



TESIS - TE142599

**AUTONOMOUS MOBILE ROBOT BERBASIS
LANDMARK MENGGUNAKAN PARTICLE FILTER
DAN OCCUPANCY GRID MAPS UNTUK NAVIGASI,
PENENTUAN POSISI, DAN PEMETAAN**

**Eko Budi Utomo
NRP.2212205016**

**DOSEN PEMBIMBING
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.**

**PROGRAM MAGISTER
BIDANG KEAHLIAN JARINGAN CERDAS MULTIMEDIA
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015**



THESIS - TE142599

**AUTONOMOUS MOBILE ROBOT BASED ON
LANDMARK USING PARTICLE FILTER AND
OCCUPANCY GRID MAPS FOR NAVIGATION,
LOCALIZATION, AND MAPPING**

**Eko Budi Utomo
NRP.2212205016**

SUPERVISOR

**Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.**

MAGISTER PROGRAM

**EXPERTISE FIELD OF MULTIMEDIA INTELLIGENT NETWORK
DEPARTMENT OF ELECTRICAL ENGINEERING
FACULTY OF INDUSTRIAL TECHNOLOGY
INSTITUTE TECHNOLOGY OF SEPULUH NOPEMBER
SURABAYA
2015**

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T.)

di

Institut Teknologi Sepuluh Nopember

oleh:

Eko Budi Utomo
NRP. 2212205016

Tanggal Ujian : 9 Januari 2015
Periode Wisuda: Maret 2015

Disetujui oleh:

1. Prof. Dr. Ir. Maurichi Hery Purnomo, M.Eng. (Pembimbing I)
NIP. 19580916 198601 1 001
2. Dr. Supeno Mardi Susiki Nugroho, S.T., M.T. (Pembimbing II)
NIP. 19700313 199512 1 001
3. Dr. Eko Mulyanto Yuniarno, S.T., M.T. (Penguji I)
NIP. 19680601 199512 1 009
4. Dr. Adhi Dharma Wibawa S.T., M.T. (Penguji II)
NIP. 19760505 200812 1 003
5. Dr. I Ketut Eddy Purnama, S.T., M.T. (Penguji III)
NIP. 19690730 199512 1 001

Direktur Program Pascasarjana,

Prof. Dr. Ir. Adi Soeprijanto, M.T.

NIP. 19640405 199002 1 001

Autonomous mobile robot berbasis landmark menggunakan particle filter dan occupancy grid maps untuk navigasi, penentuan posisi dan pemetaan.

Nama Mahasiswa : Eko Budi Utomo

NRP : 2212205016

Dosen Pembimbing : Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng

Co-Pembimbing : Dr. Supeno Mardi S.N. S.T, M.T

ABSTRAK

Implementasi SLAM umumnya menggunakan robot dengan sensor yang lengkap. Di penelitian ini menggunakan HBE Robocar dengan sensor terbatas untuk mengetahui keberhasilan unsur pembangun SLAM (*navigation, localization* dan *mapping*) dan solusi ideal SLAM.

Ketiga unsur SLAM direalisasikan dalam simulasi dan real robot dengan sistem persepsi sensor yang berbeda. Untuk penentuan posisi diberikan peta yang diketahui dan pergerakan robot menggunakan *landmark* berupa bola. Robot melakukan *tracking color* terhadap bola untuk mendapatkan *pose relative* dan diproses menggunakan algoritma *particle filter* dengan variasi jumlah partikel. Untuk navigasi, robot menghitung jalur terdekat dari *start* ke *goal* menggunakan algoritma *dynamic A**. Untuk pemetaan, digunakan *occupancy grid maps* dengan *pose global* robot (x, y, θ) diketahui.

Dari hasil pengujian, diperlukan penyesuaian untuk sistem SLAM ideal berupa penambahan sensor ultrasonik di sisi kiri dan kanan robot, sensor kompas dan kamera atas. Dan solusi yang dapat dilakukan adalah membuat gerakan berputar di tempat untuk mendeteksi penghalang dan mendapatkan data jarak pengganti sensor yang dipasang di kanan dan kiri robot. Partikel terbaik didapatkan pada iterasi ke 42 dengan jumlah partikel adalah 400 dan rata-rata *error* untuk nilai $x=3.41\%$, $y=4.03\%$, $\theta=65.79\%$. Semakin kecil jumlah partikel, diperlukan iterasi lebih banyak dalam estimasi *pose* (x, y, θ). Semakin banyak jumlah partikel, waktu komputasi yang diperlukan semakin lama. *Dynamic A** membantu robot menemukan jalur terpendek dengan *error* data arah sebesar 4.34% dan *error* jarak tempuh robot sebesar 4.8 %.

Kata kunci : *Navigation, Localization, Mapping, Particle Filter, Occupancy Grid Map, Dynamic A*, Tracking Color, Landmark, Ultrasonik, Kamera, Kompas*

Autonomous Mobile Robot based on Landmark using Particle Filter and Occupancy Grid Maps for Navigation, Localization and Mapping.

Name : Eko Budi Utomo
NRP : 2212205016
Supervisor : Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng
Co-Supervisor : Dr. Supeno Mardi S.N. S.T, M.T

ABSTRACT

SLAM implementation commonly using robot with multi sensor. In this research, HBE Robocar was used for understanding its SLAM Developing component (navigation, localization, and mapping) with sensor limitation and its ideal solution.

All of the SLAM components were conducted in simulation and real condition of robot with different sensor perception. Localization was conducted with known map and robot movement based on landmark using the balls. Robot does color tracking on the balls to get relative pose and processed using particle filter algorithm with various of particle quantity. Navigation was conducted by calculating shortest distance from start to goal using Dynamic A* algorithm. Mapping was conducted by using occupancy grid maps with known global robot position (x, y, theta).

Result shown that current robot needs compliance for ideal SLAM system by adding ultrasonic sensor in left and right side robot, compass sensor, and top camera. Best particle acquired in 42nd iteration with 400 particles and mean error for $x=3.41\%$, $y=4.03\%$, $\theta=65.79\%$. Less particles need more iterations for pose estimation (x,y,theta). More particles mean more computation times. Dynamic A* helps robot to find shortest distance with directional data error 4.34% and travel distance error 4.8%.

Keywords : *Navigation, Localization, Mapping, Particle Filter, Occupancy Grid Map, Dynamic A*, Tracking Color, Landmark, Ultrasonik, Kamera, Kompas*

KATA PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kepada Allah SWT yang telah melimpahkan berkah, rahmat, serta hidayah-Nya sehingga penulis dapat menyelesaikan tesis ini dengan judul ***Autonomous Mobile Robot berbasis Landmark menggunakan Particle Filter dan Occupancy Grid Maps untuk Navigasi, Penentuan posisi dan Pemetaan.***

Penelitian ini adalah persyaratan penulis untuk mendapatkan gelar Magister Teknik (M.T.) dari Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya. Pengerjaan tesis ini tak lepas dari bantuan berbagai pihak. Oleh karena itu penulis ingin menyampaikan terimakasih kepada:

1. Nanik Rachmawati, ibunda penulis dan Sugiarto, ayahanda penulis yang senantiasa memberikan dukungan, semangat serta doa untuk penulis.
2. *Beasiswa Fresh Graduate ITS* tahun 2012 yang telah memfasilitasi dan mendukung penuh penulis untuk menyelesaikan studi Magister di Teknik Elektro ITS.
3. Bapak Dr. Tri Arief Sardjono, S.T., M.T., selaku Ketua Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.
4. Bapak Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng., Bapak Supeno Mardi S.N S.T., M.T. dan Bapak Muhtadin, ST., MT. selaku dosen pembimbing yang selalu memberikan saran serta bantuan dalam penelitian ini.
5. Bapak Ibu dosen pengajar Jurusan Teknik Elektro ITS, yang telah memberikan banyak ilmu yang bermanfaat selama penulis menempuh kuliah S2.
6. Teman-teman Magister Teknologi Game Angkatan 2012 Ganjil (Benny, Citra, Ariyadi, Aidil, Yoze, Widyasari, Heny)
7. Rekan-rekan Dosen dan Karyawan / Teknisi D4 Teknik Mekatronika PENS yang memberikan dukungan serta semangat yang tiada henti dalam menyelesaikan studi dan tesis ini.

8. Teman-teman D4 Teknik Elektronika PENS angkatan 2008 yang selalu membawa semangat juang dalam menjalani hari-hari di perkuliahan.

