{\theta}_{r,k-1}^- \\ 0 & 1 & \Delta S \cos\hat{\theta}_{r,k}^- \\ 0 & 0 & 1 \end{bmatrix}$	(4.52)
$\boldsymbol{P}_k^- = \boldsymbol{F}\boldsymbol{P}_k\boldsymbol{F}^T + \boldsymbol{Q}$	(3.49)

Measurement Update ("Correction") Equations

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^T [\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{R}]^{-1} \quad (3.50)$$

$$\hat{\boldsymbol{x}}_{r,k} = \hat{\boldsymbol{x}}_{r,k}^- + \boldsymbol{K}_k(\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{r,k}^-, 0)) \quad (3.51)$$

$$\boldsymbol{z}_k = \begin{bmatrix} d_i \\ \theta_c \end{bmatrix} \quad (4.72)$$

$$h\left(\hat{\boldsymbol{x}}_{r,k}^-\right) = \begin{bmatrix} \sqrt{(x_{land,i} - \hat{x}_{r,k}^-)^2 + (y_{land,i} - \hat{y}_{r,k}^-)^2} \\ \hat{\theta}_{r,k}^- \end{bmatrix} \quad (4.74)$$

$$\boldsymbol{H} = \begin{bmatrix} \frac{-D_{x,i}}{r_i} & \frac{-D_{y,i}}{r_i} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.75)$$

$$D_{x,i} = x_{land,i} - \hat{x}_{r,k}^-$$

$$D_{y,i} = y_{land,i} - \hat{y}_{r,k}^-$$

$$r_i = \sqrt{(x_{land,i} - \hat{x}_{r,k}^-)^2 + (y_{land,i} - \hat{y}_{r,k}^-)^2}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H})\boldsymbol{P}_k^- \quad (3.52)$$

BAB 5
PERANCANGAN SISTEM VALIDASI

5.1 Lingkungan Uji *Mobile Robot*

Sistem validasi yang digunakan untuk menguji algoritma *mobile robot* terbagi menjadi dua bagian. Yang pertama adalah lingkungan uji *mobile robot*. Berikutnya adalah sistem penjejak *pose* absolut (absolute *pose tracke*r). Sistem penjejak *pose* absolut ini diperlukan untuk mendapatkan posisi aktual/nyata *mobile robot* pada lingkungan uji, sehingga akurasi algoritma lokalisasi dapat diuji. Sistem penjejak *pose* absolut ini juga dapat menggantikan alat ukur manual dan mewujudkan otomatisasi pengukuran *pose* aktual. Umumnya, sistem ini dibangun dengan menggunakan kamera sebagai elemen utama.

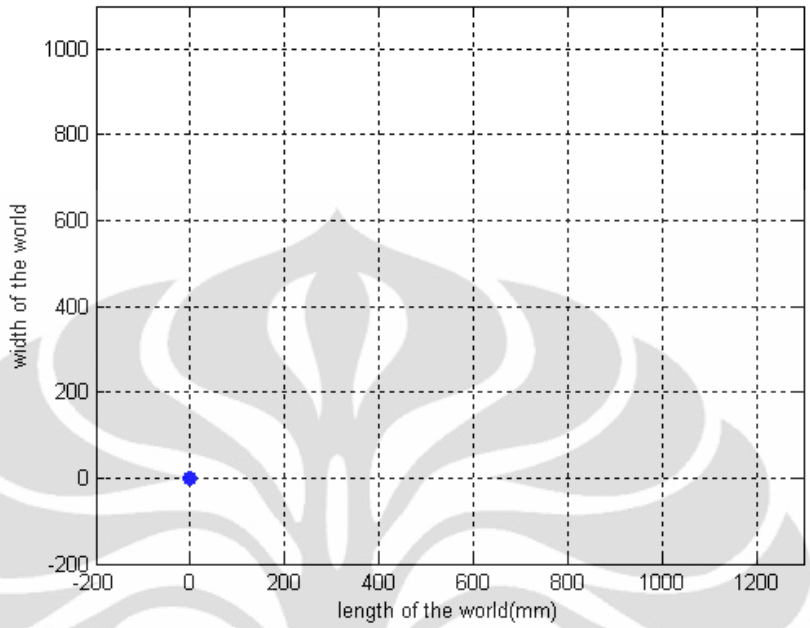
Bab ini akan menjelaskan tentang lingkungan yang digunakan pada penelitian ini untuk menguji algoritma lokalisasi *mobile robot*. Selain itu, bab ini juga akan menjelaskan tentang sistem penjejak *pose* absolut yang dirancang, meliputi hardware yang digunakan dan proses pengolahan citra yang dilakukan untuk mendapatkan *pose* aktual *mobile robot*.

Lingkungan yang digunakan untuk pengujian *mobile robot* ini memiliki dua bentuk. Pada kedua bentuk tersebut, digunakan bahan plastik sebagai dinding pembatas. Bentuk pertama adalah persegi panjang berukuran 1.5x1.3 m. Lingkungan uji ini kemudian disebut sebagai tipe 1 untuk memudahkan penyebutan. Bentuk lingkungan kedua menyerupai huruf "L", dan kemudian akan disebut sebagai tipe 2. Permukaan lantai yang digunakan dilapisi oleh karpet untuk memberikan gesekan yang tepat bagi *mobile robot* sehingga dapat bergerak. Lingkungan uji yang digunakan bersifat statis, dalam artian tidak berubah terhadap waktu. Lingkungan uji yang digunakan dapat dilihat pada gambar 5.1 dan 5.2

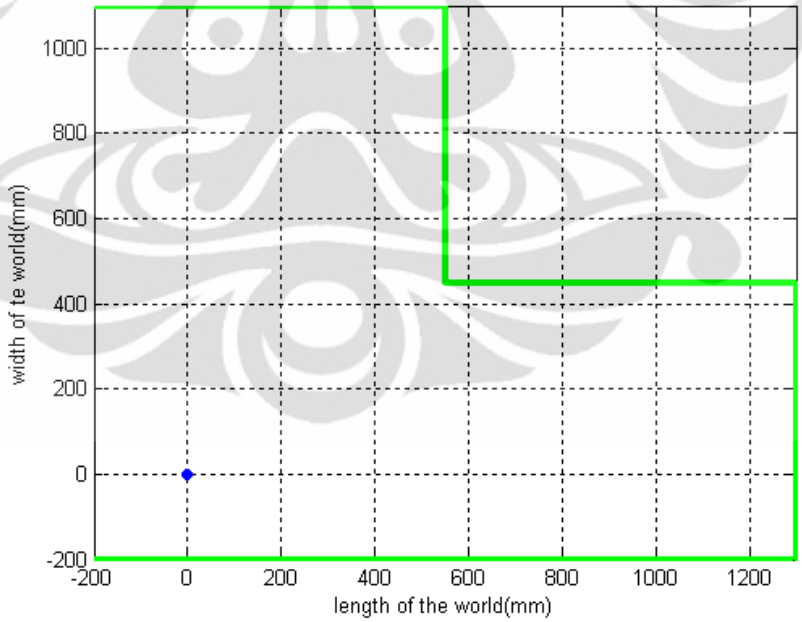
Gambar 5.1 Lingkungan uji yang digunakan (tipe 1)

Gambar 5.2 Lingkungan uji yang digunakan (tipe 2)

Lingkungan uji ini memiliki *origin* yang ditentukan berada pada titik yang berjarak 20 cm dari sisi bawah dan 20 cm dari sisi kiri. Ilustrasi *origin* dan sistem koordinat yang diterapkan pada lingkungan uji dapat dilihat pada gambar 5.3 dan 5.4



Gambar 5.3 *Origin* dan sistem koordinat pada lingkungan uji (tipe 1)



Gambar 5.4 *Origin* dan sistem koordinat pada lingkungn uji (tipe 2)

5.2 Sistem Penjejak *Pose* Absolut

5.2.1 Hardware dan Kalibrasinya

Hardware yang digunakan untuk sistem penjejak *pose* absolut ini adalah webcam berjenis Logitceh Webcam C120. Perangkat webcam ini diletakkan di atas lingkungan uji dengan ketinggian 2.5 m sehingga didapat tampilan seperti terlihat pada gambar 5.5

Gambar 5.5 Hasil pengambilan gambar

Sebelum dilakukan pengolahan citra untuk mendapatkan *pose* aktual, pertama dilakukan transformasi koordinat dari camera-frame menjadi world-frame. Pada camera frame, *origin* berada di tepat di pojok kiri atas citra, sedangkan pada world frame, *origin* berada pada titik yang berjarak 20 cm dari sisi kiri dan 20 cm dari sisi bawah. Selain itu, juga perlu dilakukan konversi, karena satuan pada camera frame adalah piksel, sedangkan satuan pada world frame adalah milimeter. Konversi dan transformasi koordinat dilakukan dengan menggunakan (5.1) dan (5.2)

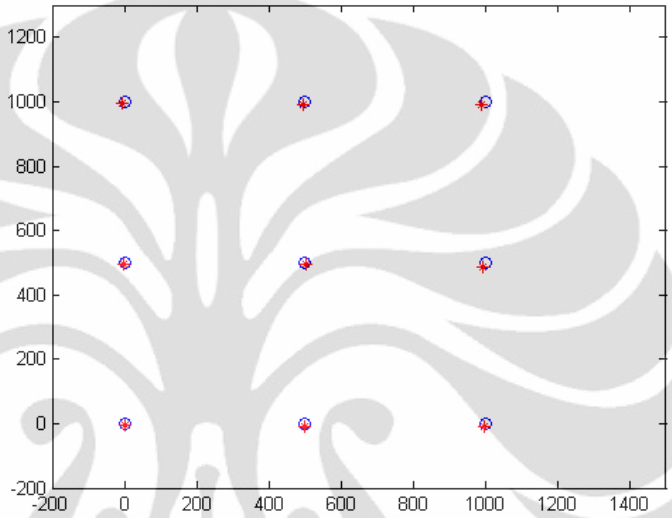
$$x_{mm} = (x_{pixel} - offset_x) \times conversion\ factor \quad (5.1)$$

$$y_{mm} = (lower\ bound - y_{pixel}) \times conversion\ factor \quad (5.2)$$

Berdasarkan pengukuran, didapatlah :

- $offset_x$ = 117
- $lower\ bound$ = 396
- $conversion\ factor$ = 3.0150754 mm/piksel

Hasil konversi ini kemudian diuji menggunakan 9 titik. Hasil pengujian dapat dilihat pada gambar 5.6



Gambar 5.6 Titik pengujian hasil konversi

Error dari hasil konversi ini adalah 1.2 mm untuk posisi x dan 4.8 mm untuk posisi y. Dengan demikian, konversi ini sudah cukup baik untuk digunakan sebagai dasar pada tahap berikutnya, yaitu pendeteksian *pose*.

5.2.2 Pendeteksian *Pose*

Proses pendeteksian *pose* dilakukan dengan cara mengenali dan mendeteksi *mobile robot*. Setelah itu, barulah dilakukan perhitungan untuk menentukan posisi dan orientasi robot. Untuk mempermudah proses pendeteksian, maka diberikan pattern pada *mobile robot* untuk membedakannya dari lingkungan sekitar. Pattern yang digunakan pada penelitian ini adalah tiga lingkaran putih yang disusun membentuk segitiga pada background hitam. Bentuk fisik pattern yang digunakan dapat dilihat pada gambar 5.7

Gambar 5.7 Pattern yang digunakan

Pattern diletakkan pada *mobile robot* sedemikian sehingga titik tengah alas segitiga berhimpit dengan titik pusat robot. Dengan demikian, penentuan posisi robot dapat dilakukan dengan menentukan titik tengah alas segitiga. Hal ini dilakukan untuk mempermudah perhitungan.

Pada gambar 5.7, terlihat bahwa pattern yang membentuk titik puncak segitiga memiliki ukuran lebih kecil dibandingkan pattern lain. Pattern ini dibuat memiliki ukuran lebih kecil untuk memberi tanda arah depan. Dengan demikian, perhitungan orientasi robot dapat dilakukan dengan menghitung sudut antara sumbu x dan garis yang dibentuk titik pattern arah depan dan titik tengah alas segitiga.

Proses pendeteksian pattern dimulai dengan melakukan akuisisi citra (image acquisition). Contoh hasil citra yang diakuisisi dapat dilihat pada gambar 5.8

Gambar 5.8 Hasil akuisisi citra

Langkah berikutnya, dilakukan pelabelan citra bagian pertama. Setelah melewati tahap ini, piksel-piksel pada citra akan memiliki label dengan nilai tersendiri dan memiliki indeks yang bernilai 0 atau 1. Pelabelan ini bertujuan untuk mengelompokkan bagian citra yang memiliki karakteristik warna yang sama. Pengelompokan dilakukan berdasarkan nilai H-S-V (Hue-Saturation-Value) yang dimiliki oleh piksel-piksel pada citra. Piksel yang memiliki karakteristik H-S-V sama akan memiliki label dengan yang nilai yang sama. Selain itu, piksel tersebut akan memiliki indeks bernilai 0. Piksel pada citra yang tidak memenuhi satu pun karakteristik H-S-V yang diinginkan akan memiliki label bernilai 0 dan indeks bernilai 1. Pada penelitian ini, karena pattern yang digunakan berwarna putih, maka batas karakteristik H-S-V yang diinginkan adalah S < 40 dan V >= 250. Dengan demikian, algoritma pelabelan tahap pertama adalah seperti tertera pada gambar 5.9

```
if(sval[ii][jj]<40 &&
vval[ii][jj]>=250)
{
      grey[ii][jj] = 250;
      greyarray[ii][jj] = 0;
}
else
{
      grey[ii][jj]=0;
      greyarray[ii][jj]=1;
}
```

Gambar 5.9 Algoritma pelabelan bagian pertama

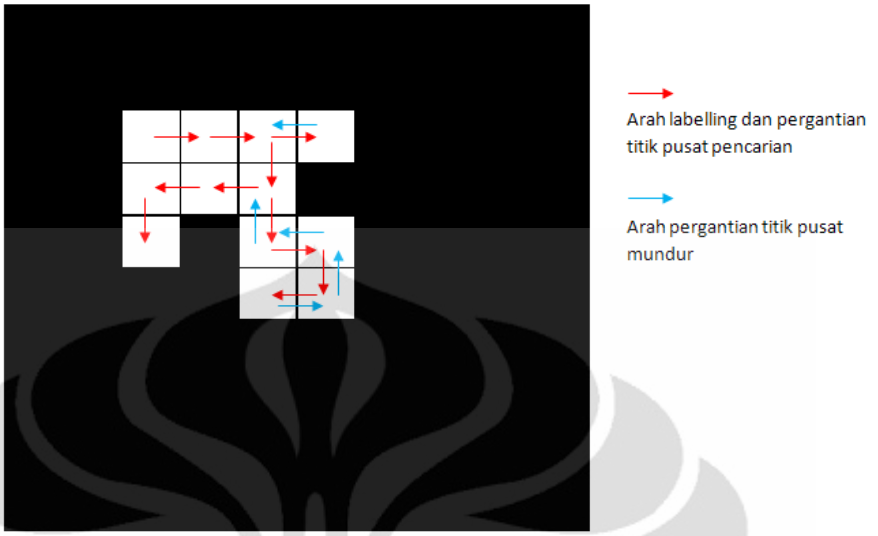
Pada gambar 5.9 terlihat bahwa piksel pada citra yang memiliki karakteristik sesuai dengan warna putih akan diberi label 250 dan indeks bernilai 0, sedangkan piksel dengan karakteristik selain itu akan memiliki label bernilai 0 dan indeks bernilai 1. Contoh citra yang telah melalui pelabelan bagian pertama dapat dilihat pada gambar 5.10

Gambar 5.10 Hasil pelabelan bagian pertama

Tahap berikutnya adalah pelabelan bagian kedua. Tahap ini bertujuan untuk memisahkan bagian citra yang memiliki label sama menjadi objek-objek individual. Langkah-langkah yang dilakukan pada tahap ini adalah:

1. Inisialisasi label untuk semua piksel dengan nilai 0
2. Scan piksel pada citra dari pojok kiri atas sampai pojok kanan bawah
3. Ketika menemukan piksel yang memenuhi kriteria, beri label dengan nilai tertentu. Pada penelitian ini digunakan label 100+n, dimana n menunjukkan objek ke-n yang ditemukan. Kriteria yang digunakan adalah :
    a. Memiliki karakteristik warna putih
    b. Memiliki indeks 0
    c. Label bernilai 0
4. Set piksel yang telah dilabeli sebagai titik pusat pencarian
5. Cek tetangga atas piksel pusat pencarian. Jika memenuhi kriteria, kembali ke langkah 2. Jika tidak, lanjut ke langkah 5
6. Cek tetangga kanan piksel pusat pencarian. Jika memenuhi kriteria, kembali ke langkah 2. Jika tidak, lanjut ke langkah 6
7. Cek tetangga bawah piksel pusat pencarian. Jika memenuhi kriteria, kembali ke langkah 2. Jika tidak, lanjut ke langkah 7
8. Cek tetangga kiri piksel pusat pencarian. Jika memenuhi kriteria, kembali ke langkah 2. Jika tidak, lanjut ke langkah 8
9. Mundurkan titik pusat pencarian. Jika titik pusat pencarian bukan titik awal, kembali ke langkah 2. Jika titik pusat pencarian adalah titik awal, tingkatkan jumlah n sebesar 1, kembali ke tahap 1, lanjutkan proses scanning di titik terakhir scanning dijalankan