9. Ayu Komala Dewi, S.Pd yang selalu menjadi semangat, penyejuk hati dan tempat berbagi ketika lelah, suka dan duka melanda.

Kesempurnaan hanya milik Allah SWT, penyusunan tesis ini tentu masih banyak kekurangan. Untuk itu penulis sangat mengharapkan kritik dan saran yang membangun. Semoga penelitian ini dapat memberikan manfaat bagi perkembangan ilmu pengetahuan dan teknologi serta masyarakat.

Surabaya, Januari 2015

Penulis

DAFTAR ISI

JUDUL	I
KATA PENGANTAR.....	V
ABSTRAK	VII
ABSTRACT	IX
DAFTAR ISI.....	XI
DAFTAR GAMBAR.....	XIII
DAFTAR TABEL	XVII
BAB 1 PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Perumusan masalah	5
1.3 Tujuan.....	6
1.4 Metodologi penelitian	6
BAB 2 KAJIAN PUSTAKA	9
2.1 <i>Probabilistic Robotics</i>	9
2.1.1 Teorema Bayes.....	13
2.2 Robot Motion	18
2.2.1 <i>Velocity Model</i>	19
2.2.2 <i>Odometry Model</i>	20
2.3 Penentuan Posisi.....	20
2.4 Representasi Hasil <i>Mapping</i>	22
2.5 Algoritma Partikel Filter	24
2.6 Algoritma Dinamik A-Star.....	27
2.7 Metode <i>Occupancy Grid Maps</i>	31
2.8 HBE Robocar <i>Embedded</i>	32
BAB 3 DESAIN DAN IMPLEMENTASI SISTEM.....	39
3.1 Diagram Blok Sistem	39

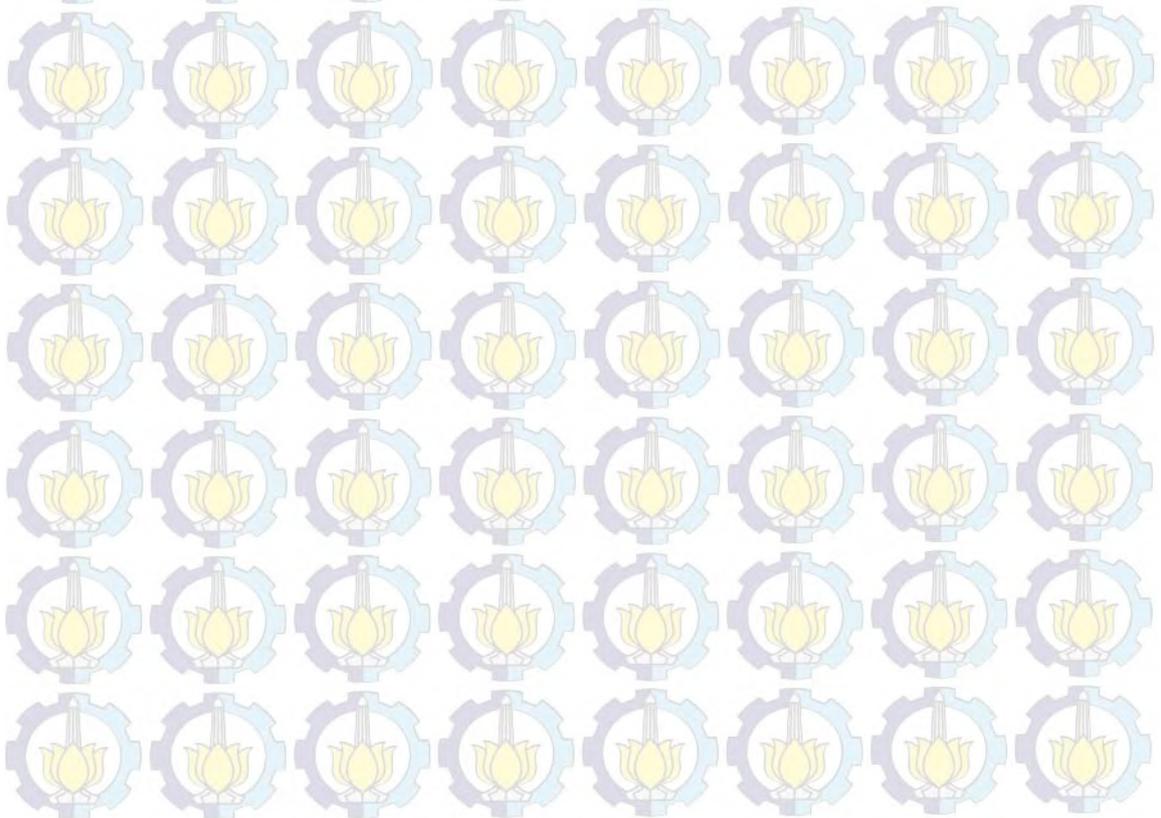
3.2	Diagram Alir Sistem	41
3.3	Perangkat Keras Sistem	42
3.4	Penentuan Posisi menggunakan algoritma Partikel Filter	44
3.4.1	<i>Sensing</i> Model.....	45
3.4.2	<i>Motion</i> Model	46
3.4.3	<i>Resampling</i>	51
3.5	Perencanaan jalur menggunakan Algoritma Dinamik A*	53
3.6	<i>Graphical User Interface (GUI)</i>	56
BAB 4	PENGUJIAN SISTEM DAN ANALISA	59
4.1	Pengujian penentuan posisi menggunakan partikel filter (simulasi) ..	59
4.2	Pengujian penentuan posisi menggunakan partikel filter (robot)	62
4.3	Pengujian navigasi menggunakan algoritma dinamik A*(simulasi) ..	65
4.4	Pengujian navigasi menggunakan algoritma dinamik A*(robot)	67
4.5	Pengujian <i>mapping</i> menggunakan Occupancy Grid Maps (simulasi) ..	71
4.6	Pengujian <i>mapping</i> menggunakan Occupancy Grid Maps (robot).....	74
BAB 5	KESIMPULAN	79
5.1	Kesimpulan	79
5.2	Penelitian Selanjutnya	80
DAFTAR PUSTAKA	81
BIOGRAFI PENULIS	81

DAFTAR GAMBAR

Gambar 1.1 Ilustrasi penentuan posisi robot	1
Gambar 1.2 Ilustrasi <i>mapping</i> pada robot.....	2
Gambar 1.3 HBE-Robocar <i>Embedded</i>	3
Gambar 1.4 Tata letak <i>sensor</i> dan <i>actuator</i> pada HBE Robocar.....	4
Gambar 2.1 Robot saat mendeteksi kondisi pintu	14
Gambar 2.2 Distribusi normal (kiri) dan triangular distribusi (kanan)	19
Gambar 2.3 Sistem Odometri robot.....	20
Gambar 2.4 Posisi awal robot.....	21
Gambar 2.5 Kemungkinan Letak Robot.....	22
Gambar 2.6 Contoh <i>Geometric Feature Map</i> [12].....	23
Gambar 2.7 Contoh <i>Grid Map</i> [12].....	23
Gambar 2.8 Proses update pada <i>Occupancy Grid</i> [13]	23
Gambar 2.9 <i>Topological Map</i> [12].....	24
Gambar 2.10 Penyebaran partikel secara acak	25
Gambar 2.11 Penentuan <i>weight</i>	26
Gambar 2.12 <i>Pseudocode resampling process</i>	26
Gambar 2.13 Hasil <i>Resampling</i>	27
Gambar 2.14 Konvergen partikel.	27
Gambar 2.15 Posisi titik <i>start</i> dan titik <i>goal</i>	29
Gambar 2.16 Pemberian nilai $g(n)$ dan $h(n)$ untuk setiap arah pada <i>start</i> dan <i>goal</i>	29
Gambar 2.17 Nilai $f(n)$ untuk grid sekitar titik start.....	30
Gambar 2.18 Nilai $f(n)$ dari <i>start</i> sampai <i>goal</i>	31
Gambar 2.19 HBE Robocar <i>Embedded</i> [14].....	32
Gambar 2.20 HBE Robocar dan <i>Embedded Camera</i> [14]	33
Gambar 2.21 Bagan sistem HBE Robocar [14].....	33
Gambar 2.22 <i>Body</i> HBE Robocar [14] dan <i>board</i> di dalamnya.....	34
Gambar 2.23 Bagan sistem Robocar <i>Embedded</i> [14].....	35
Gambar 2.24 Beberapa <i>peripheral</i> dalam Robocar <i>Embedded</i> [14].....	35
Gambar 3.1 Diagram Blok sistem SLAM	39

Gambar 3.2 Sistem koordinat <i>global</i> dan sistem koordinat robot.....	40
Gambar 3.3 Diagram alir proses SLAM.....	41
Gambar 3.4 <i>Setting</i> Konfigurasi Sistem.....	42
Gambar 3.5 <i>Pseudo code particle filter</i>	44
Gambar 3.6 Sistem persepsi yang digunakan dalam simulasi.....	45
Gambar 3.7 <i>Tracking color</i> pada robot.....	45
Gambar 3.8 Sistem persepsi yang digunakan dalam robot.....	46
Gambar 3.9 Sistem Koordinat <i>pose</i>	47
Gambar 3.10 Odometri robot.....	48
Gambar 3.11 Sistem Koordinat Tujuan.....	48
Gambar 3.12 Rotasi roda untuk perhitungan jarak [14].....	49
Gambar 3.13 Perputaran robot pada sumbu <i>center of mass</i> [14].....	50
Gambar 3.14 Robot berputar di tempat [14].....	51
Gambar 3.15 Penyebaran partikel secara acak.....	52
Gambar 3.16 <i>Resampling</i> ke 11 proses penentuan bobot.....	52
Gambar 3.17 <i>Resampling</i> ke 17 partikel dengan bobot tinggi.....	53
Gambar 3.18 <i>Resampling</i> ke 39 penyebaran partikel di sekitar partikel bobot tinggi.....	53
Gambar 3.19 Perhitungan $g(n)$ dan $h(n)$ dari start ke goal.....	54
Gambar 3.20 Perhitungan ulang $f(n)$	54
Gambar 3.21 Perhitungan ulang untuk setiap grid yang diketahui sebagai obstacle $f(n)$	55
Gambar 3.22 Robot mencapai titik goal.....	55
Gambar 3.23 Pengukuran radius putaran roda robot [22].....	56
Gambar 3.24 Tampilan User Interface.....	57
Gambar 4.1 Titik-titik pengujian.....	62
Gambar 4.2 Persepsi orientasi robot.....	63
Gambar 4.3 Posisi robot saat melakukan pengamatan <i>landmark</i> bola orange (kiri) bola biru (kanan).....	64
Gambar 4.4 <i>Start</i> awal robot.....	65
Gambar 4.5 Robot saat mengikuti <i>track</i>	66
Gambar 4.6 Robot mencari jalur alternative.....	66

Gambar 4.7 Robot kembali mencari jalur alternatif.....	67
Gambar 4.8 Robot mencapai titik <i>finish</i>	67
Gambar 4.9 Pengujian putaran 0 derajat (kiri) dan putaran 45 derajat (kanan) ..	68
Gambar 4.10 Pengujian putaran 90 derajat (kiri) dan putaran 270 derajat (kanan)	68
Gambar 4.11 Perbandingan lebar <i>beam</i> pada sensor simulasi (kiri) dan pada robot (kanan).....	71
Gambar 4.12 Robot melakukan <i>scan</i> awal.....	72
Gambar 4.13 Robot melakukan penelusuran.....	72
Gambar 4.14 Hasil <i>mapping</i> sementara.....	73
Gambar 4.15 Hasil <i>mapping</i> untuk warna hijau (tidak <i>occupied</i>) warna putih (<i>occupied</i>).....	73
Gambar 4.16 Nilai grid untuk peta yang tidak ter <i>occupied</i> (hitam) dan derajat probabilitas pengukuran sensor (warna hijau).....	74
Gambar 4.17 Nilai grid untuk peta yang ter <i>occupied</i> , tidak ter <i>occupied</i> dan penghalang.....	74
Gambar 4.18 Solusi gerakan berputar (kiri) dan penambahan sensor (kanan)....	77



DAFTAR TABEL

Tabel 1.1 <i>Sensor dan Aktuator</i> HBE Robocar [14]	4
Tabel 2.1 Detail sistem perangkat pada HBE-Robocar [14]	34
Tabel 2.2 Spesifikasi CPU [14].....	36
Tabel 2.3 Spesifikasi <i>peripheral</i> 1 [14].....	36
Tabel 2.4 Spesifikasi <i>peripheral</i> 2 [14].....	37
Tabel 3.1 Perintah Komunikasi <i>Serial</i>	56
Tabel 4.1 Hasil penentuan posisi menggunakan 600 partikel.	59
Tabel 4.2 Hasil penentuan posisi menggunakan 400 partikel.	60
Tabel 4.3 Hasil penentuan posisi menggunakan 200 partikel.	61
Tabel 4.4 Data perbandingan estimasi posisi dengan jumlah partikel.	62
Tabel 4.5 Pengujian estimasi <i>pose</i> robot.	64
Tabel 4.6 Data pengujian ke-1. Translasi dan rotasi robot saat perintah <i>serial</i> diberikan	68
Tabel 4.7 Data pengujian ke-10. Translasi dan rotasi robot saat perintah <i>serial</i> diberikan	69
Tabel 4.8 Data hasil pengujian <i>dynamic A*</i> pada robot.	69
Tabel 4.9 Hasil <i>Mapping</i> (Robot).....	75

BAB 1

PENDAHULUAN

1.1 Latar Belakang

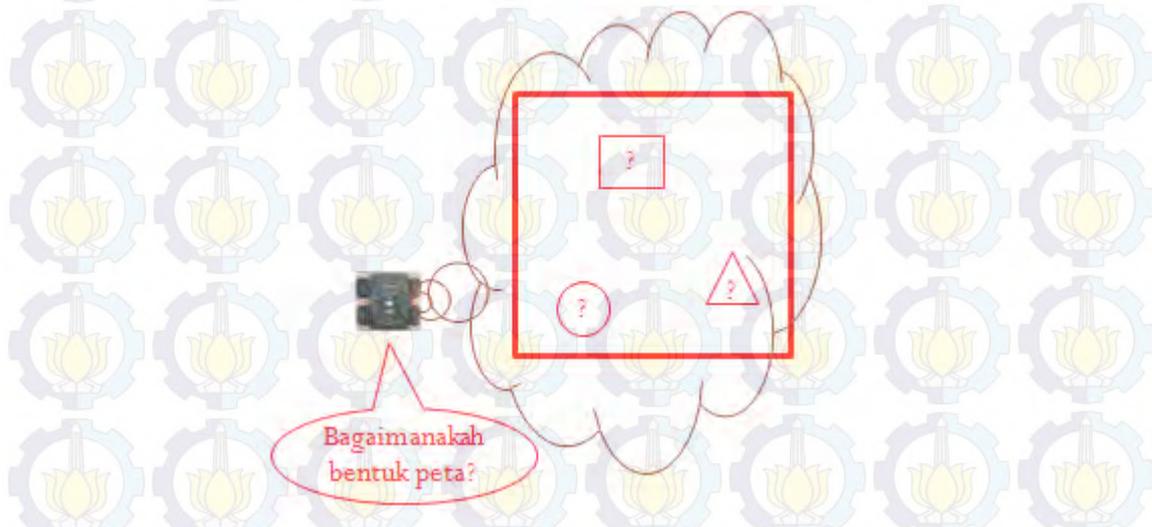
Permasalahan yang dihadapi sebuah *autonomous mobile robot* adalah masalah navigasi, masalah penentuan posisi dan masalah pemetaan. Navigasi diartikan sebagai proses atau aktivitas untuk merencanakan atau menuju jalur secara langsung dalam sebuah misi yang diberikan pada sebuah *autonomous mobile robot* dari satu tempat ke tempat yang lain tanpa kehilangan arah atau mengalami tabrakan dengan *object* yang lain. Kemudian penentuan posisi diartikan sebagai proses penentuan posisi lokal robot terhadap lingkungan atau peta yang diberikan.