Ilustrasi proses pelabelan bagian kedua dapat dilihat pada gambar 5.11



Gambar 5.11 Ilustrasi pelabelan bagian kedua

Setelah tahap ini selesai, suatu objek akan memiliki piksel dengan label yang bernilai sama, sehingga antara satu objek dengan objek lain dapat dibedakan. Pada implementasinya, pencarian tetangga piksel dilakukan hingga piksel yang berjarak 5 piksel dari titik pusat untuk menghindari kesalahan pelabelan ketika pada satu objek terdapat "lubang", yaitu satu piksel yang memiliki karakteristik warna berbeda. Contoh citra hasil pelabelan bagian kedua dapat dilihat pada gambar 5.12

Gambar 5.12 Hasil pelabelan bagian kedua

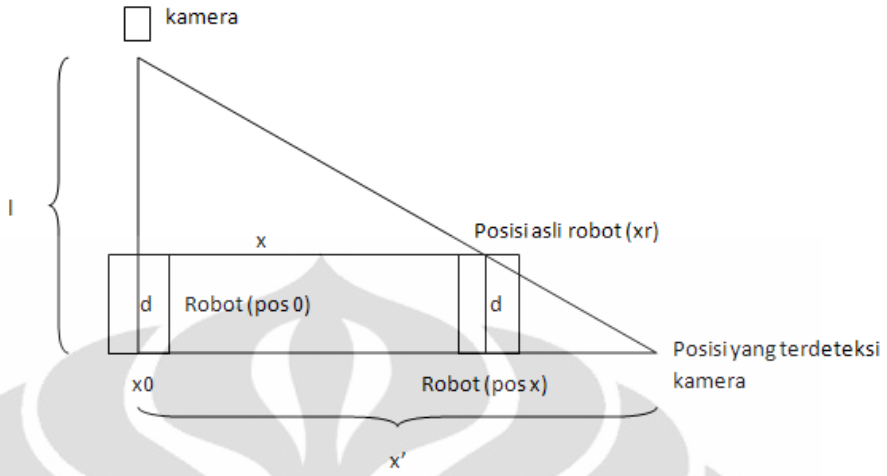
Setelah objek berhasil diidentifikasi, langkah selanjutnya adalah filtering. Pada langkah ini ada beberapa kriteria yang harus dipenuhi bagi suatu objek agar dapat dianggap sebagai pattern. Kriteria tersebut antara lain:

1. Berukuran lebih dari 50 piksel
2. Memiliki tinggi dan lebar lebih dari sama dengan 5 piksel
3. Memiliki selisih tinggi dan lebar kurang dari 5 piksel

Berikutnya, objek yang lolos filtering dihitung titik beratnya. Selanjutnya objek ditentukan mana pattern kiri, kanan dan depan. Pattern kiri dan kanan ditetapkan memiliki range jumlah piksel antara 240-400 piksel, sedangkan pattern depan memiliki range jumlah piksel antara 80-200 piksel.

Langkah selanjutnya, adalah menentukan *pose* robot. Untuk posisi x dan y didapat dari titik tengah antara titik berat pattern kanan dan kiri. Orientasi robot didapat dari sudut yang dibentuk antara garis yang ditarik dari titik berat antara pattern-pattern samping dengan titik berat pattern depan dengan sumbu x. Setelah itu, dilakukan konversi untuk mengubah satuan posisi x dan y dari piksel ke milimeter.

Data *pose* yang diperoleh dari langkah-langkah di atas masih memiliki error. Hal ini terjadi karena untuk posisi robot yang tidak tepat di bawah kamera, tinggi robot akan terdeteksi oleh kamera sehingga menyebabkan posisi robot yang tertangkap kamera bergeser dari aslinya. Ilustrasi geometris dari keadaan ini dapat dilihat pada gambar 5.13



Gambar 5.13 Ilustrasi geometris error posisi yang tertangkap oleh kamera

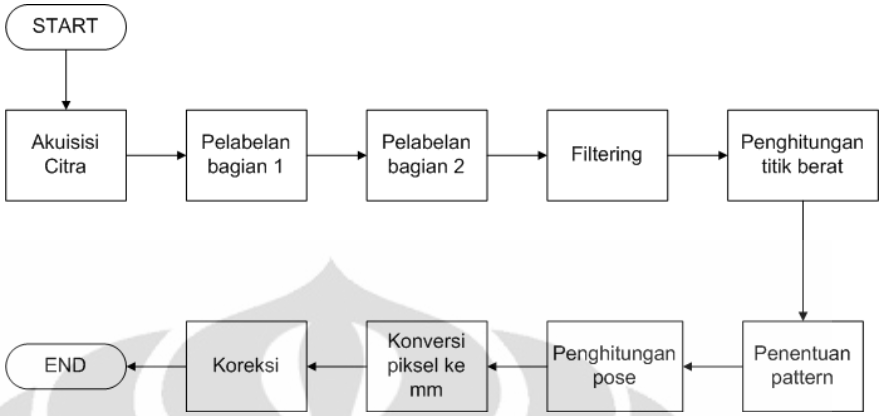
Untuk mendapatkan *pose* asli robot, dilakukanlah koreksi terhadap *pose* yang telah diperoleh. Dengan menggunakan prinsip kesebangunan, diperoleh (5.3)

$$x_r = x_0 + x' - \left(\frac{x' \times d}{l}\right) \qquad (5.3)$$

dimana rumus yang sama juga berlaku untuk posisi y

Nilai x0 dicari dengan menentukan titik yang berada tepat di bawah kamera. Dapat juga dikatakan bahwa titik ini adalah titik dimana posisi yang dideteksi kamera memiliki error yang kecil jika terhadap posisi aktual. Dari pengukuran didapat bahwa titik tersebut adalah titik (690.45, 464.32) mm.

Langkah-langkah pendeteksian *pose* yang telah dijelaskan dapat dirangkum dalam sebuah *flowchart* yang dapat dilihat pada gambar 5.14



Gambar 5.14 *Flowchart* langkah pendeteksian *pose*

Berdasarkan hasil pengujian, didapat bahwa error rata-rata dari sistem absolute *pose track*er ini adalah 0.84 cm untuk posisi x, 1 cm untuk posisi y dan ±5º untuk theta. Dengan demikian, sistem absolute *pose track*er ini dapat digunakan untuk melakukan validasi terhadap hasil lokalisasi.

BAB 6
PENGUJIAN

6.1 Skema Pengujian

Perancangan sistem *mobile robot* mulai dari desain hardware hingga implementasi algoritma lokalisasinya telah dijelaskan pada bab 4. Sistem yang digunakan untuk melakukan validasi terhadap hasil lokalisasi telah dijelaskan pada bab 5. Pada bab ini akan dijelaskan skema pengujian yang dilakukan, hasil yang didapat dan analisanya. Pengujian dilakukan dengan tujuan mengetahui performa masing-masing strategi yang digunakan dan mencari parameter *Extended Kalman Filter* yang dapat memberikan hasil yang paling optimal. Kriteria optimal yang diinginkan adalah:

1. $\overline{err}_{pos} \leq 5cm$
2. $\overline{err}_{\theta} \leq 5°$

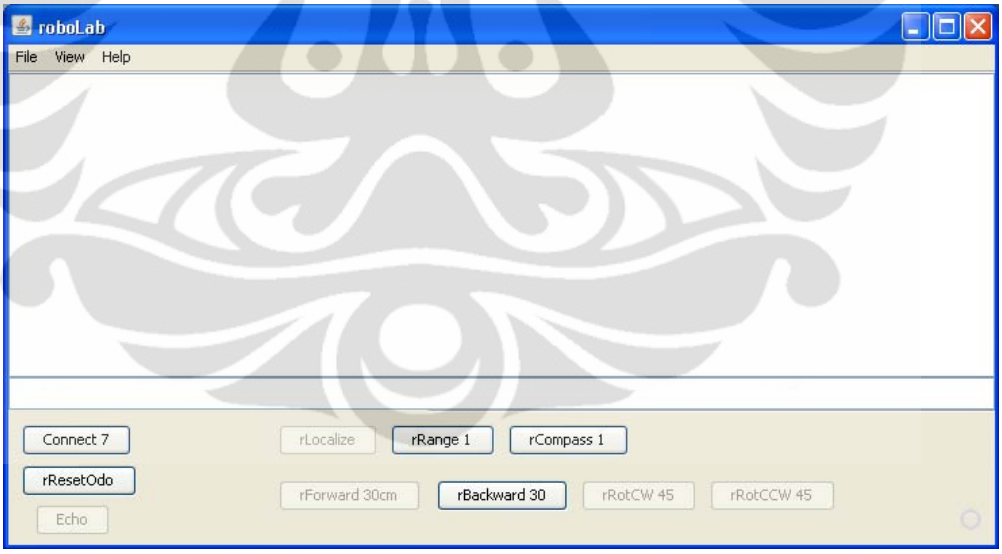
Didefinisikan pula batas acceptable untuk poin-poin secara individual. Untuk poin secara individual, didefinisikan batas *acceptable* adalah memiliki error kurang dari 5cm.

Batas *acceptable* dan kriteria optimal ini didapatkan dengan memperhatikan jarak tempuh maksimal yang mungkin dilakukan oleh robot. Jarak tempuh maksimal yang mungkin dilakukan oleh robot adalah 480cm, dengan demikian nilai error acceptable ini cukup kecil dibandingkan jarak tempuh keseluruhan; 1.04% dari nilai jarak tempuh maksimal. Jika dibandingkan dengan ukuran robot, nilai error *acceptable* ini juga cukup kecil, ¼ dari ukuran robot. Poin-poin yang memenuhi kriteria *acceptable* ini kemudian juga dijadikan kriteria optimal strategi. Strategi dikatakan optimal jika poin yang memenuhi kriteria *acceptable* berjumlah lebih dari 70%.

Pengujian dilakukan dengan menjalankan robot pada lintasan sirkuler berlawanan arah jarum jam. Pengujian ini kemudian dilakukan untuk nilai parameter $\boldsymbol{Q}$ dan $\boldsymbol{R}$ yang berbeda-beda untuk mendapatkan hasil yang optimal. Setelah didapatkan hasil yang optimal, atau mendekati optimal, algoritma diuji dengan menggunakan lintasan yang berbeda, yaitu lintasan sirkuler searah jarum jam dan lintasan bebas yang mengandung unsur arah berlawanan dan searah jarum jam.

Pengujian berikutnya adalah menguji strategi lokalisasi pada lingkungan uji tipe 2. Untuk pengujian ini, strategi yang diuji adalah strategi yang telah mencapai hasil yang optimal. Pengujian ini dilakukan untuk mengetahui kehandalan strategi pada bentuk lingkungan uji yang berbeda

Pada pengujian digunakan komputer untuk memberikan perintah pada robot, melakukan akuisisi data lokalisasi hasil perhitungan algoritma dan melakukan pengolahan citra untuk memvalidasi hasil lokalisasi. Pengujian dilakukan dengan menggunakan aplikasi bernama roboLab [5]. Aplikasi ini adalah aplikasi berbasis Java yang dikembangkan oleh Vektor Dewanto dan Erik DT. Dari aplikasi ini, user dapat memerintahkan robot untuk melakukan gerakan maju, mundur, berputar searah dan berlawanan arah jarum jam. Selain itu, user juga dapat memerintahkan robot untuk mendapatkan data dari sensor dan mengirimnya ke komputer, melakukan algoritma lokalisasi dan mengolah data yang didapat dari sensor. Tampilan aplikasi roboLab dapat dilihat pada gambar 6.1



Gambar 6.1 Tampilan aplikasi roboLab

Lebih jelasnya, untuk strategi *landmark detection*, prosedur pengujian yang dilakukan memiliki langkah-langkah sebagai berikut :

1. Berikan perintah pada robot untuk bergerak
2. Lakukan lokalisasi
3. Validasi *pose* dengan data *pose* absolut yang didapat dari sistem absolute *pose track*er

4. Ulangi langkah 1 hingga didapat data yang cukup

Untuk strategi *line extraction*, prosedur pengujian yang dilakukan memiliki langkah-langkah sebagai berikut :

1. Perintahkan robot untuk melakukan langkah *line extraction*
2. Lakukan lokalisasi
3. Validasi *pose* dengan data *pose* absolut yang didapat dari sistem absolute *pose tracke*r
4. Ulangi langkah 1 hingga didapat data yang cukup

6.2 Hasil Pengujian dan Analisa

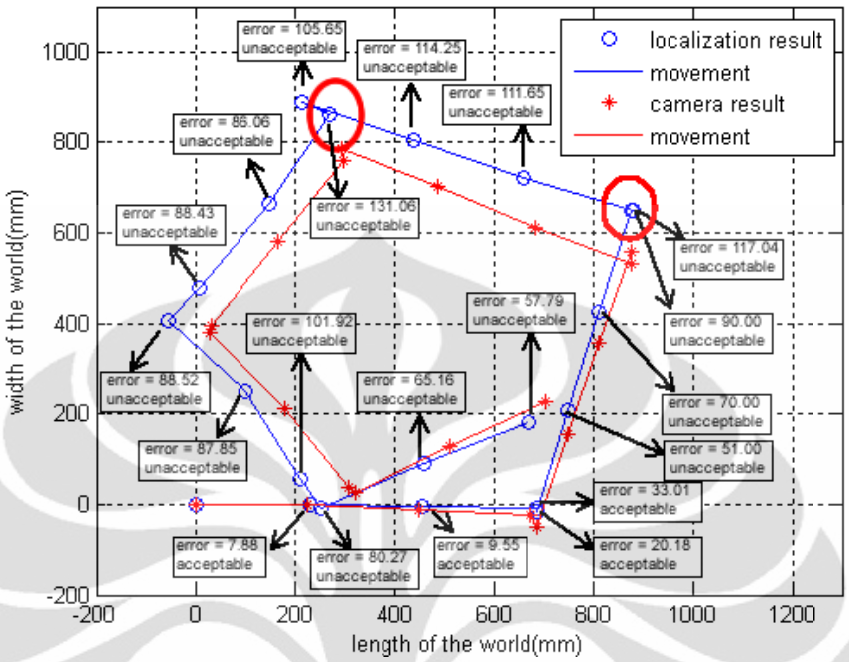
6.2.1 Strategi *Landmark detection*

Pengujian pertama dilakukan dengan parameter awal yang telah ditetapkan pada subbab 4.4, seperti disajikan pada (4.54) dan (6.1)

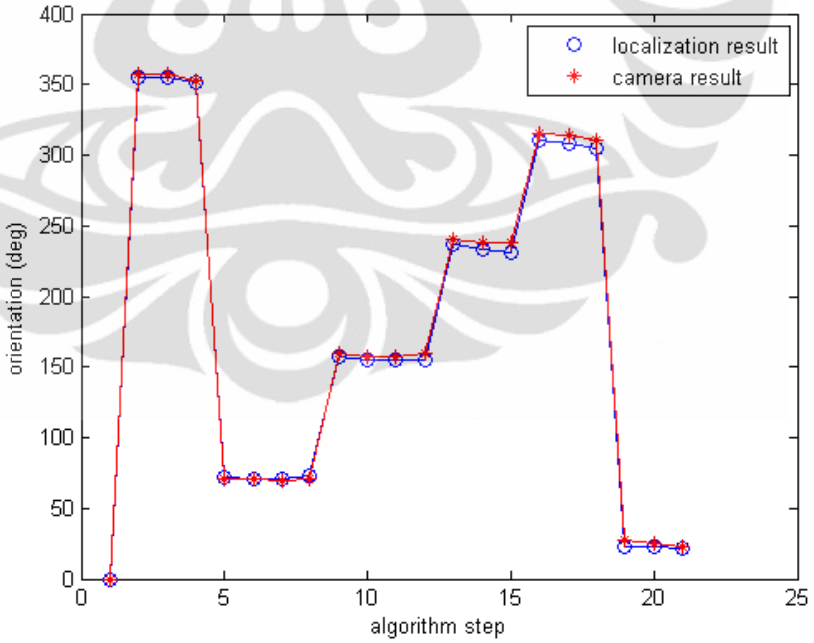
$$\boldsymbol{Q} = \begin{bmatrix} 18.49 & 0 & 0 \\ 0 & 18.12 & 0 \\ 0 & 0 & 3.43 \end{bmatrix} \tag{4.54}$$

$$\boldsymbol{R} = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 0.000002 \end{bmatrix} \tag{6.1}$$

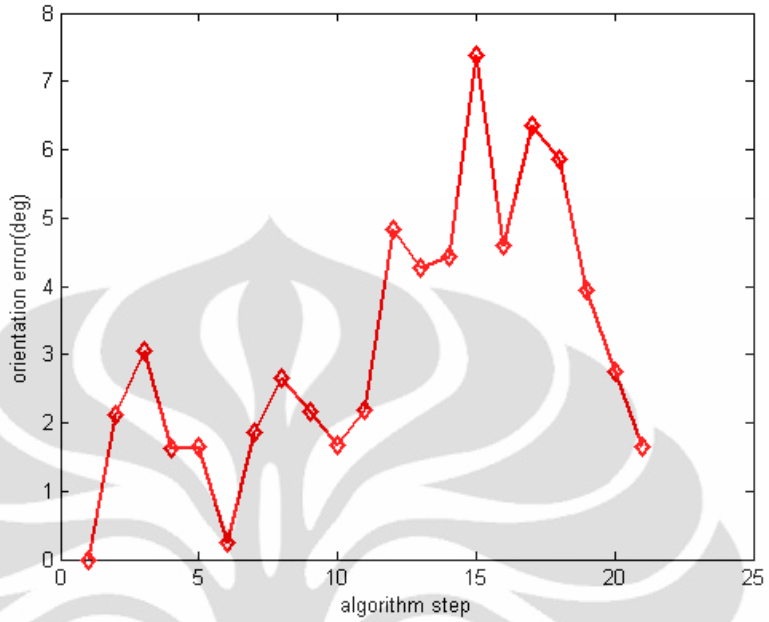
Parameter $\boldsymbol{R}$ yang digunakan pada pengujian pertama ini berukuran 3x3 karena pada pengujian awal hanya digunakan data dari dua sensor PING. Hasil yang didapat dari pengujian algoritma lokalisasi menggunakan parameter ini disajikan dalam gambar 6.2-6.4



Gambar 6.2 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera. Lingkaran merah menunjukkan posisi dengan error tertinggi



Gambar 6.3 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.4 Plot error orientasi

Hasil yang didapat memperlihatkan bahwa pola pergerakan yang diperoleh dari hasil lokalisasi memiliki kemiripan dengan pola pergerakan yang didapat dari kamera. Meskipun demikian, masih terdapat error pada *pose* yang didapat dari hasil lokalisasi. Pada beberapa titik, error ini bernilai kecil. Akan tetapi pada titik lain, error ini bernilai cukup besar. Beberapa contoh titik yang mempunyai error cukup besar adalah titik pada *step* ke 9 dan *step* ke 18 yang masing-masing menghasilkan error sebesar 11.7 cm untuk posisi y dan 9.9 cm untuk posisi x. Lebih jelasnya, error yang terjadi pada pengujian ini adalah sebagai berikut :

1. $\overline{err}_{pos} = 7.22cm$
2. $\overline{err}_{\theta} = 3.11°$
3. *Acceptable* : 23.8%

Apabila dicocokkan dengan konfigurasi posisi robot di lingkungan uji yang ditunjukkan oleh gambar 6.2, didapat dua kemungkinan penyebab error tersebut

1. Kegagalan penentuan koordinat *landmark* karena terjadi error pada pembacaan jarak.

2. Kesalahan menentukan posisi (x,y) setelah robot melakukan gerakan berputar.

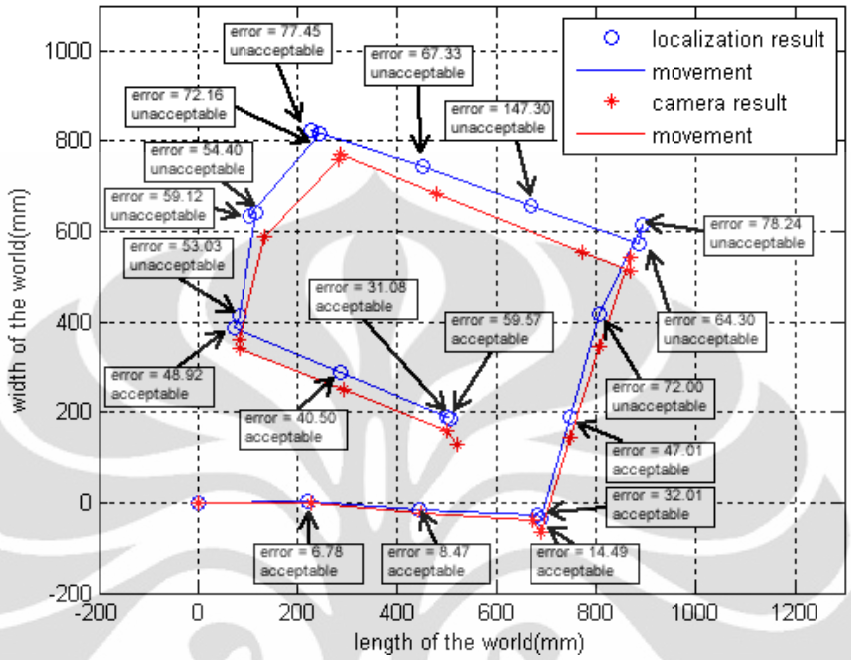
Terdapat fakta menarik tentang hasil lokalisasi yang didapat. Meskipun terdapat error di beberapa titik, tetapi begitu robot menemukan titik *landmark* yang benar, algoritma lokalisasi yang diterapkan mampu mengoreksi *pose* robot dan memperkecil error yang terjadi pada poin-poin sebelumnya.

Di sisi lain, estimasi nilai orientasi robot memberikan hasil yang cukup bagus. Hal ini dikarenakan pengukuran yang digunakan untuk melakukan koreksi nilai orientasi berasal dari pengukuran nilai absolut orientasi. Dengan demikian, koreksi yang dilakukan tidak perlu melewati fungsi *measurement prediction*, sehingga hasil yang didapat lebih baik. Di samping itu, sensor CMPS03 yang digunakan juga memiliki nilai ketidakpastian yang kecil.

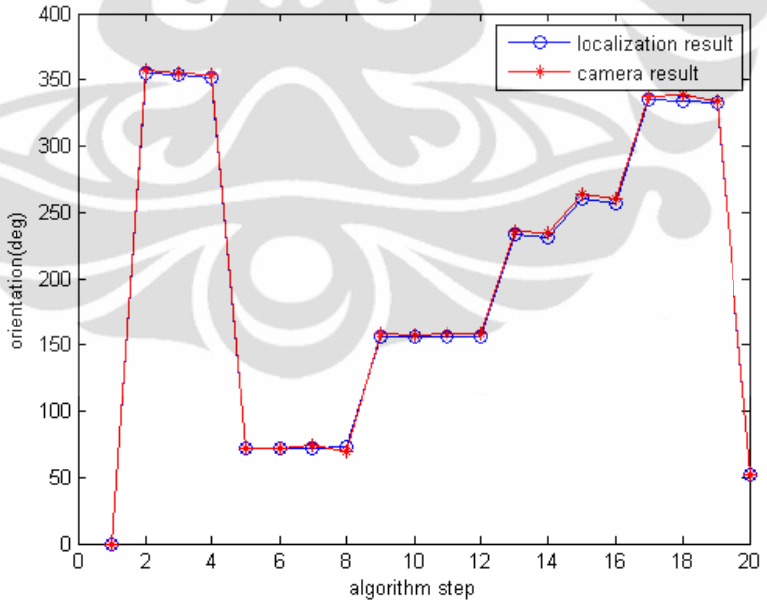
Berdasarkan hasil yang didapat pada pengujian pertama, dilakukanlah pengujian kedua dengan mengubah parameter matriks $\boldsymbol{R}$ yang digunakan. Pada pengujian ini digunakan matriks $\boldsymbol{R}$ yang memiliki bentuk seperti terlihat pada 6.2

$$\boldsymbol{R} = \begin{bmatrix} 500 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 0.000002 \end{bmatrix} \quad (6.2)$$

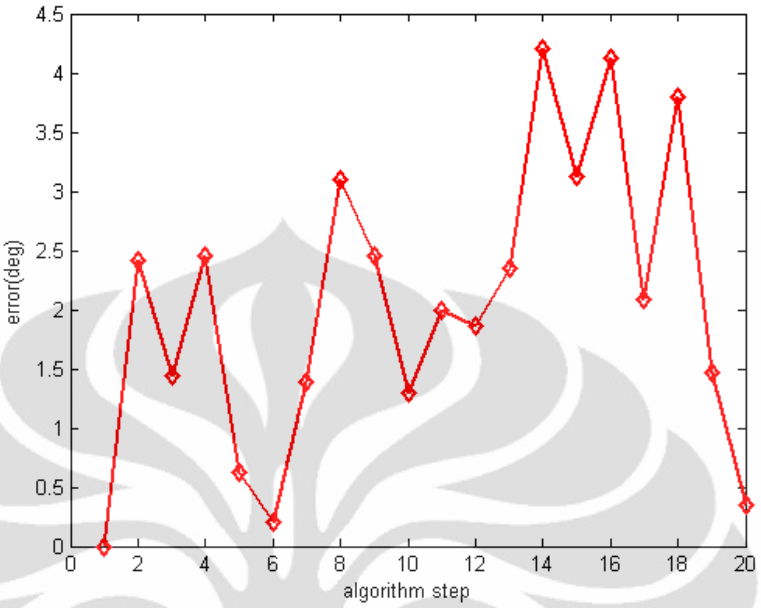
Nilai elemen matriks $\boldsymbol{R}$ yang bersesuaian dengan data sensor sonar PING diperbesar karena sumber error dianggap berasal dari ketidakpastian data jarak yang didapat dari sensor sonar. Hasil yang didapat dari pengujian ini dapat dilihat pada gambar 6.5-6.7



Gambar 6.5 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



Gambar 6.6 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.7 Plot error orientasi

Hasil yang didapat menunjukkan bahwa error yang terjadi pada pengujian kedua ini masih cukup besar. Lebih jelasnya, nilai error yang terjadi pada pengujian ini adalah sebagai berikut:

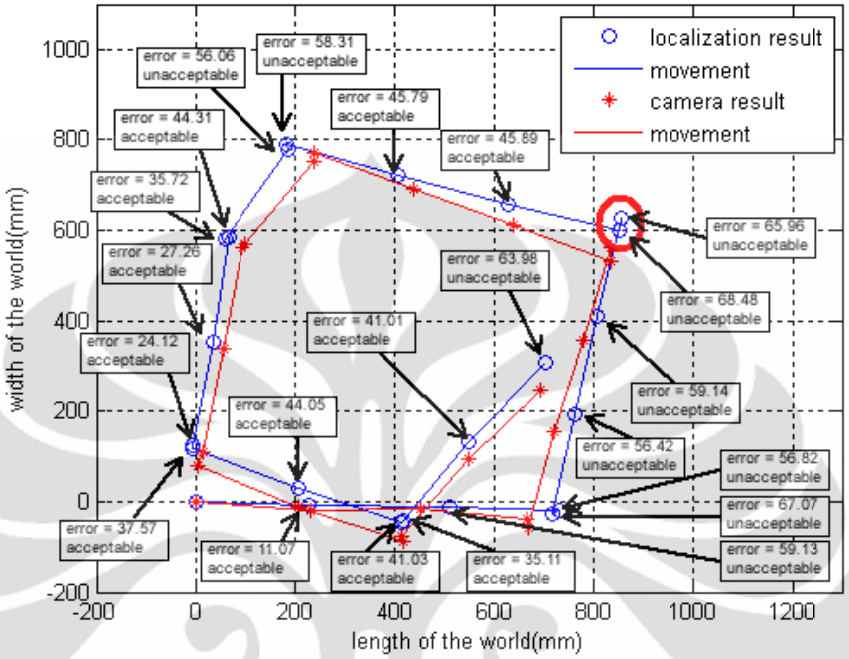
1. $\overline{err}_{pos} = 5.17cm$
2. $\overline{err}_{\theta} = 2.04°$
3. *Acceptable* : 40%

Terlihat bahwa pengubahan parameter matriks ***R*** mampu memberikan perbaikan error yang cukup signifikan pada nilai posisi dan nilai orientasi. Meskipun begitu, error yang didapat masih tidak sesuai kriteria.

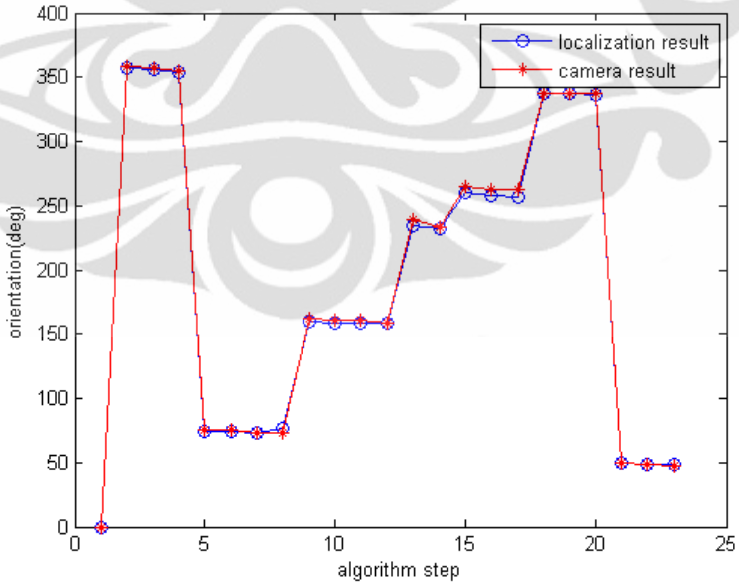
Pengujian berikutnya dilakukan dengan menambah jumlah sensor sonar yang digunakan, yaitu sonar, dari 2 menjadi 4. Penambahan jumlah sensor ini dimaksudkan untuk memperbanyak kemungkinan data yang dapat digunakan untuk melakukan koreksi. Konsekuensinya adalah terjadi perubahan matriks parameter ***R*** seperti terlihat pada (6.3)

$$\boldsymbol{R} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0.000002 \end{bmatrix} \quad (6.3)$$

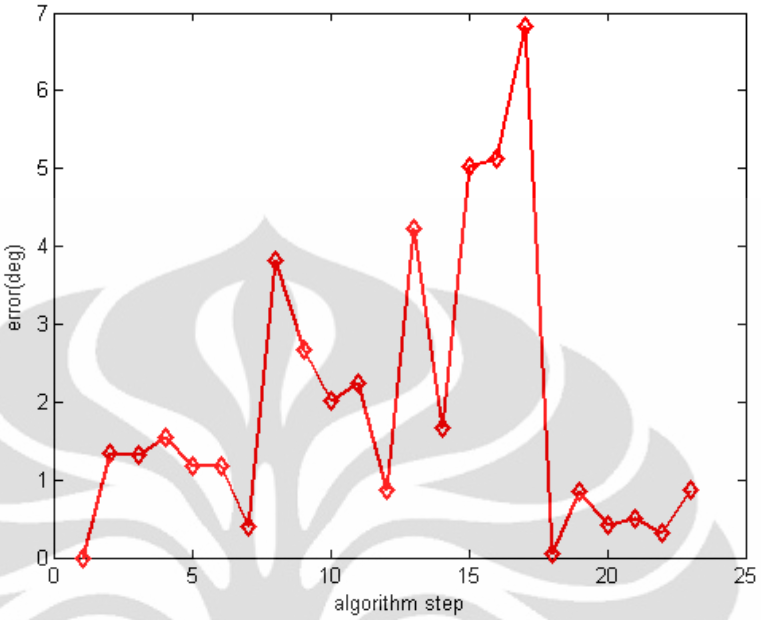
Hasil yang didapat dari pengujian ini disajikan dalam gambar 6.8-6.10



Gambar 6.8 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera. Lingkaran merah menunjukkan posisi dengan error tertinggi



Gambar 6.9 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.10 Plot error orientasi

Hasil yang didapat pada pengujian ketiga ini menunjukkan adanya perbaikan error, dimana pada pengujian ini nilai error maksimal turun menjadi di bawah 7 cm. Lebih detil lagi, error yang didapat pada pengujian ini adalah sebagai berikut :

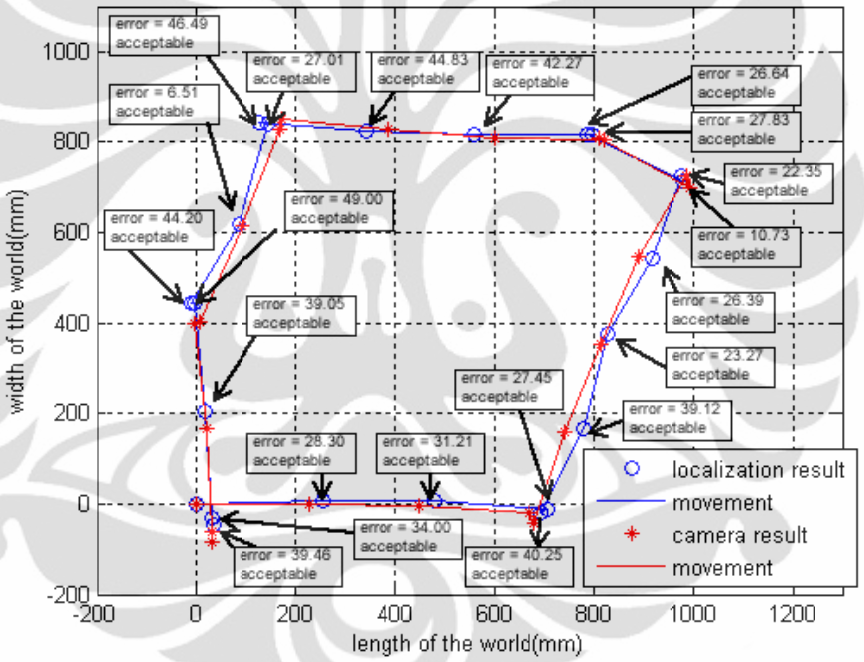
1. $\overline{err}_{pos} = 4.54cm$
2. $\overline{err}_{\theta} = 1.93°$
3. *Acceptable* : 56.52%

Terlihat bahwa terjadi perbaikan pada error posisi, akan tetapi, persentase *acceptable* masih belum sesuai dengan kriteria. Pada pengujian ini, error terbesar terjadi pada posisi yang hampir sama dengan pengujian 1. Titik yang dimaksudkan dapat dilihat pada gambar 6.8. Pada titik tersebut, error dapat terjadi karena dua hal, error pada pembacaan sensor jarak, atau error karena gerakan berputar yang dilakukan robot. Kemungkinan pertama telah diantisipasi dengan memperbesar nilai matriks ***R*** yang bersesuaian dengan data sonar. Karena error masih terjadi di posisi yang kurang lebih sama, maka berikutnya akan dilakukan pengujian dengan mengubah parameter matriks ***Q***.