Gambar 1.1 Ilustrasi penentuan posisi robot

Pada gambar 1.1 digambarkan robot diletakkan pada posisi tertentu dan muncul pertanyaan dimanakah saya? pertanyaan koordinat letak robot terhadap peta tersebut. Selanjutnya adalah pemetaan. Pemetaan diartikan sebagai proses robot membangun peta hasil penelusuran. Pada gambar 1.2 digambarkan peta lingkungan yang belum diketahui dan posisi robot diketahui. Dengan diketahuinya posisi robot maka posisi tersebut merupakan posisi awal robot untuk menelusuri lingkungan. Dengan navigasi robot, setiap gerakan robot (maju, mundur, kanan, kiri, mendeteksi penghalang, dan pengukuran odometri) robot diharapkan bisa

membangun hasil peta penelusuran sehingga mempunyai gambaran seperti apa peta lingkungan yang dilalui robot.



Gambar 1.2 Ilustrasi *mapping* pada robot.

Untuk melakukan penentuan posisi atau pemetaan robot memerlukan proses navigasi atau bergerak. Tiga unsur yang disebutkan sebelumnya yaitu navigasi, penentuan posisi dan pemetaan adalah unsur pembangun SLAM. SLAM (*Simultaneous Localization And Mapping*) diartikan sebagai proses yang harus dilakukan robot untuk melakukan penentuan posisi dan pembangunan peta dalam waktu yang bersamaan. Mengapa robot harus melakukan penentuan posisi dan pembangunan peta dalam waktu bersamaan? karena dalam kasus SLAM, robot akan dijalankan pada lingkungan yang belum diketahui atau belum didefinisikan sebelumnya. Padahal untuk bergerak robot membutuhkan peta, sedangkan peta dalam hal ini belum diberikan atau belum diketahui. Jika peta belum diketahui, maka proses penentuan posisi juga tidak dapat dilakukan. Dua permasalahan yang diibaratkan seperti permasalahan mana yang lebih dulu antara “*egg or chicken?*”, “*map or motion?*” [1].

Penelitian ini adalah tahapan penelitian untuk menuju SLAM. Langkah yang dilakukan adalah dengan membangun satu per satu dari ketiga unsur yaitu navigasi, penentuan posisi dan pemetaan.

Dalam penentuan posisi, beberapa penelitian sebelumnya digunakan metode *Extended Kalman Filter* [2], *Monte Carlo Localization* [3], *Rao*

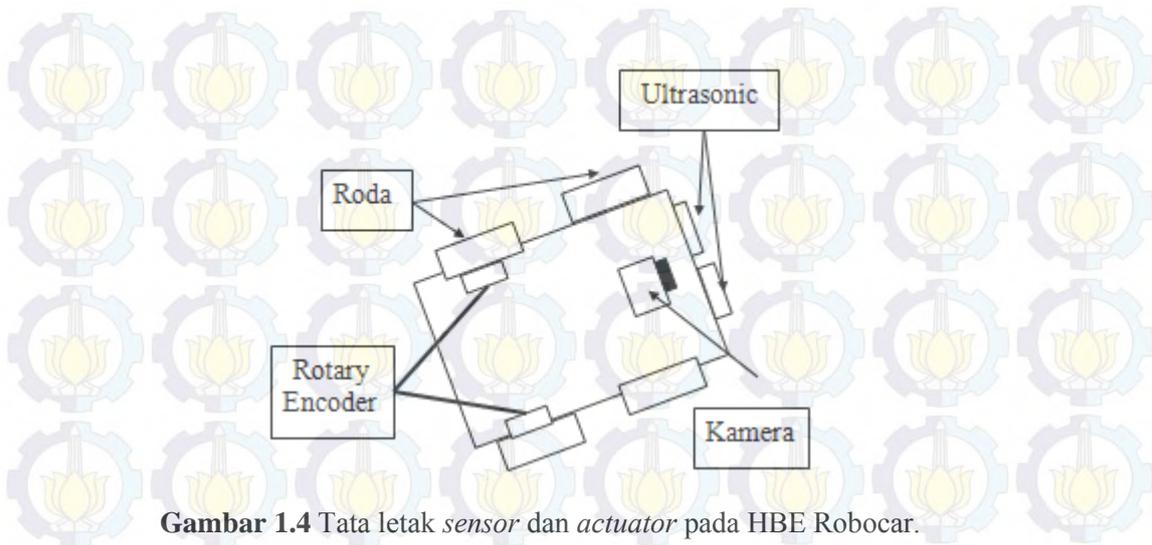
Blackwellized Particle Filter [4], Dalam penelitian ini penentuan posisi dilakukan dengan kondisi peta sudah diketahui dan pergerakan robot menggunakan pemandu *landmark* berupa bola yang dipasang di sudut lapangan. Robot melakukan *tracking color* terhadap bola untuk mendapatkan data *pose* relatif. Data *pose* relatif tersebut akan diproses menggunakan algoritma *particle filter* dengan *customize* jumlah partikel yang digunakan. Untuk navigasi, robot diberikan pengetahuan berupa tujuan robot yang harus dilalui dari *start point* (titik awal) ke *goal point* (titik akhir) menggunakan algoritma A* (A-Star). Algoritma ini digunakan untuk mendapatkan jarak terdekat dari titik awal (*starting point*) ke titik tujuan (*goal*). Kemudian untuk pemetaan, menggunakan metode *occupancy grid maps* dengan posisi robot (x, y, θ) sudah diketahui.

Ketiga konfigurasi tersebut dibangun pada robot dengan platform HBE Robocar.



Gambar 1.3 HBE-Robocar *Embedded*

Pada HBE Robocar terdapat beberapa perangkat keras meliputi *sensor*, *actuator* dan *processor* ditunjukkan pada Gambar 1.4.



Gambar 1.4 Tata letak sensor dan actuator pada HBE Robocar.

Untuk detail konfigurasi yang dipakai HBE Robocar adalah menggunakan *dual processor* ATmega 128 sebagai kontrol utama dan ATmega 8 untuk mengolah tegangan *analog* baterai yang ditampilkan pada *display 7 segment*. HBE Robocar mempunyai dua sensor ultrasonik yang berada didepan, 1 kamera dan 1 *rotary encoder*.

Tabel 1.1 Sensor dan Actuator HBE Robocar [14]

구분	사양
ATmega128L	8-bit AVR, Microcontroller with 128K Bytes, main control
ATmega8L	8-bit AVR, Microcontroller with 8K Bytes, Voltmeter control
L298P	Up to 4A DC Motor Driver 2EA
Ultrasonic sensor	40.0 ± 0.5KHz Frequency, 2.0KHz Bandwidth, 2EA
Accelerometer sensor	Dual-Axis Accelerometer sensor 1EA, Duty Cycle
PSD sensor	Distance Measuring Sensors 1EA, 10-80cm
Phototransistors	8-groups Infrared rays sensor
Motors	DC geared motor 2EA, DC geared encoder motor 2EA
Buzzer	5V input Buzzer 1EA
LED	10mm high brightness LED, White 2EA, RED 2EA
7-Segment	Voltmeter Display, 3-Digit 1EA
Regulator	+11.1V DC Input, DC output : +5V, +3.3V
Battery	+11.1V, 5200mA Lithium Ion 1EA
Charge & Adapter	+12.6V 1.2A Battery Charger 1EA

Implementasi SLAM dalam penelitian sebelumnya [5]-[10], umumnya menggunakan robot dengan sensor yang lengkap.

	Sensors		
	Kinect	Hokuyo	SICK
Maximum range [m]	3-6	4	8-80
Dead range [m]	0.8/0.4	0.06	0.07
Horizontal angle [°]	57	240	100-180
Distance resolution [mm]	2.5-48	1	1-10
Angular resolution [°]	≈ 0.097	0.3515	0.25-1
Accuracy [mm]	+/-6 (1m) +/-130 (4m)	+/-30 (1m) +/-120 (4m)	+/-10 (10m)
Geometry [mm]	65x290x70	50x50x70	155x156x210
Weight [kg]	0.55	0.16	4.5
Power voltage [V]	12	5	24
Power consumption [W]	5	4	30
Refresh rate [Hz]	30	10	18-75
Output data [kB/s]	18000	5.4	500
Interfaces	USB	USB	RS232, RS422
approx. costs \$	150	1000	5000



Gambar 1.5 Robot dan spesifikasi yang digunakan untuk implemmentasi SLAM pada [6].

Jika dibandingkan dengan kemampuan *sensor* serta aktuator yang dimiliki HBE Robocar maka dalam penelitian ini dilakukan pengujian apakah robot dengan keterbatasan sensor dapat melakukan unsur pembangun SLAM (*navigation, localization* dan *mapping*). Selain itu diperlukan solusi jika SLAM ingin diterapkan pada robot berkemampuan terbatas seperti HBE Robocar.