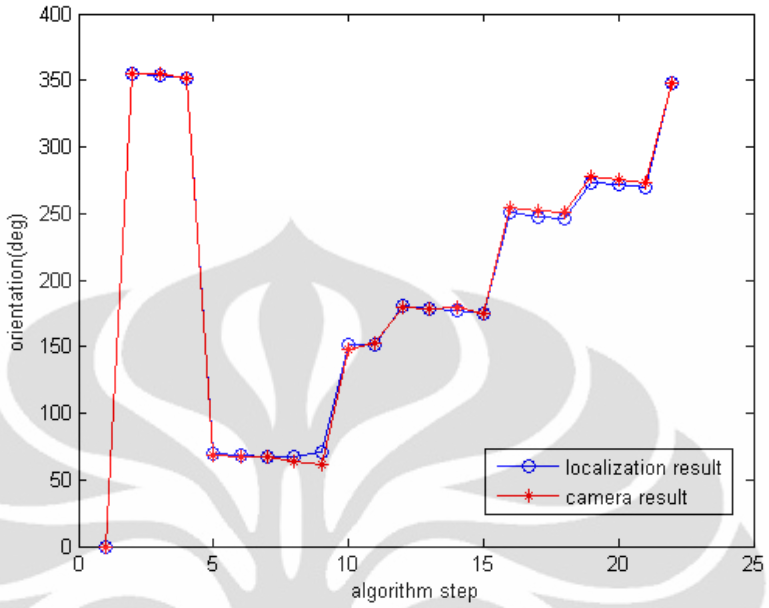
Perubahan matriks $\boldsymbol{Q}$ yang akan digunakan pada pengujian keempat disajikan pada 6.4

$$\boldsymbol{Q} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{6.4}$$

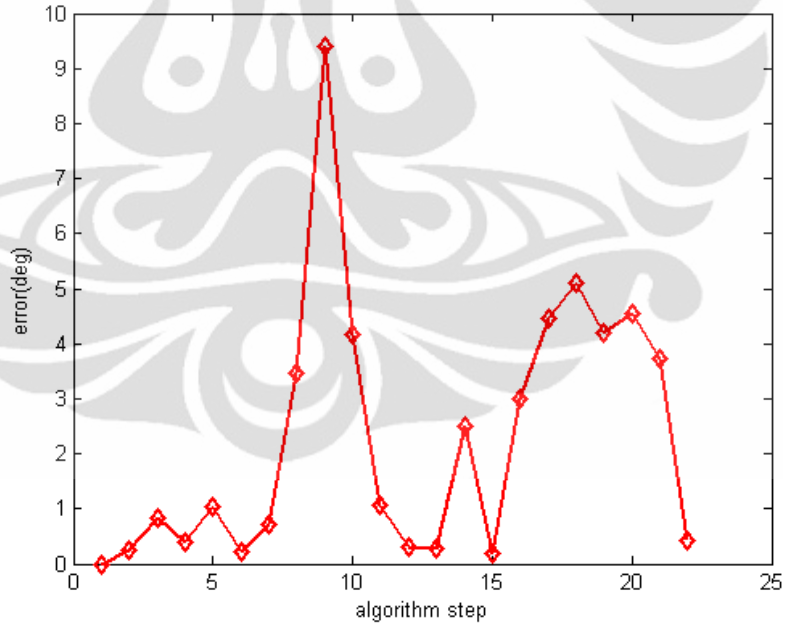
Perubahan ini dilakukan untuk mengatasi kesalahan estimasi posisi karena gerak berputar robot. Hasil yang diperoleh dari pengujian ini disajikan dalam 6.11-6.13



Gambar 6.11 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



Gambar 6.12 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.13 Plot error orientasi

Hasil yang didapat menunjukkan perbaikan error yang signifikan. Error maksimum sistem dengan parameter ini mengalami penurunan menjadi di bawah 5 cm. Error untuk masing-masing komponen *pose* juga mengalami penurunan, yaitu sebagai berikut:

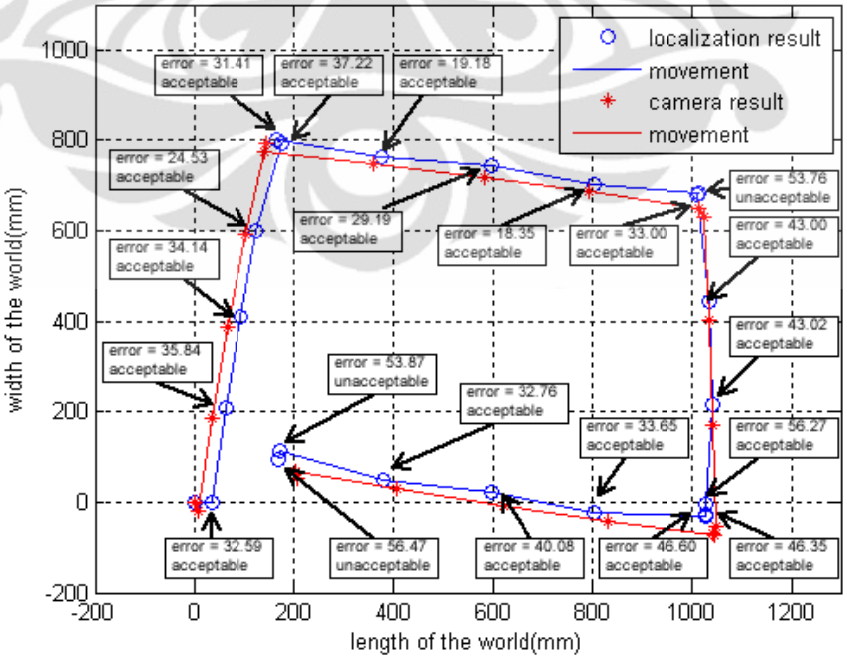
1. $\overline{err}_{pos} = 3.07cm$
2. $\overline{err}_{\theta} = 2.23°$
3. *Acceptable* : 100%

Error yang didapat pada pengujian ini memenuhi kriteria yang ditetapkan, sehingga dapat dikatakan bahwa parameter yang digunakan ini adalah parameter optimal. Dengan demikian parameter optimal untuk strategi *landmark detection* adalah seperti disajikan pada (6.4) dan (6.5)

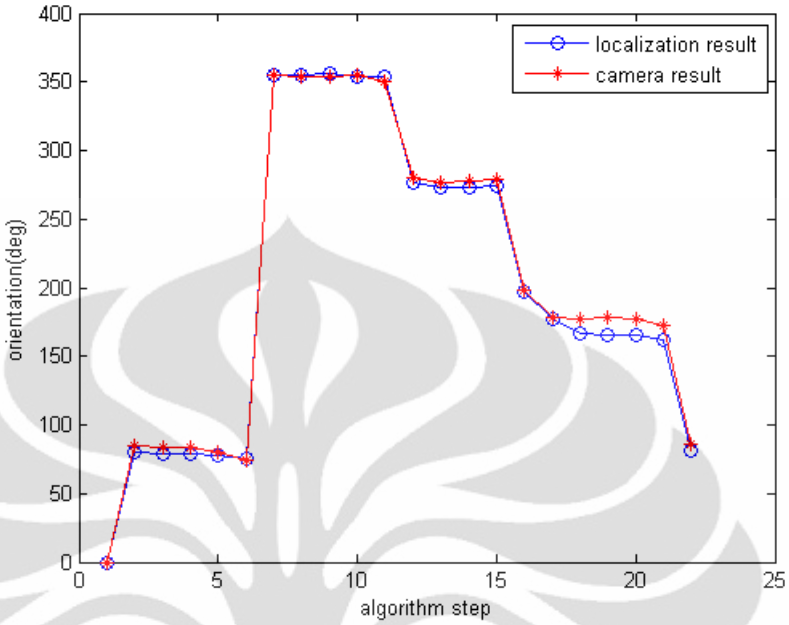
$$\boldsymbol{Q} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{6.4}$$

$$\boldsymbol{R} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0.000002 \end{bmatrix} \tag{6.5}$$

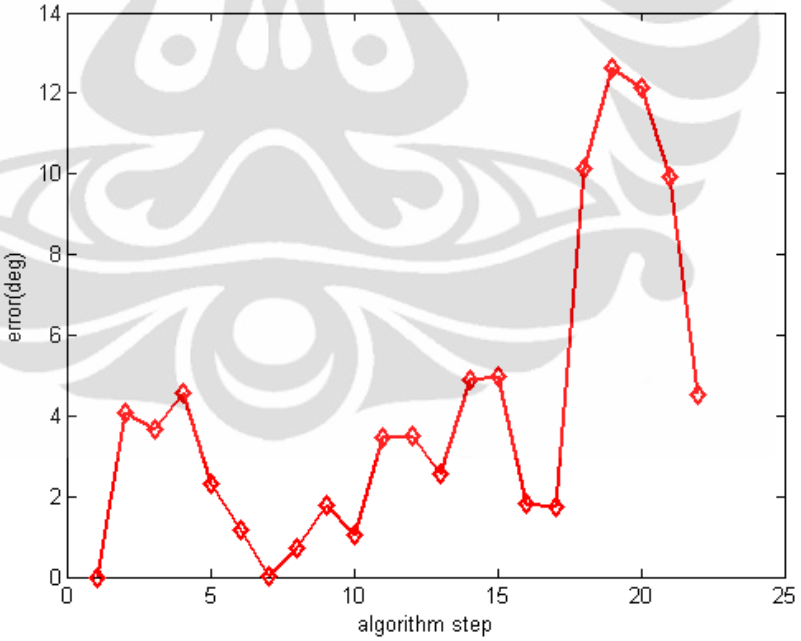
Berikutnya, parameter yang telah optimal ini diuji menggunakan lintasan yang berbeda. Hasil yang diperoleh untuk pengujian dengan lintasan searah jarum jam disajikan pada gambar 6.14-6.16.



Gambar 6.14 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



Gambar 6.15 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.16 Plot error orientasi

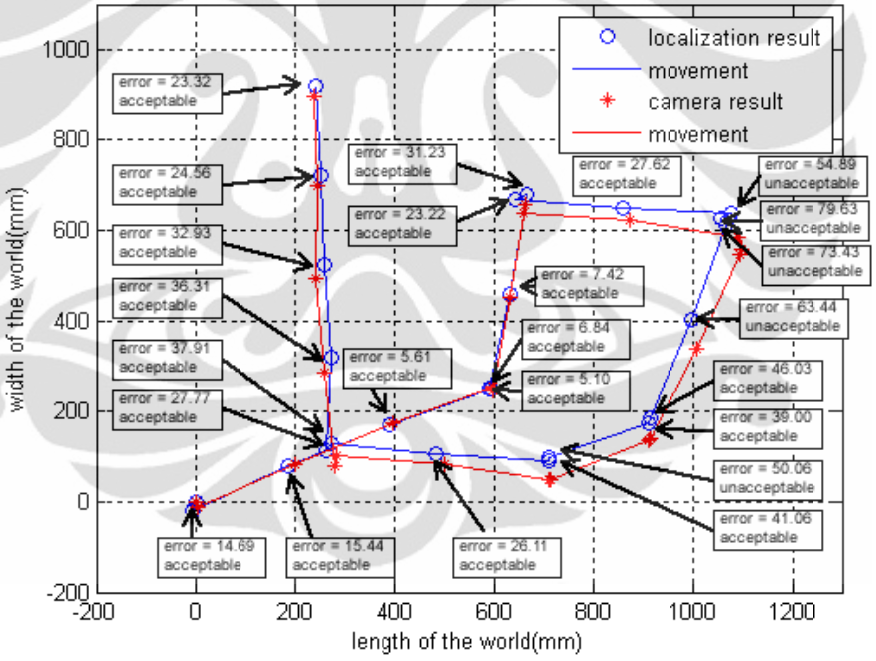
Hasil pengujian dengan lintasan searah jarum jam menunjukkan bahwa error yang didapat masih sesuai dengan kriteria optimal yaitu sebagai berikut :

1. $\overline{err}_{pos} = 3.64cm$
2. $\overline{err}_{\theta} = 4.17°$
3. *Acceptable* : 81.82%

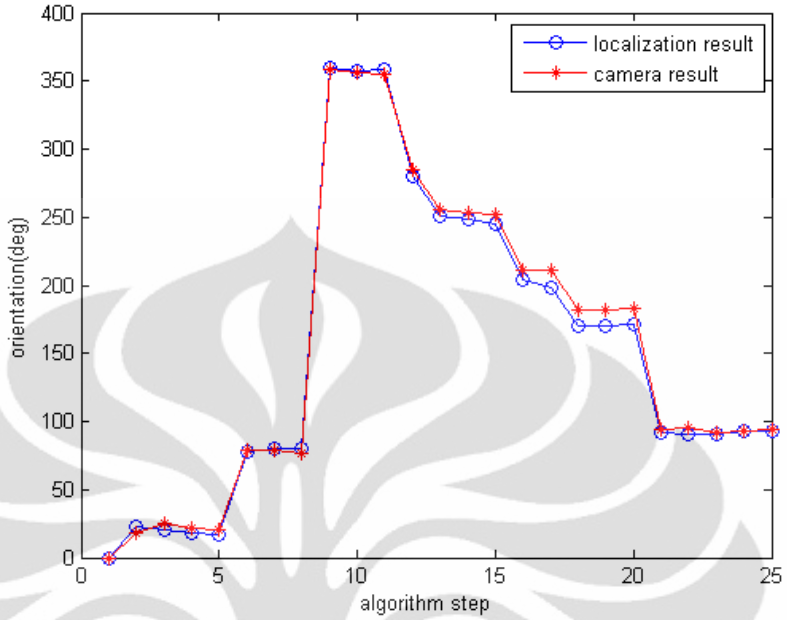
Berikutnya dilakukan pengujian dengan lintasan bebas yag mengandung arah berlawanan dan searah jarum jam. Hasil yang diperoleh adalah sebagai berikut :

1. $\overline{err}_{pos} = 3.17cm$
2. $\overline{err}_{\theta} = 4.16°$
3. *Acceptable* : 80%

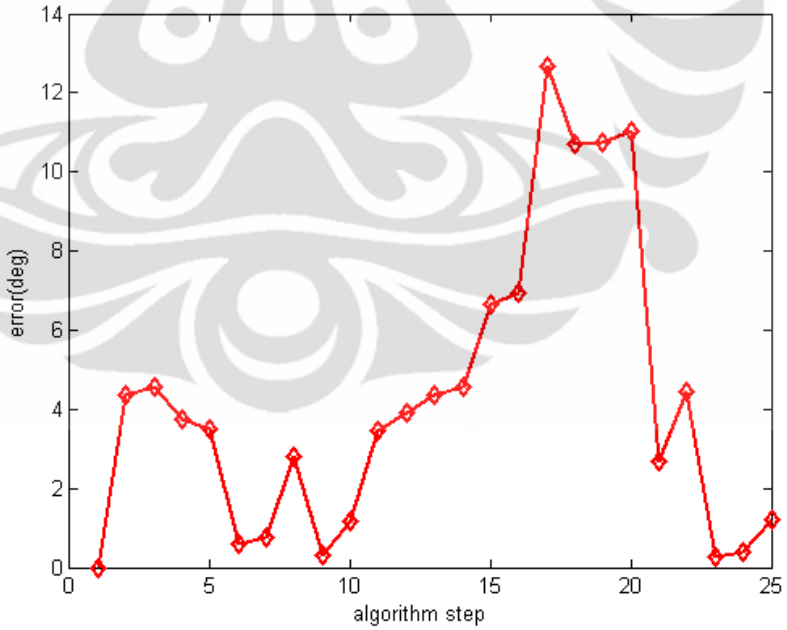
Detail dari hasil yang diperoleh disajikan pada gambar 6.17-6.19