1.2 Perumusan masalah

Permasalahan pada penelitian ini adalah sebagai berikut:

1. Diperlukan pengujian apakah robot dengan sensor terbatas memiliki kemampuan untuk melakukan unsur pembangun SLAM (*navigasi, penentuan posisi dan pemetaan*).
2. Belum adanya solusi yang harus dilakukan jika unsur pembangun SLAM diaplikasikan pada robot yang memiliki sumber daya (*sensor, aktuator, processor*) terbatas seperti HBE Robocar.

Sehubungan dengan pemakaian beberapa algoritma yang digunakan dalam penelitian ini maka terdapat sub permasalahan sebagai berikut:

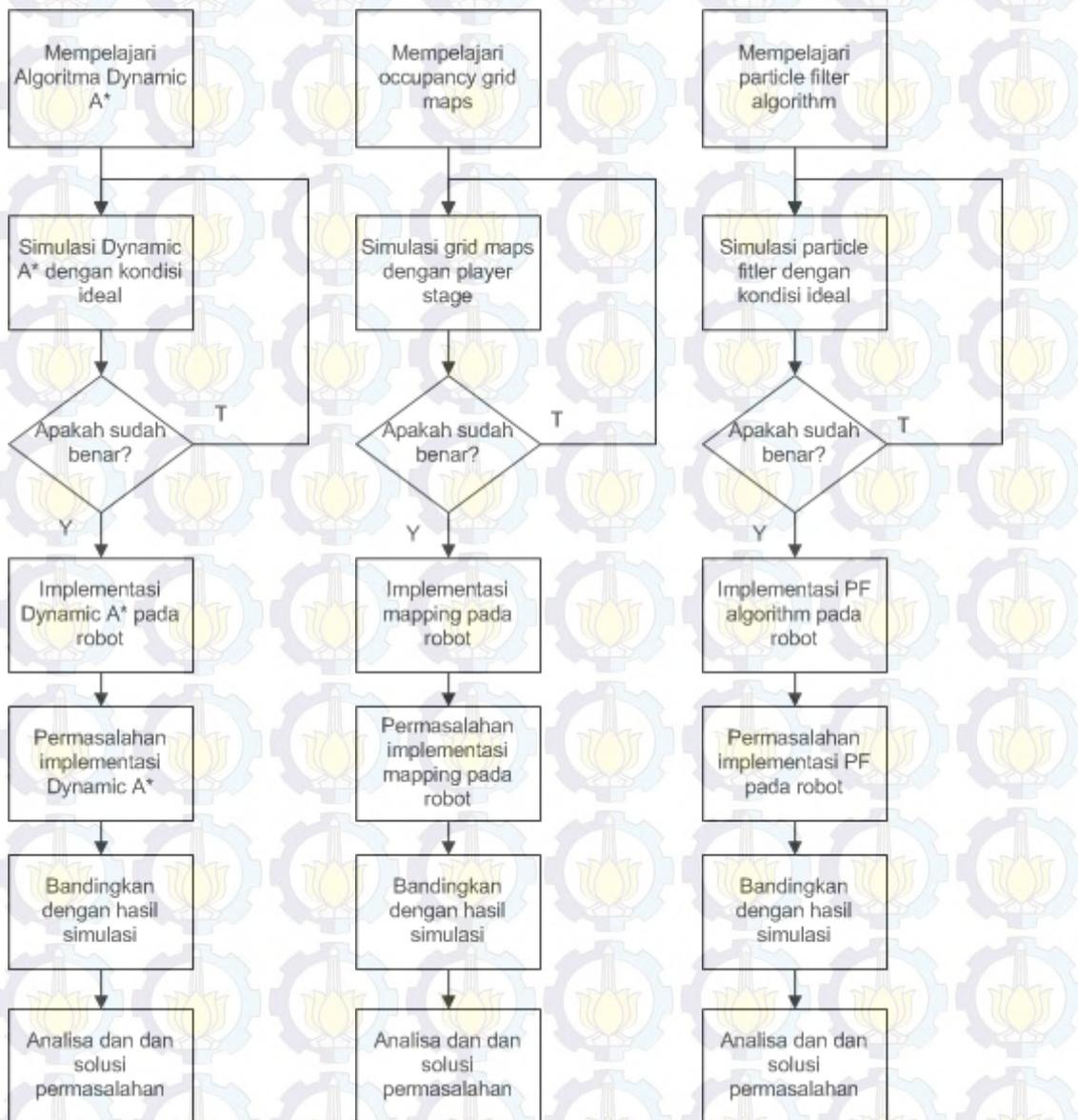
1. Apakah pemakaian partikel dalam jumlah banyak memberikan waktu komputasi yang lama?
2. Belum adanya penerapan *dynamic A** pada mobile robot secara *online*
3. Bagaimanakah hasil *occupancy grid maps* untuk *mapping*.

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah:

1. Menerapkan metode metode pembangun *SLAM* yaitu navigasi, pemetaan dan penentuan lokasi pada robot yang memiliki keterbatasan sensor (HBE Robocar).
2. Menjelaskan solusi-solusi yang sesuai agar HBE robocar mampu melakukan *SLAM*.

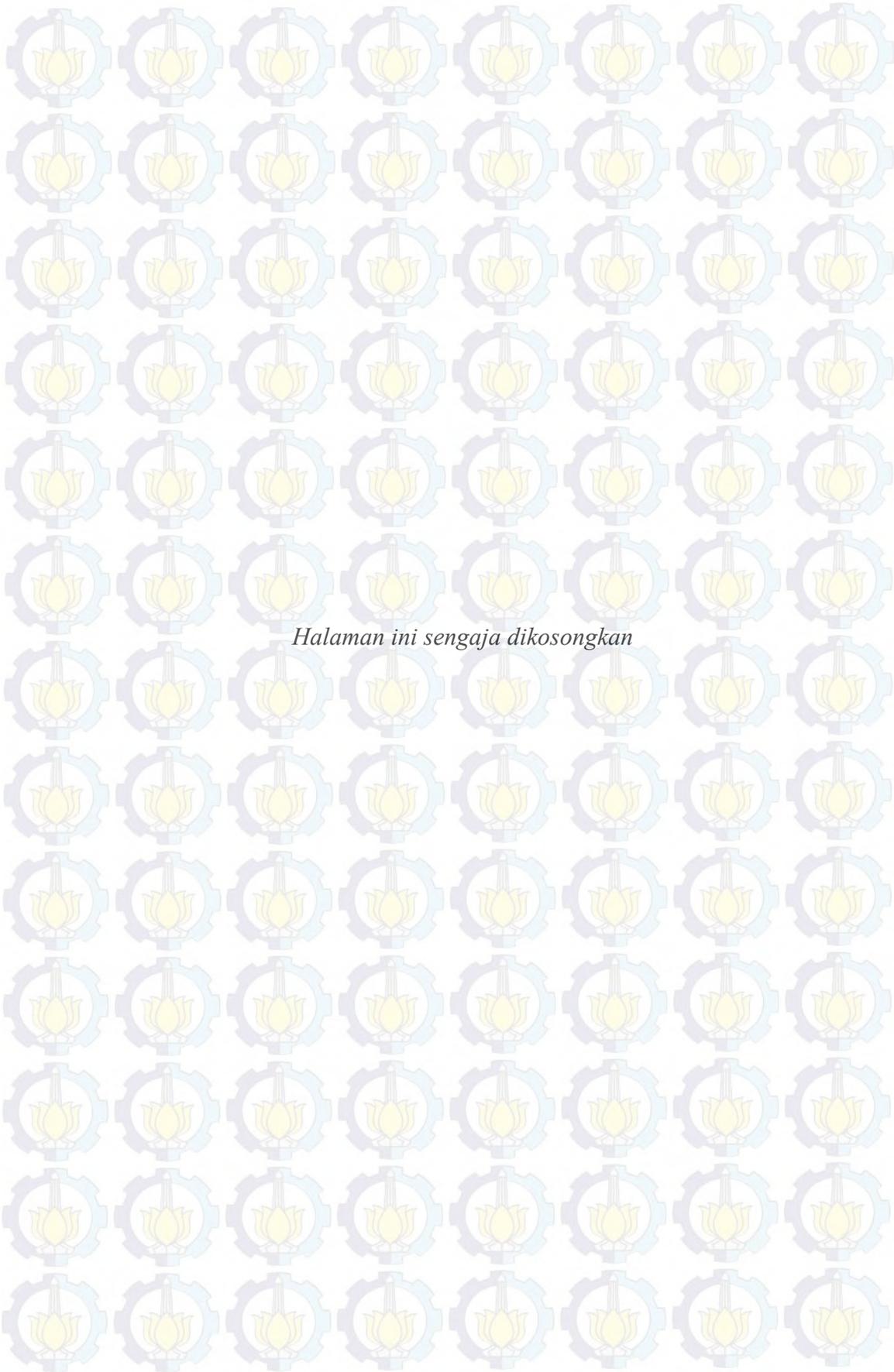
1.4 Metodologi penelitian



Gambar 1.6 Diagram alir metodologi penelitian.