Gambar 6.17 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



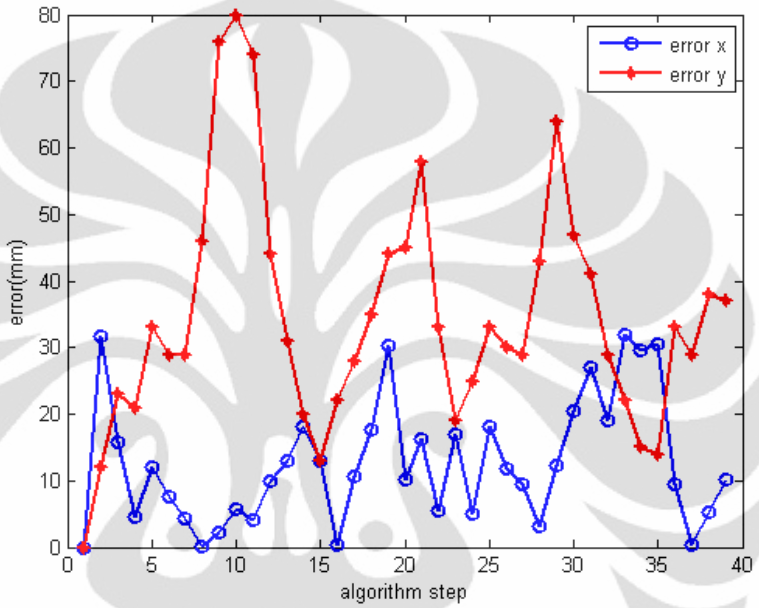
Gambar 6.18 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



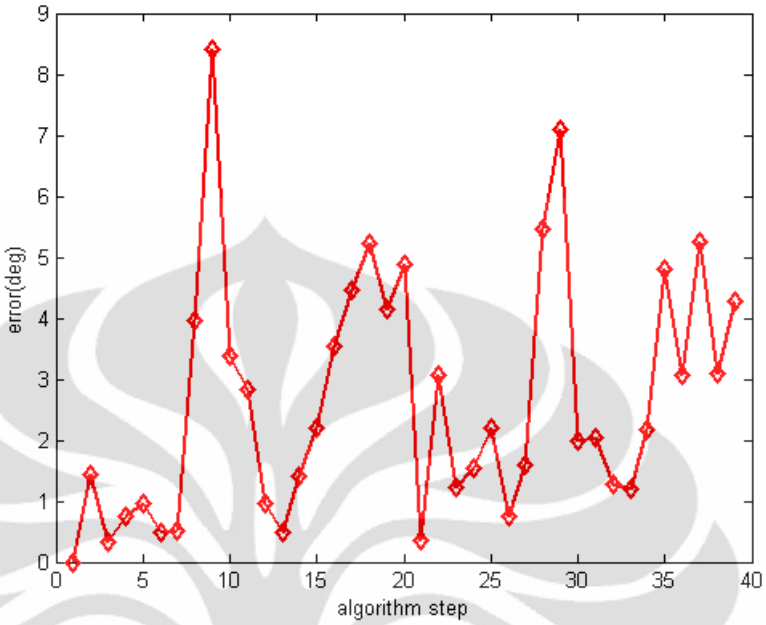
Gambar 6.19 Plot error orientasi

Hasil yang didapat menunjukkan bahwa untuk semua komponen, kriteria optimal dapat tercapai.

Terakhir dilakukan pengujian untuk menentukan konsistensi algoritma lokalisasi yang digunakan. Pengujian dilakukan dengan menjalankan robot pada lintasan sirkular dua kali putaran. Setelah itu, dilakukan analisis terhadap error dari *pose* yang didapat. Hasil pengujian disajikan pada gambar 6.20 dan 6.21



Gambar 6.20 Plot error posisi x dan y



Gambar 6.21 Plot error orientasi

Hasil pengujian menunjukkan bahwa untuk komponen x dan y, tidak terdapat kecenderungan bahwa error yang didapat akan bertambah menuju tak hingga. Dengan demikian, estimasi komponen x dan y dapat dikatakan konsisten untuk jumlah *step* tertentu.

Untuk komponen orientasi, error yang didapat untuk *step* selama pengujian tidak menunjukkan tanda-tanda bahwa nilainya akan bertambah menuju tak hingga. Meskipun demikian, pola yang dibentuk cenderung naik. Oleh karena itu, dapat dikatakan bahwa meskipun estimasi komponen orientasi konsisten untuk jumlah *step* tertentu, namun terdapat kemungkinan terjadi inkonsistensi pada estimasi nilai orientasi.

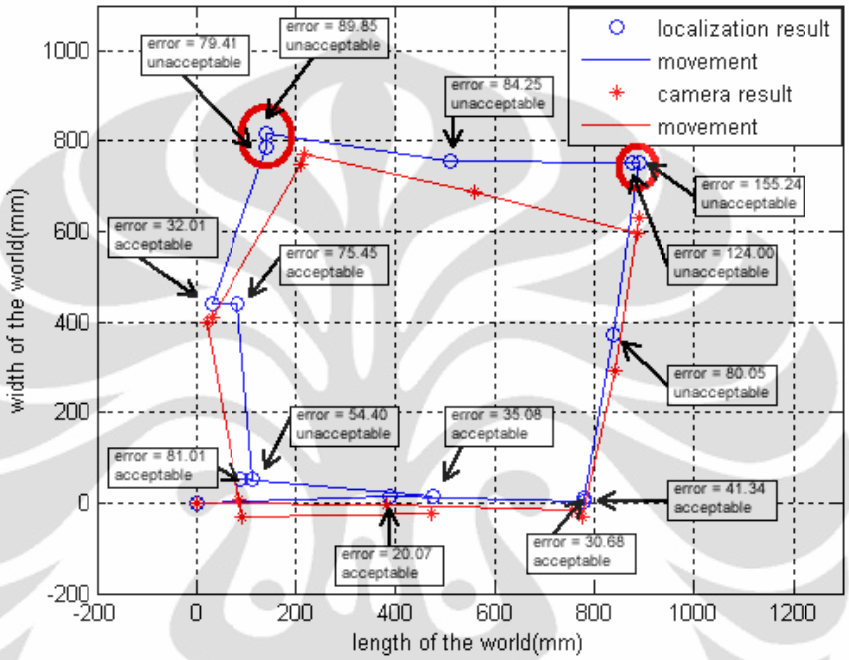
6.2.2 Strategi *Line extraction*

Pengujian pertama yang dilakukan untuk strategi *line extraction* adalah pengujian dengan parameter standar, seperti terlihat pada (4.54) dan (4.69)

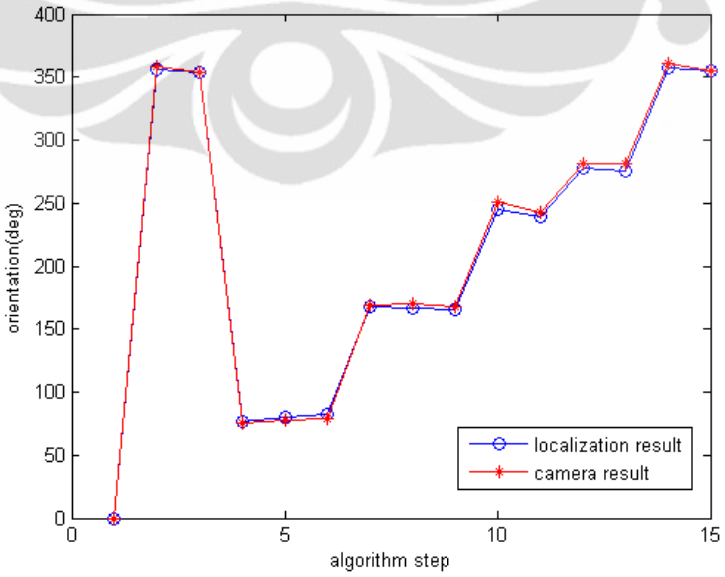
$$\boldsymbol{Q} = \begin{bmatrix} 18.49 & 0 & 0 \\ 0 & 18.12 & 0 \\ 0 & 0 & 3.43 \end{bmatrix} \quad (4.54)$$

$$\boldsymbol{R} = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 0.000002 \end{bmatrix} \tag{4.69}$$

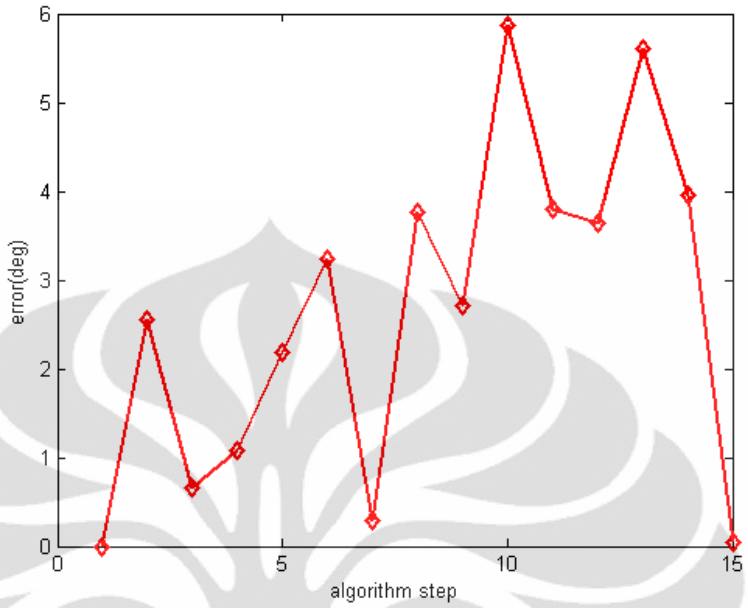
Hasil pengujian yang dilakukan disajikan dalam gambar 6.22-6.24



Gambar 6.22 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera. Lingkaran merah menunjukkan posisi dengan error tertinggi



Gambar 6.23 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.24 Plot error orientasi

Hasil pengujian menunjukkan bahwa terdapat error yang cukup besar pada beberapa titik. Titik yang memiliki error terbesar berada pada *step* ke 7 dan ke 9 yang masing-masing memberikan nilai error sebesar 15.5 cm pada posisi y dan 7.7 cm pada posisi x. Titik yang memberikan nilai error besar ini apabila diamati pada gambar 6.22 terletak pada posisi yang serupa dengan yang terjadi pada strategi *landmark detection*. Oleh karena itu, penyebabnya kemungkinan sama, yaitu :

1. Terjadi error pada pembacaan jarak. Hal ini mengakibatkan kesalahan penentuan parameter garis pada proses *line extraction* yang pada akhirnya menyebabkan kesalahan pada proses koreksi *pose*.
2. Kesalahan menentukan posisi (x,y) setelah robot melakukan gerakan berputar.

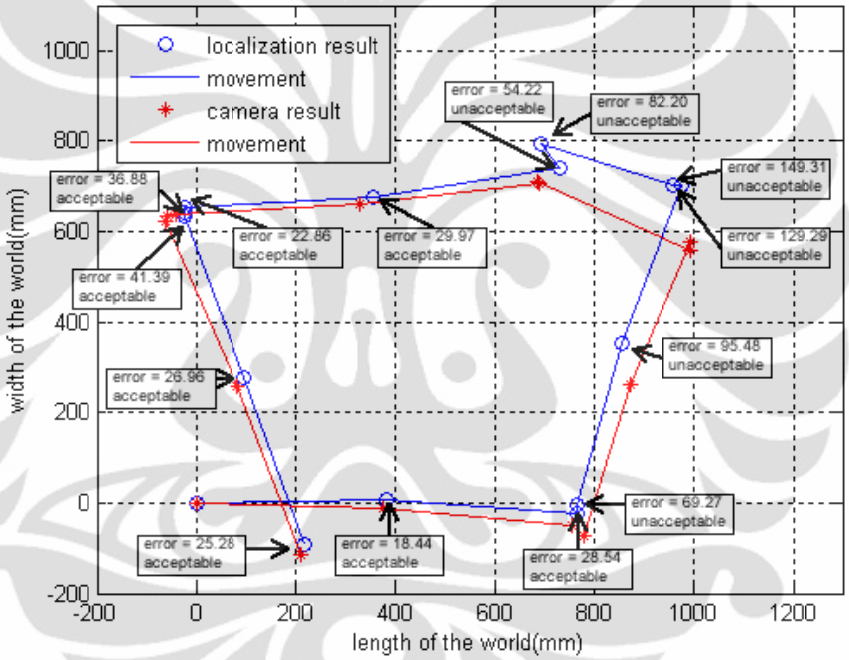
Oleh karena itu, pada pengujian kedua, digunakan parameter $\boldsymbol{Q}$ dan $\boldsymbol{R}$ yang telah diubah nilainya seperti terlihat pada (6.4) dan (6.6)

$$\boldsymbol{Q} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{6.4}$$

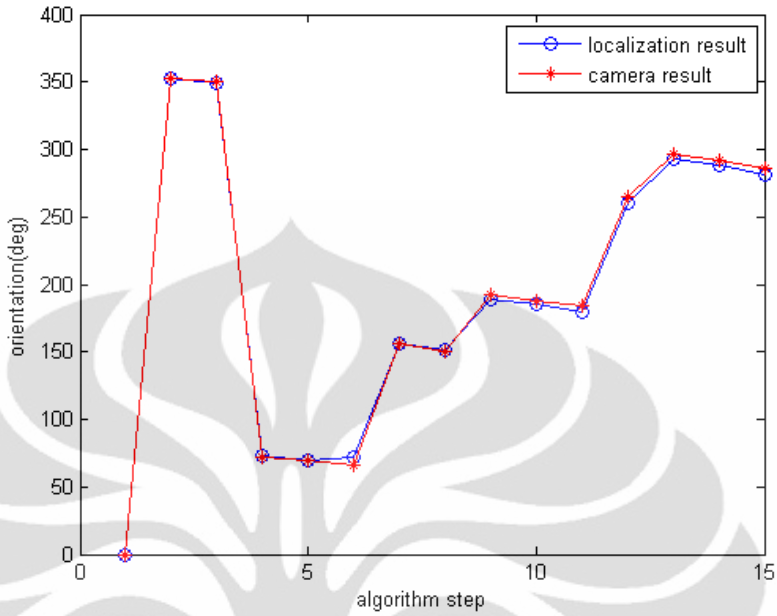
$$\boldsymbol{R} = \begin{bmatrix} 500 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 0.000002 \end{bmatrix} \tag{6.6}$$

Hasil yang didapat dari pengujian kedua dapat dilihat pada gambar 6.25-6.27, sedangkan error yang diperoleh adalah sebagai berikut:

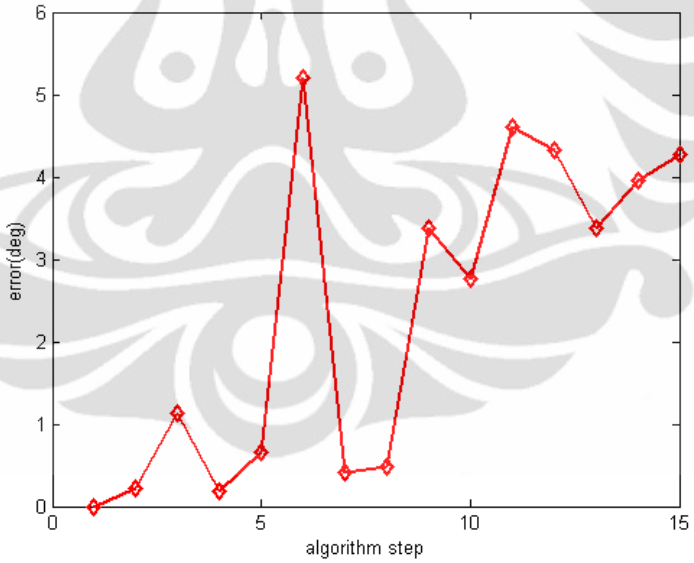
1. $\overline{err}_{pos} = 5.40cm$
2. $\overline{err}_{\theta} = 2.31°$
3. *Acceptable :* 60%



Gambar 6.25 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



Gambar 6.26 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.27 Plot error orientasi