Dalam penelitian ini terdapat 3 unsur SLAM yang akan dibangun yaitu navigasi, penentuan posisi dan pemetaan. Untuk merealisasikan sistem tersebut maka metodologi atau tahapan penelitian yang dilakukan terdiri dari 3 diagram alir pada gambar 1.6. Untuk diagram alir pertama adalah untuk proses navigasi. Setiap proses dibangun dengan melakukan simulasi terlebih dahulu dengan anggapan hal-hal ideal dan fungsional suatu algoritma dapat direalisasikan. Setelah dibangun simulasi atas sistem tersebut maka dilanjutkan dengan implementasi pada robot yang sebenarnya.

Pada saat implementasi pada robot tentunya ada keterbatasan-keterbatasan serta permasalahan meliputi perangkat keras, perangkat lunak serta pengukuran lain yang mempengaruhi. Untuk itu hasil yang didapatkan dari implementasi robot akan dibandingkan dengan hasil simulasi. Sehingga didapatkan hasil perbandingan dan analisa perbedaan jika suatu sistem dibangun menggunakan simulasi dengan penerapan sistem pada robot yang sebenarnya. Metodologi ini berlaku untuk 3 unsur yang akan dibangun yaitu algoritma *dynamic A**, algoritma *particle filter* dan *occupancy grid maps*. Hasil akhir yang diharapkan adalah solusi atas setiap penerapan algoritma pada robot yang sebenarnya. Kemudian bentuk-bentuk penyesuaian atau tambahan sistem yang diperlukan adalah digunakan untuk implementasi *SLAM* pada robot yang sebenarnya.



Halaman ini sengaja dikosongkan

BAB 2 KAJIAN PUSTAKA

2.1 Probabilistic Robotics

Dalam *Probabilistic Robotics* [11]. Ketidakpastian (*uncertainty*) dalam dunia robotika meliputi:

1. Lingkungan.

Pada proses eksplorasi lingkungan yang dilakukan oleh robot. lingkungan dianggap bersifat dinamik, sangat luas dan tidak dapat diprediksi. Komposisi alam misalnya tumbuhan, angin, air, manusia menjadi satu kondisi yang memberikan permasalahan dinamik bagi robot untuk memahami lingkungan tersebut.

2. Sensor.

Sensor atau indra bagi robot juga bersifat dinamik. Kita tidak dapat menganggap bahwa sensor untuk robot adalah ideal (tidak ada *error*). Keterbatasan sensor meliputi jangkauan (*range*) dan resolusi (ketepatan / kepresisian) untuk menyatakan satuan dari pengukuran. Sebagai contoh sensor jarak yang mempunyai *beam* (lebar pancaran) yang bermacam-macam, tentunya galat pengukuran yang kita dapatkan juga beragam. Contoh selanjutnya kamera yang sangat dipengaruhi oleh faktor pencahayaan, resolusi, kedalaman serta *noise-noise* yang berasal dari lingkungan juga mempengaruhi hasil pengukuran.

3. Aktuator.

Akurasi dari sebuah aktuator juga menjadi penyebab dalam ketidakpastian dalam dunia robotika untuk menyelesaikan suatu permasalahan yang bersifat ideal. Sebagai contoh pada *arm robot*, motor penggerak dengan derajat *gear* memberikan pengaruh *error* letak / posisi dari EoE (*End of Effector*)

4. Komputasi.

Robot diharapkan menjadi sistem yang *realtime* dalam mengolah suatu proses. Tapi dalam kenyataannya fungsi waktu dibutuhkan sehingga mempengaruhi proses yang terjadi. Jika suatu robot mempunyai komputasi yang tinggi maka suatu proses dapat diselesaikan dalam waktu yang cepat. Misalnya robot dengan kemampuan 100 MIPS (*Million Instruction Per Second*) pasti lebih

cepat dalam mengolah proses dibandingkan robot dengan kemampuan 10 MIPS (*Million Instruction Per Second*).

Dalam *probabilistics robotics* sejumlah variabel seperti pengukuran sensor, kontrol, dan posisi robot diasumsikan sebagai variabel yang tidak pasti artinya meskipun dalam pengukuran kita mendapatkan nilai, tapi nilai tersebut mempunyai *range* (jangkauan) yang terkadang sewaktu-waktu juga dapat berubah. Meskipun tidak pasti atau tidak dapat diprediksi tetapi nilai tersebut tetap kita butuhkan untuk input sebuah proses. Pada pembahasan ini akan dijelaskan contoh permasalahan yang dihadapi robot pada lingkungan yang bersifat *undeterministic* dan pembahasan tentang simbol dan notasi yang dipakai dalam *probabilistic robotics*.

X adalah variabel acak yang dapat diisi dengan semua variabel misalnya jarak, kecepatan, dll. x adalah dari kejadian (event) tertentu. Sebagai contoh kejadian pada koin mata uang. Probabilitas *random* variabel X mempunyai nilai x , maka kita dapat menuliskan

$$p(X = x) \quad (2.1)$$

Untuk menyatakan kemungkinan ($X = \text{nilai}$) = $p(X = \text{gambar}) = \frac{1}{2}$. Probabilitas ini disebut diskrit dengan jumlah semua kemungkinan adalah 1. Untuk itu kita tuliskan

$$\sum p(X = x) = 1 \quad (2.2)$$

Dan tentunya suatu probabilitas tidak bernilai negatif, $p(X = x) \geq 0$. Keadaan diatas adalah untuk menyatakan nilai yang diskrit (hanya ada 2 kemungkinan). Sedangkan untuk menyatakan estimasi dan pengambilan keputusan atas keadaan yang bersifat kontinyu digunakan *probability density function* (PDF). PDF adalah sebuah distribusi normal dengan unsur rata-rata μ (*mean*) dan σ^2 (*variance*) yang dituliskan dalam bentuk sebagai berikut:

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}\right\} \quad (2.3)$$

Distribusi normal mempunyai peran penting untuk menyatakan rata-rata nilai dengan standar deviasi suatu pengukuran atau galat dari suatu pengukuran.

Biasanya untuk menyatakan suatu distribusi normal digunakan penulisan singkat $N(x; \mu, \sigma^2)$ dengan keterangan variabel yang digunakan adalah rata-rata dan varians dari variabel yang digunakan. Untuk menyatakan probabilitas diskrit dalam bentuk PDF (*Probability Density Function*) maka dituliskan:

$$\int p(x)dx = 1 \quad (2.4)$$

Ini juga berlaku untuk variabel yang lebih dari satu. Diasumsikan semua variabel yang akan digunakan nanti adalah tidak hanya bersifat diskrit, tapi juga bersifat kontinyu. Dua variabel yang kontinyu mempunyai probabilitas sebagai berikut:

$$p(x, y) = p(X = x \text{ dan } Y = y) \quad (2.5)$$

Bentuk diatas menunjukkan probabilitas dua variabel yang saling berkaitan yaitu nilai x yang didasarkan pada nilai y atau sebaliknya. Jika dalam kasus X dan Y adalah tidak terikat / tidak ada hubungan maka berlaku:

$$p(x, y) = p(x)p(y) \quad (2.6)$$

Seringkali dua probabilitas variabel dianggap mempunyai ketergantungan atau bersyarat. Lebih jauh bisa kita asumsikan kita ingin mengetahui probabilitas X dari Y atau sebaliknya. Maka dituliskan

$$p(x|y) = p(X = x|Y = y) \quad (2.7)$$

Kondisi ini disebut probabilitas bersyarat. Dan jika $p(y) = 0$ probabilitas bersyarat dinyatakan dengan

$$p(x|y) = \frac{p(x,y)}{p(y)} \quad (2.8)$$

Dan jika X dan Y tidak saling berhubungan maka

$$p(x|y) = \frac{p(x)p(y)}{p(y)} = p(x) \quad (2.9)$$

Dapat dituliskan kembali untuk diskrit dan kontinyu sebagai berikut:

$$p(x) = \sum p(x|y)p(y) \quad (\text{diskrit})$$

$$p(x) = \int p(x|y)p(y)dy \quad (\text{kontinyu}) \quad (2.10)$$

Jika $p(x|y)$ atau $p(y) = 0$ maka kita bisa mendapatkan $p(x|y)p(y)$ menjadi 0. Bentuk ini sama dengan bentuk rumus *Bayesian rule* dimana hubungan probabilitas bersyarat $p(x|y)$ atau $p(y|x)$ dengan syarat $p(y) = 0$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{x'} p(y|x')p(x')} \quad (\text{diskrit})$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x')p(x')dx'} \quad (\text{kontinyu}) \quad (2.11)$$