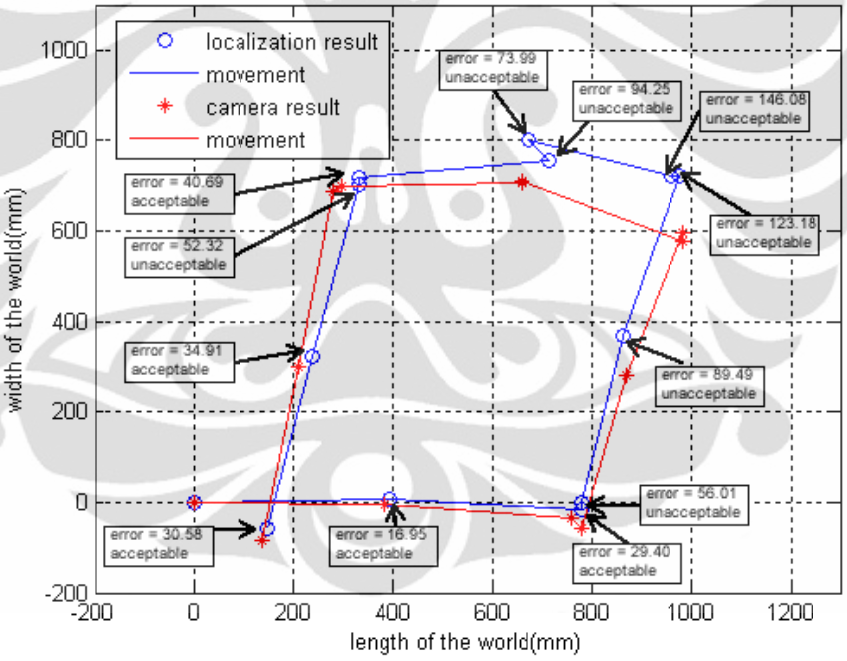
Setelah nilai parameter diubah ***Q*** dan ***R***, ketidakpastian data sensor jarak dan posisi setelah berputar belum dapat diantisipasi dengan sempurna. Hal ini dapat dilihat dari error yang dihasilkan, dimana error posisi dan persentase *acceptable* yang dihasilkan belum memenuhi kriteria. Oleh karena itu, untuk pengujian berikutnya, dilakukan

penambahan nilai parameter ***R*** sehingga parameter ***R*** akan memiliki bentuk seperti terlihat pada (6.7)

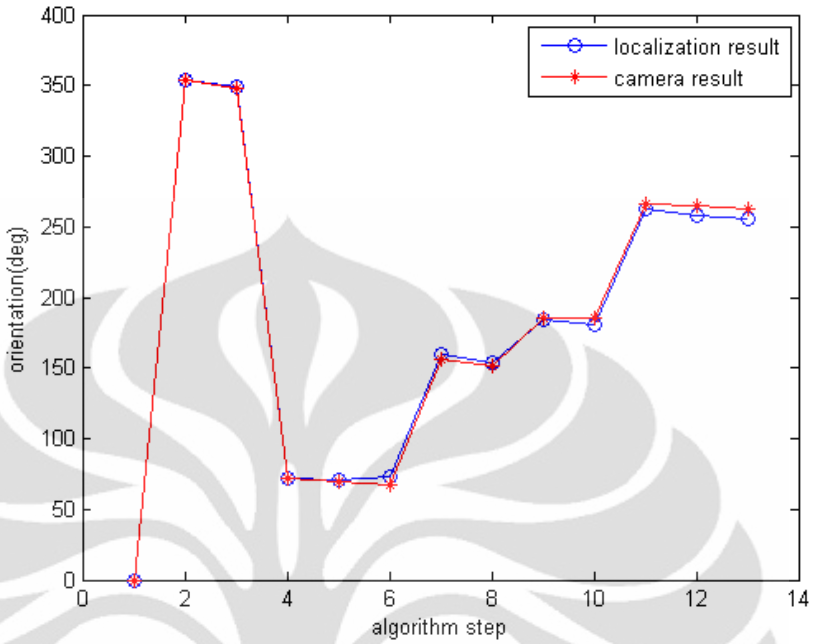
$$R = \begin{bmatrix} 700 & 0 & 0 \\ 0 & 700 & 0 \\ 0 & 0 & 0.000002 \end{bmatrix} \tag{6.7}$$

Pengujian dengan menggunakan parameter ***R*** yang digambarkan oleh (6.7) memberikan hasil yang dapat dilihat pada gambar 6.28-6.30 dan nilai error sebagai berikut :

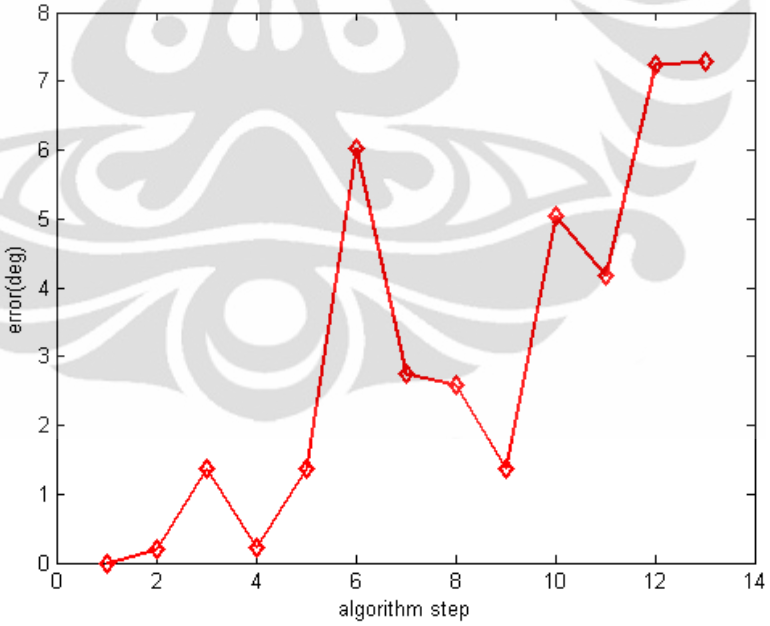
1. $\overline{err}_{pos} = 6.06cm$
2. $\overline{err}_{\theta} = 3.05°$
3. *Acceptable* : 46.15%



Gambar 6.28 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



Gambar 6.29 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.30 Plot error orientasi

Hasil yang didapat menunjukkan bahwa penambahan nilai elemen matriks ***Q*** dan ***R*** malah menyebabkan nilai error data bertambah. Hal ini dikarenakan perbaikan error dan nilai matriks ***Q*** dan ***R*** tidak mengikuti

hubungan linear. Kenaikan nilai matriks $\boldsymbol{Q}$ dan $\boldsymbol{R}$ dapat dipandang sebagai menurunnya derajat kepercayaan terhadap proses odometri dan hasil pengukuran sonar. Jika nilai $\boldsymbol{Q}$ dan $\boldsymbol{R}$ tidak sesuai dengan ketidakpastian yang dimiliki proses odometri dan pengukuran sonar, maka estimasi *pose* yang didapatkan akan memiliki error yang cukup besar.

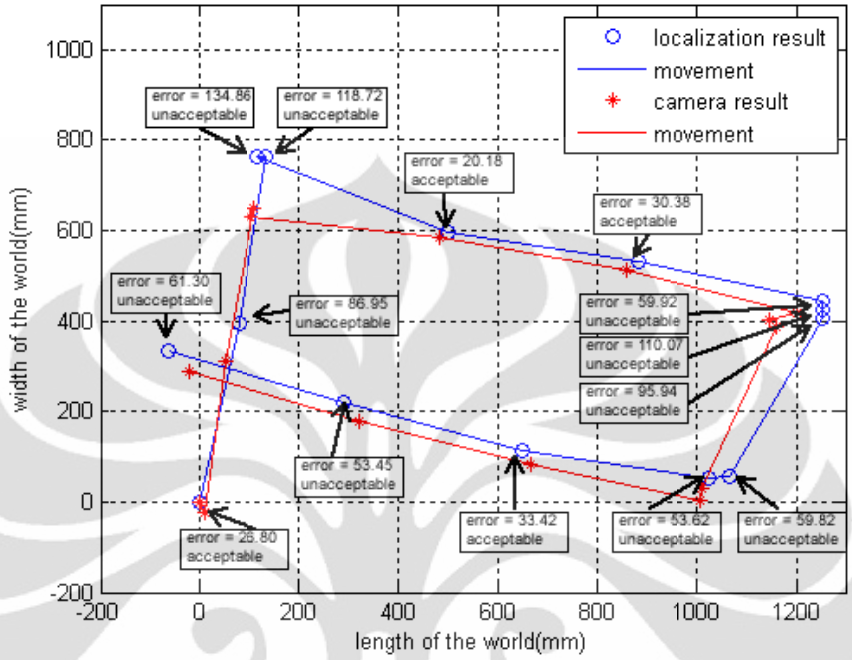
Dengan demikian dapat dikatakan bahwa parameter $\boldsymbol{Q}$ dan $\boldsymbol{R}$ yang ditulis pada (6.4) dan (6.6) adalah parameter optimal. Selain itu, dapat dikatakan pula bahwa error minimal yang dapat dicapai dengan menggunakan strategi *line extraction* tidak dapat memenuhi kriteria optimal sepenuhnya.

Kriteria optimal ini tidak tercapai salah satunya adalah karena error saat berputar yang tidak tertangani dengan sempurna. Penyebab dari terjadinya hal ini adalah vektor *measurement* yang tidak memadai. Setelah robot berputar, langkah *line extraction* belum dapat dijalankan. Dengan kata lain, garis yang baru belum tersedia. Pengoreksian posisi robot hanya dapat dilakukan menggunakan garis yang didapat dari *step* sebelumnya. Pada satu keadaan, dimana posisi robot tidak bergeser setelah berputar, koreksi posisi menggunakan garis yang didapat pada *step* sebelumnya masih memungkinkan karena jarak garis terhadap robot tidak berubah. Akan tetapi, perputaran robot yang terjadi tidak ideal. Ada kalanya posisi robot bergeser saat melakukan gerakan berputar. Pada kondisi ini garis yang didapat dari *step* sebelumnya tidak lagi memungkinkan untuk digunakan pada tahap koreksi.

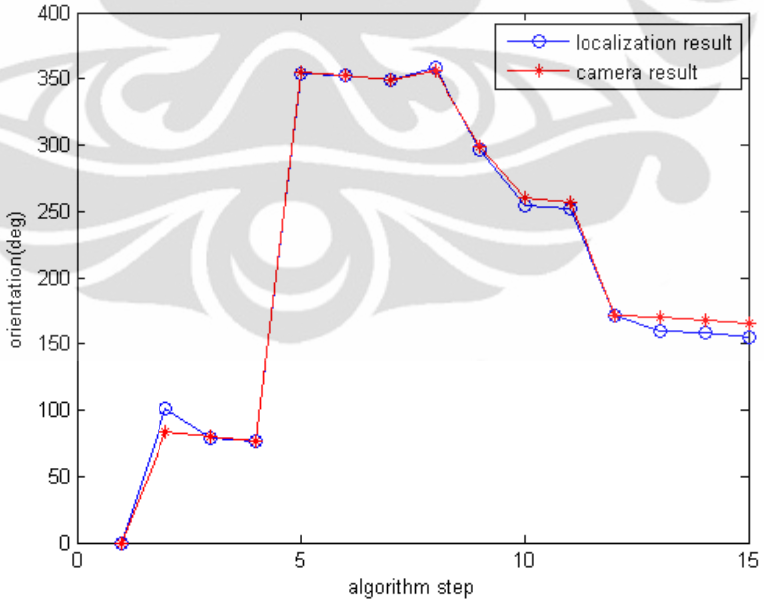
$$\boldsymbol{Q} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{6.4}$$

$$\boldsymbol{R} = \begin{bmatrix} 500 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 0.000002 \end{bmatrix} \tag{6.6}$$

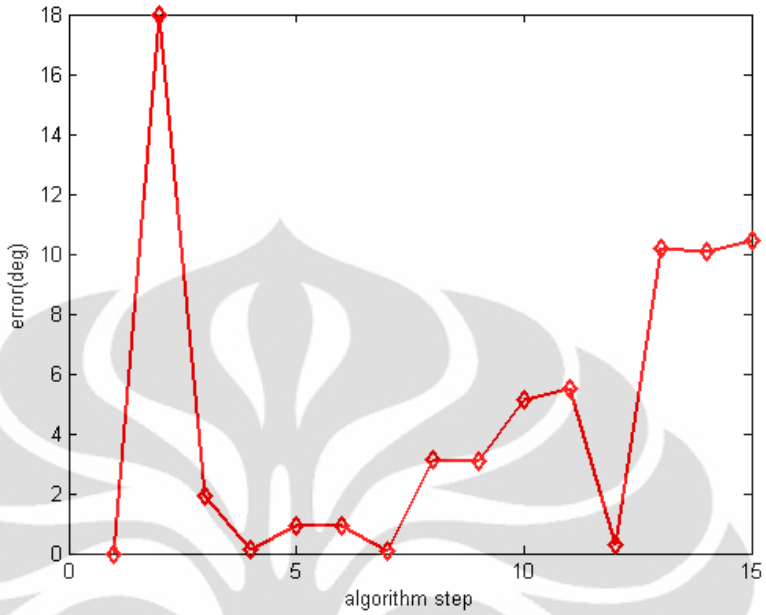
Berikutnya, dilakukan pengujian dengan lintasan lain untuk nilai $\boldsymbol{Q}$ dan $\boldsymbol{R}$**.** Hasil pengujian untuk lintasan searah jarum jam diperlihatkan oleh gambar 6.31-6.33



Gambar 6.31 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



Gambar 6.32 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.33 Plot error orientasi

Dari hasil yang didapat kemudian dapat diperoleh error sebagai berikut :

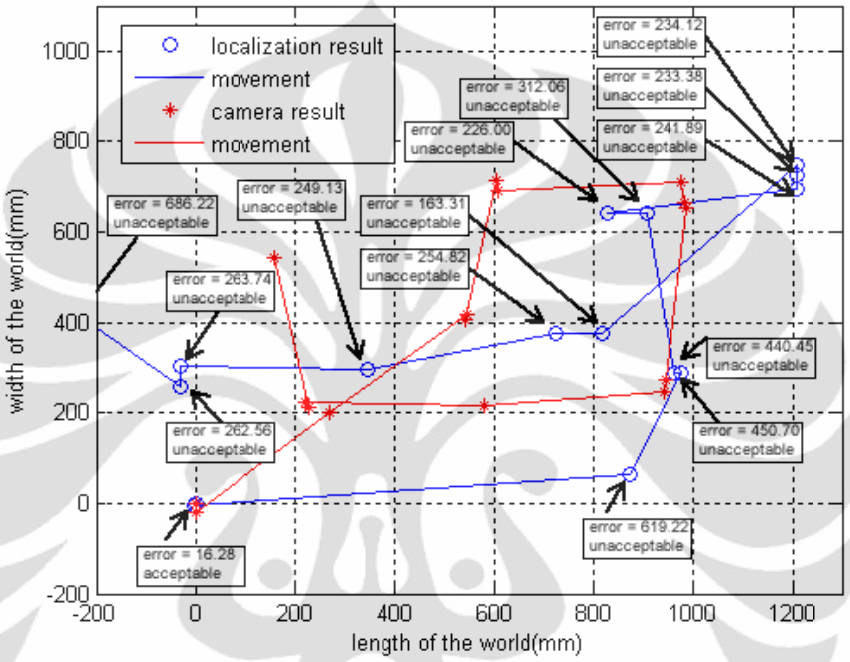
1. $\overline{err}_x = 6.30cm$
2. $\overline{err}_\theta = 4.65°$
3. *Acceptable* : 33.33%

Nilai error ini menunjukkan bahwa *pose* yang didapat untuk lintasan ini tidak memenuhi kriteria optimal yang diinginkan. Hal yang menarik terlihat pada error θ. Pada *step* ke-2 terlihat bahwa estimasi θ memiliki nilai error yang besar, mencapai 18º. Hal ini terjadi karena pada *step* awal, koreksi *pose* tidak dapat dilakukan karena belum ada garis yang tersedia.