Notasi selanjutnya yaitu membahas tentang data pengukuran dan kontrol. Robot menggunakan hasil pengukuran data sensor untuk membentuk persepsi terhadap lingkungannya. Sebagai contoh data dari sensor jarak atau kamera adalah merupakan data yang menunjukkan kondisi lingkungan yang dilihat oleh robot. Dalam hal ini berlaku pengamatan lingkungan dari sensor robot berlaku selama t maka ini disimbolkan dalam z_t dan biasanya ditulis dalam bentuk:

$$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \dots, z_{t_2} \quad (2.12)$$

Persamaan diatas menunjukkan semua pengukuran dari waktu t_1 sampai waktu t_2 , dengan syarat $t_1 \leq t_2$. Kemudian tentang kontrol, kontrol ini digunakan untuk perpindahan robot terhadap lingkungan yang sudah diamati ke lingkungan yang baru. Kontrol didasarkan pada data-data odometri yang menunjukkan perpindahan robot. Kontrol dinotasikan dengan u_t dengan fungsi waktu selama t_1 dan t_2 . Dan ditulis dengan bentuk:

$$u_{t_1:t_2} = u_{t_1+1}, u_{t_1+2}, \dots, u_{t_2} \quad (2.13)$$

Hal penting yang lain dalam *probabilistic robotics* adalah *belief* (keyakinan). *Belief* adalah pengetahuan *internal* robot dalam memahami lingkungan dilalui / diamati. Sebagai contoh robot *pose* (x, y, θ) adalah di koordinat $(5, 5, 45^\circ)$. Pengukuran ini adalah pengukuran yang bersifat estimasi (perkiraan) atau relatif, bukan merupakan pengukuran langsung yang berasal dari sensor (misalnya GPS). Oleh karena itu hasil robot *pose* x, y, θ adalah bentuk keyakinan robot terhadap dirinya sendiri. Dalam *probabilistic robotics*, *belief* dinyatakan dengan bentuk

$$bel(x_t) = p(x_t | z_{1:t}, u_{t:1}) \quad (2.14)$$

Dari persamaan diatas dapat diartikan sebagai *posterior* $bel(x_t)$ (keyakinan setelah pengukuran) dari sebuah *pose* awal x_t dengan waktu t terhadap semua pengukuran $z_{1:t}$ menggunakan kontrol pergerakan $u_{1:t}$. Kita dapat juga mengartikan asumsi *belief* yang dilakukan setelah pengukuran z_t dengan kondisi sebelum digabungkan dengan parameter kontrol u_t maka *posterior* dinotasikan dengan bentuk:

$$\overline{bel}(x_t) = p(x_t | z_{1:t}, u_{1:t}) \quad (2.15)$$

Probabilitas ini adalah sebagai prediksi. Proses perhitungan $bel(x_t)$ dari $\overline{bel}(x_t)$ adalah sebagai koreksi (*correction*) atau perbaruan pengukuran (*measurement update*).

2.1.1 Teorema Bayes

Algoritma *bayes* pada *probabilistic robotics* [11] adalah menghitung distribusi *belief* dari pengukuran. Secara umum langkahnya ditulis dengan

```

1:  Algoritma_Bayes_Filter ( $bel(x_{t-1}), u_t, z_t$ ):
2:  forall  $x_t$  do
3:   $\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx$ 
4:   $bel(x_t) = \eta p(z_t | x_t) \overline{bel}(x_t)$ 
5:  endfor
6:  return  $bel(x_t)$ 

```

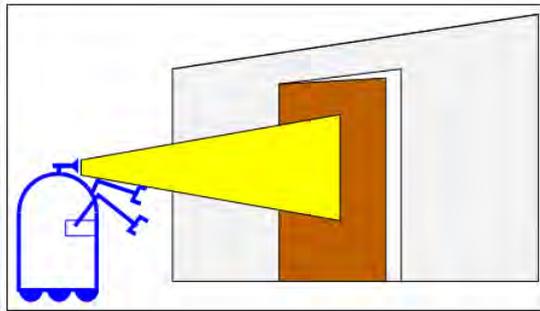
(2.16)

Penjelasan *pseudo-code* dari algoritma diatas adalah keyakinan $bel(x_t)$ pada waktu t dihitung dari keyakinan sebelumnya $bel(x_{t-1})$ pada saat waktu ke $t - 1$. Inputnya adalah *bel* terhadap kontrol u_t yang diberikan dan terhadap pengukuran z_t . Dan untuk outputnya adalah *belief* $bel(x_t)$ pada saat waktu t .

Selanjutnya pada *line* 3 dijelaskan bahwa $\overline{bel}(x_t)$ adalah didapatkan dari total probabilitas *pose* x_t dengan syarat kontrol u_t diterapkan untuk *pose* sebelumnya x_{t-1} dan dikalikan dengan $bel(x_{t-1})$. Dapat disimpulkan dari sini terjadi *step prediction*. Yaitu prediksi $\overline{bel}(x_t)$ yang didapatkan dari $bel(x_{t-1})$. Jika melihat faktor t maka *belief* dengan waktu sekarang didapatkan nilainya melalui $t - 1$.

Kemudian jika dilanjutkan ke *line* 4, disinilah dilakukan *step update* atau dapat disebut *measurement update* dari $bel(x_t)$ dari pengukuran z_t terhadap *pose* robot sekarang x_t dengan dikalikan dengan $\overline{bel}(x_t)$ yaitu *belief* sebelumnya yang sudah

diobservasi. Dan hasil dari langkah ini adalah berbentuk perataan (normalisasi) yang bergantung pada *normalizer* η yaitu normalisasi probabilitas *pose* terhadap pengukuran dengan waktu sekarang. Sebagai ilustrasi sebuah *teorema bayes* maka berikut ini dicontohkan sebuah permasalahan robot pada saat menelusuri lingkungan:



Gambar 2.1 Robot saat mendeteksi kondisi pintu

Sebagai contoh sebuah *mobile robot* mendeteksi sebuah pintu. Untuk mempermudah digunakan kondisi pintu yaitu tertutup dan terbuka. Berlaku *prior probability* untuk dua kondisi pintu tersebut yaitu:

$$\begin{aligned} \text{bel}(X_0 = \text{pintu_terbuka}) &= 0,5 \\ \text{bel}(X_0 = \text{pintu_tertutup}) &= 0,5 \end{aligned} \quad (2.17)$$

Kemudian kita asumsikan *noise sensor* dalam mendeteksi pintu yaitu:

$$\begin{aligned} p(Z_t = \text{sensor_terbuka} | X_t = \text{pintu_terbuka}) &= 0.6 \\ p(Z_t = \text{sensor_tertutup} | X_t = \text{pintu_terbuka}) &= 0.4 \end{aligned} \quad (2.18)$$

dan

$$\begin{aligned} p(Z_t = \text{sensor_terbuka} | X_t = \text{pintu_tertutup}) &= 0.2 \\ p(Z_t = \text{sensor_tertutup} | X_t = \text{pintu_tertutup}) &= 0.8 \end{aligned} \quad (2.19)$$