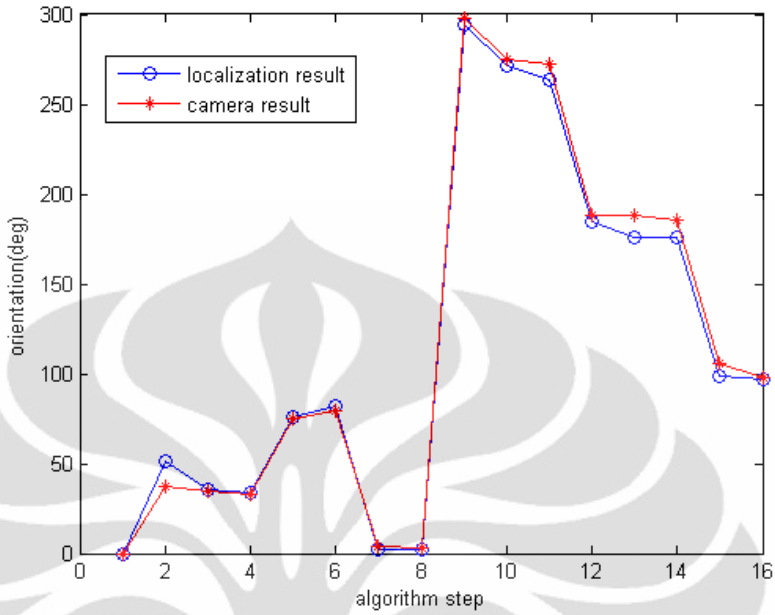
Pengujian berikutnya dilakukan untuk lintasan dengan arah bebas. Detil hasil pengujian dapat dilihat pada gambar 6.34-6.36 sedangkan nilai error yang diperoleh untuk lintasan ini adalah sebagai berikut:

1. $\overline{err}_x = 29.09cm$
2. $\overline{err}_\theta = 4.36°$
3. *Acceptable* : 12.50%

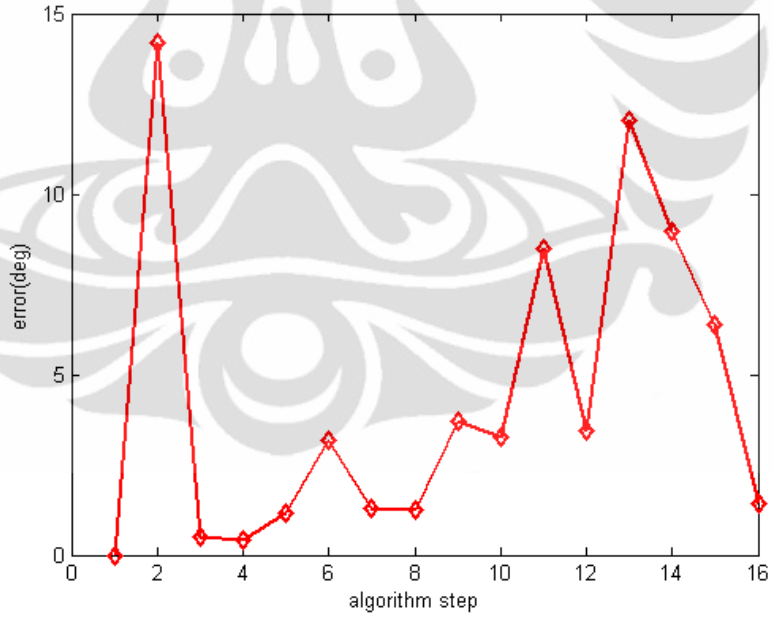
Dari hasil yang diperoleh dapat dikatakan bahwa strategi *line extraction* tidak mampu melakukan estimasi *pose* dengan sempurna pada lintasan bebas ini.



Gambar 6.34 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera

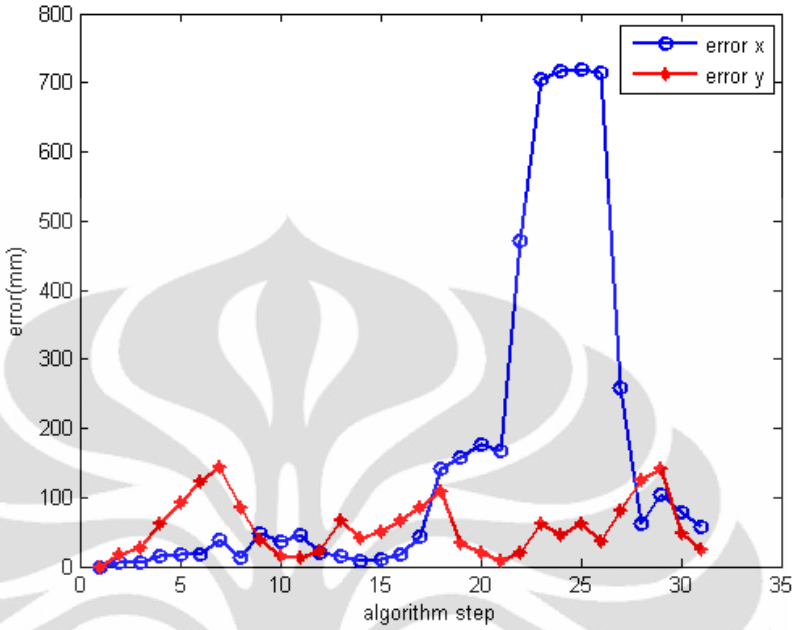


Gambar 6.35 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera

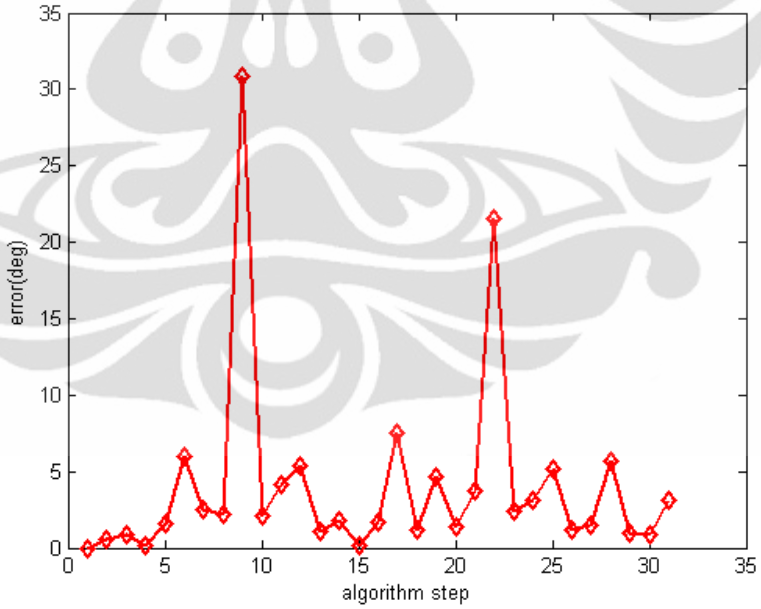


Gambar 6.36 Plot error orientasi

Terakhir dilakukan pengujian konsistensi strategi *line extraction* ini. Hasil yang diperoleh disajikan dalam gambar 6.37 dan 6.38



Gambar 6.37 Plot error posisi x dan y

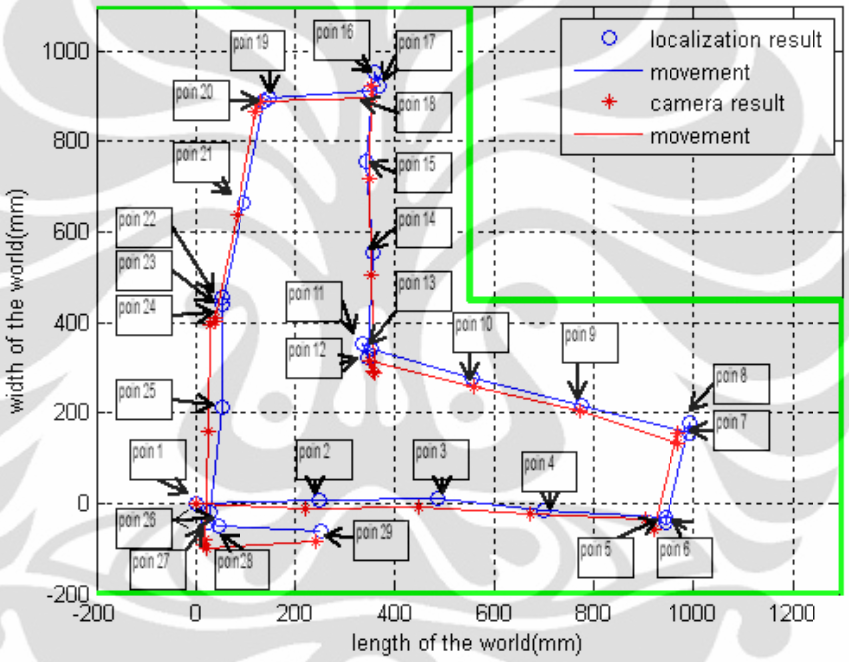


Gambar 6.38 Plot error orientasi

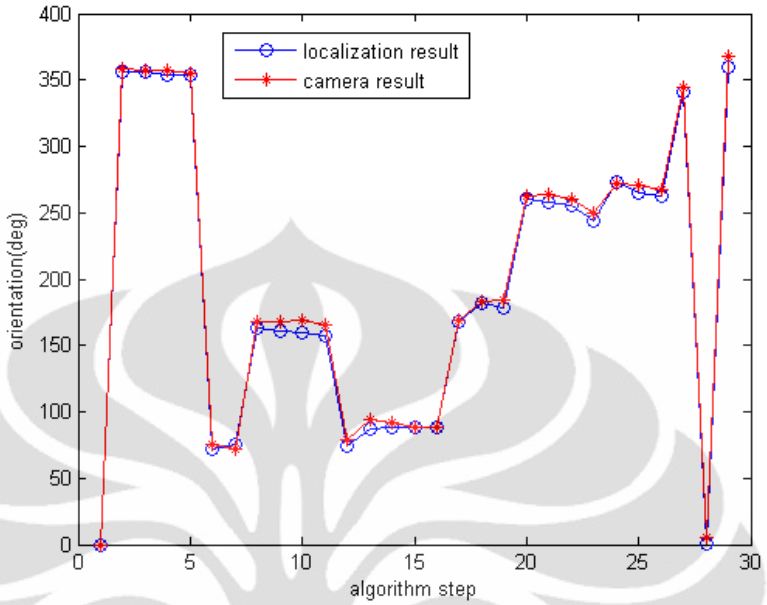
Berdasarkan data yang diperoleh, didapat bahwa error untuk ketiga komponen (x,y, θ) tidak memiliki kecenderungan untuk bertambah menuju tak hingga. Dengan demikian, untuk jumlah *step* tertentu, estimasi *pose* yang dihasilkan dapat dikatakan konsisten.

6.2.3 Pengujian Lingkungan Uji Tipe 2

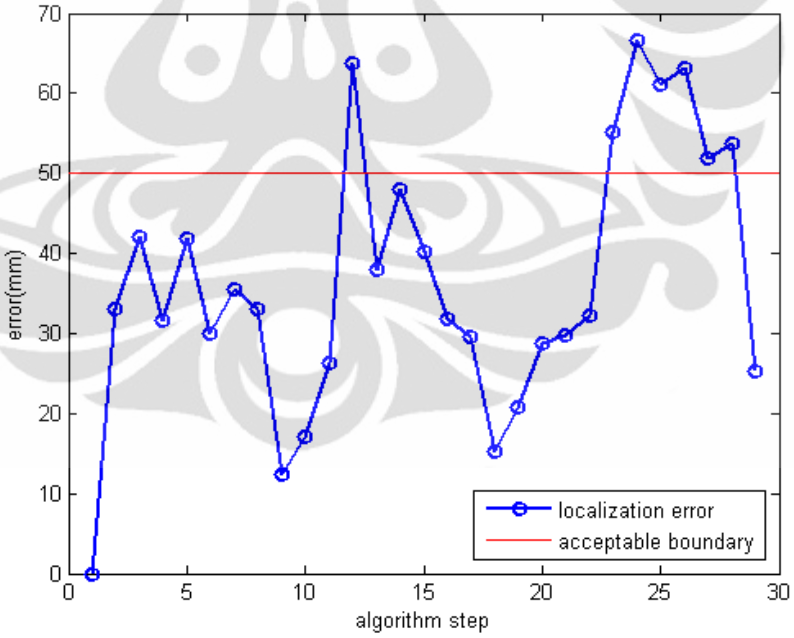
Pengujian dengan menggunakan lingkungan uji tipe 2 dilakukan pada strategi yang telah mencapai hasil optimal. Pada penelitian ini, strategi tersebut adalah strategi *landmark detection*. Pertama dilakukan pengujian dengan arah berlawanan arah jarum jam. Hasil yang diperoleh dapat dilihat pada gambar 6.39-6.42



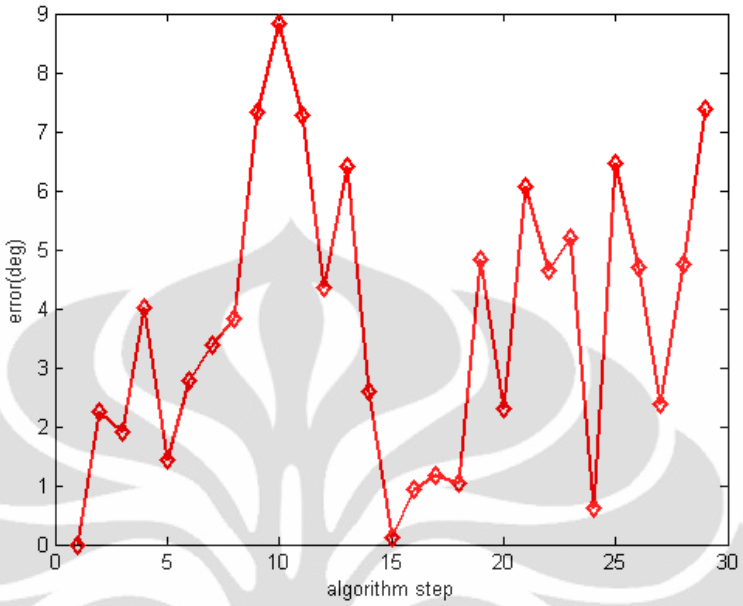
Gambar 6.39 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



Gambar 6.40 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.41 Plot error posisi x dan y



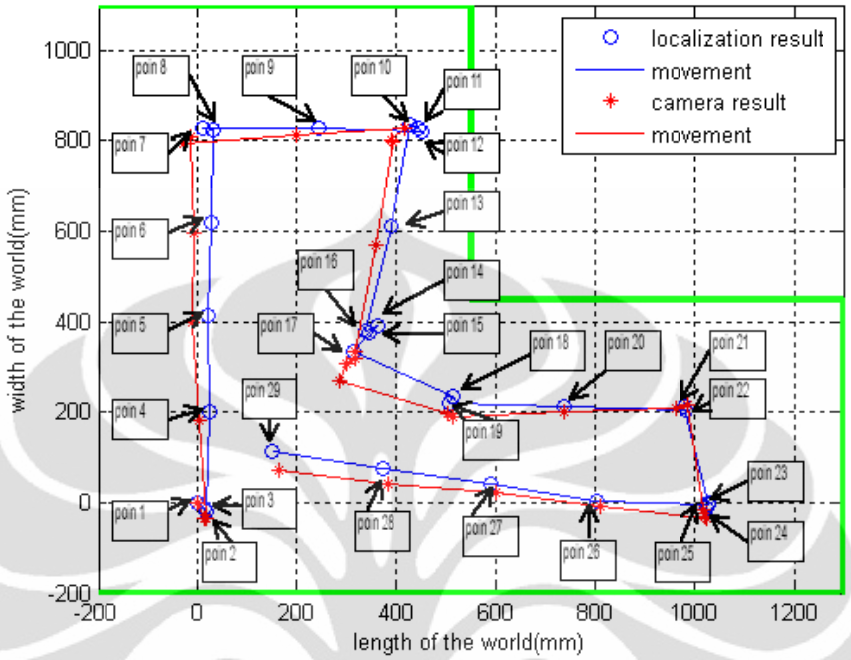
Gambar 6.42 Plot error orientasi

Nilai error yang diperoleh dari pengujian dengan lingkungan uji tipe 2 ini adalah sebagai berikut :

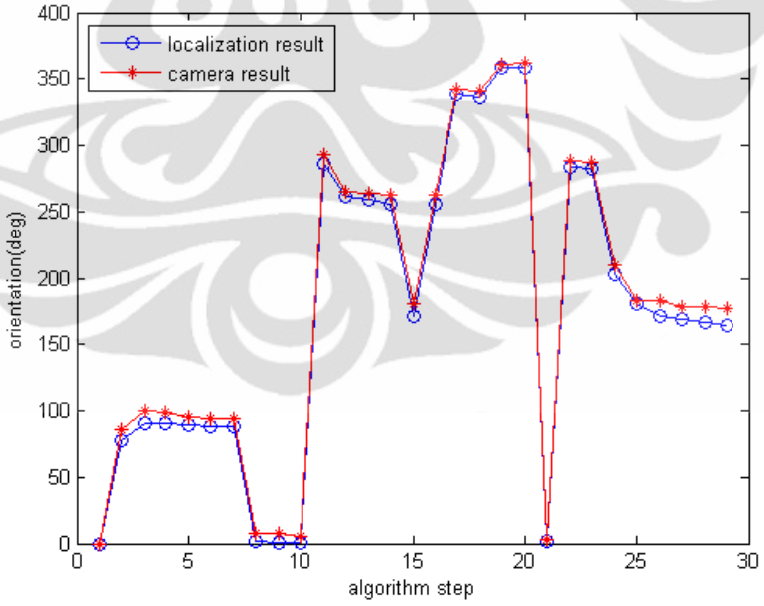
1. $\overline{err}_{pos} = 3.65cm$
2. $\overline{err}_{\theta} = 3.76°$
3. *Acceptable* : 75.86%

Dari error yang diperoleh, dapat dikatakan bahwa hasil lokalisasi yang optimal dapat diperoleh pada pengujian dengan menggunakan lingkungan uji tipe 2. Dengan kata lain, strategi landmark detection dapat digunakan untuk melakukan lokalisasi pada lingkungan berbentuk huruf "L" dengan arah gerakan berlawanan arah jarum jam.

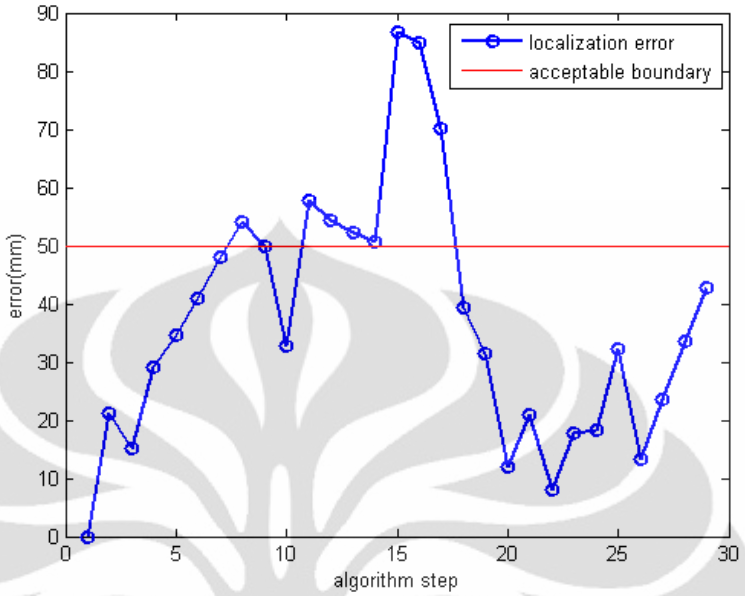
Berikutnya dilakukan pengujian dengan arah gerakan searah jarum jam. Hasil yang diperoleh ditunjukkan pada gambar 6.43-6.46



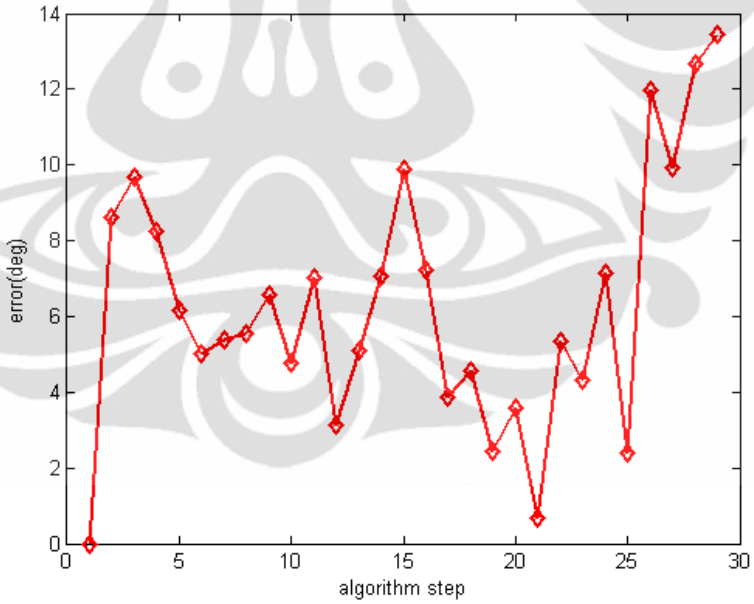
Gambar 6.43 Perbandingan poin (x,y) hasil lokalisasi dan hasil validasi kamera



Gambar 6.44 Perbandingan data orientasi hasil lokalisasi dengan hasil validasi kamera



Gambar 6.45 Plot error posisi x dan y



Gambar 6.46 Plot error orientasi

Berdasarkan hasil yang telah disajikan dalam gambar 6.43-6.46, didapat error sebagai berikut:

1. $\overline{err}_{pos} = 3.17cm$
2. $\overline{err}_{\theta} = 6.27°$

3. *Acceptable* : 72.41%

Hasil error yang didapat menunjukkan bahwa hasil optimal hanya dapat diperoleh untuk posisi x dan y, sedangkan untuk θ error yang terjadi masih di atas 5º. Akan tetapi, meskipun error yang terjadi untuk θ masih di atas batas yang diinginkan, nilai error tersebut tidak terlalu jauh dari batas. Dengan demikian, dapat dikatakan strategi *landmark detection* mampu melokalisasi robot pada lingkungan uji 2 dengan arah gerakan searah jarum jam, dengan hasil yang mendekati optimal. Lebih lanjut lagi, dari hasil pengujian dua araha gerakan, dapat dikatakan bahwa strategi *landmark detection* telah berhasil melokalisasi robot pada lingkungan uji 2.

6.2.4 Rangkuman Hasil

Berdasarkan penelitian yang telah dilakukan, dapat diperoleh karakteristik strategi lokalisasi yang digunakan. Karakteristik ini kemudian disajikan dalam tabel 6.1.

Tabel 6.1 Rangkuman karakteristik strategi lokalisasi

Strategi *Landmark Detection*	Strategi *Line Extraction*
1. Optimal pada semua variasi ruangan dan gerakan yang diujikan 2. Memiliki estimasi posisi (x,y) yang konsisten dan estimasi orientasi yang berkemungkinan tidak konsisten 3. Mampu mengupdate pose pada setiap step 4. Error terbesar dipengaruhi oleh gerakan berputar dan kesalahan pembacaan sonar karena sudut ruangan	1. Hanya optimal pada gerakan berlawanan arah jarum jam 2. Memiliki estimasi pose yang konsisten 3. Update tidak dapat dilakukan pada setiap step; hanya dapat dilakukan ketika garis berhasil diekstrak 4. Error terbesar dipengaruhi oleh gerakan berputar dan kesalahan pembacaan sonar karena sudut ruangan

BAB 7
PENUTUP

7.1 Kesimpulan

Dari penelitian yang dilakukan, dapat diperoleh kesimpulan sebagai berikut:

1. Lokalisasi dengan menggunakan *Extended Kalman Filter* telah berhasil dilakukan
2. Strategi lokalisasi menggunakan *landmark detection* memberikan hasil yang lebih baik daripada strategi lokalisasi menggunakan *line extraction* karena koreksi dapat dilakukan pada setiap *step*
3. Strategi lokalisasi menggunakan *landmark detection* telah mampu memberikan estimasi *pose* dengan error x dan y yang lebih kecil dari 3 cm dan error θ lebih kecil dari 5º
4. Strategi lokalisasi menggunakan landmark detection dapat melakukan estimasi *pose* yang memenuhi kriteria pada lintasan berlawanan arah jarum jam, searah jarum jam maupun pada lintasan yang mengandung kombinasi arah berlawanan dan searah jarum jam
5. Strategi lokalisasi menggunakan *landmark detection* memiliki hasil estimasi *pose* yang konsisten, namun ada kemungkinan estimasi orientasi yang dihasilkan dapat mengalami inkonsistensi
6. Strategi lokalisasi menggunakan *line extraction* belum mampu memberikan estimasi *pose* yang memenuhi kriteria untuk semua komponen. Error minimal yang dapat dihasilkan adalah 1.89 cm untuk posisi x, 4.62 cm untuk posisi y dan 2.31º untuk orientasi
7. Strategi lokalisasi menggunakan *line extraction* belum mampu mengestimasi *pose* untuk lintasan bebas yang mengandung kombinasi arah berlawanan dan searah jarum jam
8. Strategi lokalisasi menggunakan *line extraction* memiliki hasil estimasi *pose* yang konsisten
9. Pada lokalisasi,sumber error terbesar diperoleh dari ketidakpastian posisi akibat gerakan berputar dan ketidakpastian pada data yang diperoleh dari sensor jarak

10. Pada penerapan lokalisasi menggunakan algoritma *Extended Kalman Filter*, pengetahuan tentang matriks ***P***, ***Q***, dan ***R*** sangat mempengaruhi kualitas estimasi yang dihasilkan

7.2 Pengembangan Lebih Lanjut

Pengembangan yang dapat dilakukan pada penelitian ini untuk ke depannya antara lain adalah penggunaan occupancy grid untuk representasi lingkungan yang disimpan dalam peta internal robot. Representasi occupancy grid ini dapat lebih mengantisipasi ketidakpastian pembacaan sensor sonar. Selain itu, dapat pula diimplementasikan algoritma adaptif untuk mendapatkan nilai matriks ***Q*** dan ***R*** yang optimal. Pengembangan lain yang dapat dilakukan adalah implementasi IMU pada *mobile robot*, sehingga *mobile robot* memiliki kemampuan untuk melakukan lokalisasi pada lingkungan tak terstruktur sekalipun.

DAFTAR ACUAN

[1] Wikipedia. [Online]. http://en.wikipedia.org/wiki/History_of_robots

[2] Roland Siegwart and Illah R Nourbaksh, *Introduction to Autonomous Mobile Robot*. Massachusets: MIT Press, 2004, hal. 1-5.

[3] Erico Guizzo, "When My Avatar Went to Work," *IEEE Spectrum*, vol. 47, no. 9, hal. 24-29, Oktober 2010.

[4] Carlos Marques, "A search and rescue robot with tele-operated tether docking system," *Industrial Robot : An International Journal*, vol. 34, no. 4, hal. 332-338, 2007.

[5] Vektor Dewanto, "Kalibrasi Odometri pada Tracked Mobile Robot Alfathvrss," Universitas Indonesia, Depok, Seminar 2009.

[6] Luca Teslic, Igor Skrjanc, dan Gregor Klancar, "EKF-Based Localization of a Wheeled Mobile Robot in Structured Environment," *Journal of Intelligent Robot System*, 2010.

[7] Fred Martin, *The 6.270 Robot Builder's Guide*, 2nd ed. Massachusetts: The Massachusetts Institute of Technology, 1992.

[8] J Borenstein, H.R. Everett, dan L Feng, "Where Am I? Sensors and Methods for Mobile RobotPositioning," University of Michigan, 1996.

[9] Parallax. [Online]. http://www.parallax.com/tabid/768/ProductID/92/Default.aspx

[10] "PING))) Ultrasonic Distance Sensor (#28015)," Parallax Inc., Datasheet 2009.

[11] "CMPS03-Compass Module," Devantech, Datasheet 2007.

[12] Gordon McComb dan Myke Predko, *Robot Builder's Bonanza*. New York: McGraw-Hill, 2006, bab 20, hal. 367-368.

[13] Jeanne Sullivan Falcon, "Sensors and Actuators," dalam *Robotics and Automations Handbook*, Thomas R. Kurfess, Ed. New York: CRC Pres LLC, 2005, bab 12.

[14] "MCF51JM128 ColdFire Microcontroller," Freescale Semiconductor,

Datasheet : Technical Data 2009.

[15] "ATmega 16 datasheet," Atmel, Datasheet 2003.

[16] "TraxsterII Datasheet," RoboticsConnection, Datasheet.

[17] "YS1020U RF Transceiver datasheet," Hong Kong HuaWei Internatioal Electronics, Ltd., Datasheet.

[18] J Borenstein dan L Feng, "Correction of Systematic Odometry Errors in Mobile Robots," Prosiding dalam *The 1995 International Conference on Intelligent Robots and Systems*, Pittsburgh, 1995, hal. 569-574.

[19] Jason Gu, et. al, "Sensor Fusion in Mobile Robot : Some Perspectives," Prosiding dalam *The 4th World Congress on Intelligent Control and Automation*, Shanghai, 2002, hal. 1194-1199.

[20] Jeong-Gwan Kang, Choi Won-Seok, et. al, "Augmented EKF based SLAM Method for Improving the Accuracy of the Feature Map," dalam *The 2010 IEEE/RSJ Internatonal Conference on Intelligent Robots and Systems*, Taipei, 2010, hal. 3725-3731.

[21] Edouard Ivanjko, Mario Vasak, dan Ivan Petrovic, "Kalman Filter Theory Based Mobile Robot Pose Tracking Using Occupancy Grid Maps," dalam *2005 International Conference on Control and Automation (ICCA2005)*, Budapest, 2005, hal. 869-874.

[22] Gianluca Ippoliti, Alesia La Manna, dan Longhi Sauro, "Robust Robot Localization by Sensors with Different Degree of Accuracy," *Journal of Intelligent Robot Sytem*, vol. 56, hal. 259-276, 2009.

[23] Greg Welch dan Gary Bishop, "An Introduction to Kalman Filter," University of North Carolina at Chapel Hill, Chapel Hill, Course Material SIGGRAPH 2001.

[24] Jose-Luis Blanco, et al, "Mobile Robot Ego-Motion Estimation by Proprioceptive Sensor Fusion," dalam *9th International Symposium on Signal Processing and Its Application*, 2007.

[25] R.E. Kalman, "A new Approach to Linear Filtering and Prediction Problems," *Transaction of the ASME-Journals of Basic Engineering* , vol. 82, hal. 35-45, 1960.

[26] Simon Haykin, *Kalman Filtering and Neural Network*. New York: John Wiley and Sons, Inc, 2001, hal. 1-2.

[27] Simon Haykin, *Kalman Filtering and Neural Networks*. New York: John Wiley and Sons, Inc, 2001, hal. 5-9.

[28] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Orthogonality_principle

[29] Viet Nguyen, Agostino Martinelli, et. al, "A Comparison of Line Extraction Algorithms Using 2D Laser Rangefinder for Indoor Mobile Robotics," Prosiding dalam *The 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, hal. 1929-1934.

[30] J.L. Martinez, A. Mandow, et. al, "Kinematics Modelling of Tracked Vehicles by Experimental Identification," Prosiding dalam *The 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems* , Sendai, 2004, hal. 1487-1492.

[31] James L. Crowley, "Mathematical Foundations of Navigation and Perception For an Autonomous Mobile Robot," University of Amsterdam, Amsterdam, Tutorial, dipresentasikan pada International Workshop on Reasoning with Uncertainty in Robotics 1995.

[32] J. Borenstein and L. Feng, "UMBmark - A Method for Measuring, Comparing, and Correcting Dead-reckoning Errors in Mobile Robots," University of Michigan, Technical Report 1994.

[33] Leopoldo Armesto and Josep Tornero, "SLAM Based on Kalman Filter for Multi-rate Fusion of Laser and Encoder Measurements," dalam *The 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, 2004, hal. 1860-1865.

[34] J. Vandorpe, H. Van Brussel, dan H. Xu, "Exact Dynamic Map Building for a Mobile Robot using Geometrical Primitives Produced by a 2D Range Finder," dalam *International Conference on Robotics and Automation*, Minneapolis, 1996, hal. 901-908.

[35] Robin R Murphy, *Introduction to AI Robotics*. Massachusetts: The MIT Press, 2000.

DAFTAR PUSTAKA

Armesto, Leopoldo dan Josep Tornero. "SLAM Based on Kalman Filter for Multi-rate Fusion of Laser and Encoder Measurements ," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sendai. 2004. hal. 1860-1865.

Borenstein, J. dan L Feng, "Correction of Systematic Odometry Errors in Mobile Robots," Prosiding dalam *The 1995 International Conference on Intelligent Robots and Systems*. Pittsburgh. 1995. hal. 569-574.

Chanier, Francois, et. al. "SLAM Process using Polynomial Extended Kalman Filter : Experimental Assessment," in *10th International Conference on Control. Automation, Robotics and Vision*. Hanoi. 2008. hal. 365-370.

Dewanto, Vektor. "Kalibrasi Odometri pada Tracked Mobile Robot Alfathvrss," Universitas Indonesia. Depok. Seminar 2009.

_____. "Dynamic Monte Carlo Localization Untuk Tracked Mobile Robot," Universitas Indonesia. Depok. Skripsi 2010.

Haykin, Simon. *Kalman Filtering and Neural Network*. New York: John Wiley and Sons, Inc. 2001.

Kang, Jeong-Gwan, et. al. "Augmented EKF based SLAM Method for Improving the Accuracy of the Feature Map," dalam *The 2010 IEEE/RSJ Internatonal Conference on Intelligent Robots and Systems*. Taipei. 2010. hal. 3725-3731.

Martinez, J.L, et. al, "Kinematics Modelling of Tracked Vehicles by Experimental Identification," Prosiding dalam *The 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems* . Sendai. 2004. hal. 1487-1492.

Menegatti, Emanuelle, et. al. "Range-only SLAM with a Mobile Robot and a Wireless Sensor Networks," dalam *2009 IEEE International Conference on Robotics and Automation*. Kobe. 2009. hal. 8-14.

Teslic, Luca, et.al. "EKF-Based Localization of a Wheeled Mobile Robot in Structured Environment." *Journal of Intelligent Robot System*. 2010.

Welch, Greg dan Gary Bishop. "An Introduction to Kalman Filter," University of North Carolina at Chapel Hill. Chapel Hill. Course Material SIGGRAPH 2001.