Kemudian sekarang kita asumsikan robot mendorong pintu. Dengan kemungkinan terbukanya pintu setelah didorong atau robot mendorong pintu saat pintu terbuka maka berlaku:

$$\begin{aligned} p(X_t = \text{pintu_terbuka} | U_t = \text{dorong}, X_{t-1} = \text{pintu_terbuka}) &= 1 \\ p(X_t = \text{pintu_tertutup} | U_t = \text{dorong}, X_{t-1} = \text{pintu_terbuka}) &= 0 \end{aligned} \quad (2.20)$$

dan

$$\begin{aligned}
 p(X_t = \text{pintu_terbuka} | U_t = \text{dorong}, X_{t-1} = \text{pintu_tertutup}) &= 0.8 \\
 p(X_t = \text{pintu_tertutup} | U_t = \text{dorong}, X_{t-1} = \text{pintu_tertutup}) &= 0.2 \quad (2.21)
 \end{aligned}$$

Kemudian kemungkinan lainnya yang masih bisa terjadi adalah:

$$\begin{aligned}
 p(X_t = \text{pintu_terbuka} | U_t = \text{diam}, X_{t-1} = \text{pintu_terbuka}) &= 1 \\
 p(X_t = \text{pintu_tertutup} | U_t = \text{diam}, X_{t-1} = \text{pintu_terbuka}) &= 0 \quad (2.22)
 \end{aligned}$$

dan

$$\begin{aligned}
 p(X_t = \text{pintu_terbuka} | U_t = \text{diam}, X_{t-1} = \text{pintu_tertutup}) &= 0 \\
 p(X_t = \text{pintu_tertutup} | U_t = \text{diam}, X_{t-1} = \text{pintu_tertutup}) &= 1 \quad (2.23)
 \end{aligned}$$

Sehubungan dengan $\overline{bel}(x_t)$ yang sebelumnya sudah dibahas dalam *teorema bayes* maka dapat digambarkan pada saat waktu t , robot diam dan melakukan *sensing* terhadap pintu yang terbuka maka tingkat keyakinan $bel(X_0)$ dan $U_1 = \text{diam}$ dengan $X_0 = \text{sensor_terbuka}$ sebagai input berlaku:

$$\begin{aligned}
 \overline{bel}(X_1) &= \int p(x_1 | u_1, x_0) bel(x_0) dx_0 \\
 &= \sum_{x_0} p(x_1 | u_1, x_0) bel(x_0) \quad (2.24) \\
 &= p(x_1 | U_1 = \text{diam}, X_0 = \text{pintu_terbuka}) bel(X_0 = \text{pintu_terbuka}) + \\
 &\quad p(x_1 | U_1 = \text{diam}, X_0 = \text{pintu_tertutup}) bel(X_0 = \text{pintu_tertutup})
 \end{aligned}$$

Kemudian kita dapat melakukan substitusi dua kemungkinan untuk X_1 . Yang pertama untuk $X_1 = \text{pintu_terbuka}$ maka

$$\begin{aligned}
 \overline{bel}(X_i = \text{pintu_terbuka}) &= p(X_1 = \text{pintu_terbuka} | U_1 = \text{diam}, X_0 \\
 &\quad = \text{pintu_terbuka}) \\
 &\quad * bel(X_0 = \text{pintu_terbuka}) \\
 &\quad * p(X_1 = \text{pintu_terbuka} | U_1 = \text{diam}, X_0 = \text{pintu_tertutup}) \\
 &\quad * bel(X_0 = \text{pintu_tertutup}) \\
 &= 1 \times 0.5 + 0 \times 0.5 = 0.5 \quad (2.25)
 \end{aligned}$$

Dan untuk $X_1 = \text{pintu_tertutup}$ maka

$$\begin{aligned} \overline{bel}(X_1 = \text{pintu_tertutup}) &= p(X_1 = \text{pintu_tertutup} | U_1 = \text{diam}, X_0 \\ &= \text{pintu_terbuka}) \\ &* bel(X_0 = \text{pintu_terbuka}) \\ &+ p(X_1 = \text{pintu_tertutup} | U_1 = \text{diam}, X_0 = \text{pintu_tertutup}) \\ &* bel(X_0 = \text{pintu_tertutup}) \\ &= 1 \times 0,5 + 0 \times 0,5 = 0,5 \end{aligned} \quad (2.26)$$

Jika kita amati $\overline{bel}(X_1)$ sama dengan prior $bel(x_0)$ dengan aksi $U_1 = \text{diam}$ maka hasil yang didapatkan adalah tidak merubah kondisi sebelumnya. Kemudian jika dihubungkan dengan tahapan dalam teorema *Bayesian* yaitu *line 4* maka:

$$bel(x_1) = \eta p(Z_1 = \text{sensor_terbuka} | x_1) \overline{bel}(x_1) \quad (2.27)$$

Untuk dua kemungkinan saat $X_1 = \text{pintu_terbuka}$ dan $X_1 = \text{pintu_tertutup}$ kita mendapatkan:

$$\begin{aligned} bel(X_1 = \text{pintu_terbuka}) &= \eta p(Z_1 = \text{sensor_terbuka} | X_1 = \text{pintu_terbuka}) \\ &* \overline{bel}(X_1 = \text{pintu_terbuka}) \\ &= \eta 0,6 \times 0,5 = \eta 0,3 \end{aligned} \quad (2.28)$$

dan

$$\begin{aligned} bel(X_1 = \text{pintu_tertutup}) &= \eta p(Z_1 = \text{sensor_terbuka} | X_1 = \text{pintu_tertutup}) \\ &* \overline{bel}(X_1 = \text{pintu_tertutup}) \\ &= \eta 0,2 \times 0,5 = \eta 0,1 \end{aligned} \quad (2.29)$$

Untuk itu normalizer η dapat dihitung sebagai berikut:

$$\eta = (0,3 + 0,1)^{-1} = 2,5 \quad (2.30)$$

jadi kita mempunyai

$$\begin{aligned} bel(X_1 = \text{pintu_terbuka}) &= 0,75 \\ bel(X_1 = \text{pintu_tertutup}) &= 0,25 \end{aligned} \quad (2.31)$$

Dan sekarang perhitungan menjadi mudah untuk $u_2 = \text{dorong}$ dan $z_2 = \text{sensor_terbuka}$ yaitu

$$\begin{aligned}\overline{bel}(X_2 = \text{pintu_terbuka}) &= 1 \times 0.75 + 0.8 \times 0.25 = 0.95 \\ \overline{bel}(X_2 = \text{pintu_tertutup}) &= 0 \times 0.75 + 0.2 \times 0.25 = 0.05\end{aligned}\quad (2.32)$$

dan

$$\begin{aligned}bel(X_2 = \text{pintu_terbuka}) &= \eta 0.6 \times 0.95 = 0.983 \\ bel(X_2 = \text{pintu_tertutup}) &= \eta 0.2 \times 0.05 = 0.017\end{aligned}\quad (2.33)$$

Partikel filter menggunakan pendekatan *posterior* dengan waktu yang terbatas. Kunci utama partikel filter adalah menyatakan *posterior* $bel(x_t)$ dengan persebaran acak. Dalam partikel filter, *sample posterior* disebut partikel dan dinotasikan dengan:

$$X_t = x_t^{[1]}, x_t^{[2]}, \dots, x_t^{[M]}\quad (2.34)$$

Setiap partikel $x_t^{[M]}$ (dengan $1 \leq m \leq M$) adalah pada saat kondisi waktu t . kemudian M menotasikan banyaknya partikel dalam partikel set X_t . Misalnya $M=1000$. Dalam beberapa implementasi M adalah fungsi t atau jumlah partikel dari sebuah *belief* $bel(x_t)$

Dalam bayes filter, partikel menggunakan $bel(x_t)$ secara rekursif dari $bel(x_{t-1})$ yaitu 1 step lebih awal. Masukkan dari algoritma ini adalah partikel set X_{t-1} dengan kontrol pergerakan dan pengukuran z_t . Kemudian algoritma membangun partikel dengan hasil yang sama atau ekuivalen dengan $\overline{bel}(x_t)$.

Berikut ini sistematika proses partikel filter $x_{t-1}^{[m]}$ dengan input partikel X_{t-1} .

Line 4 menunjukkan $x_t^{[m]}$ untuk waktu t berdasarkan partikel $x_{t-1}^{[m]}$ dengan kontrol u_t . Dan hasilnya adalah partikel dengan index m , menunjukkan partikel ke- m yang menunjukkan X_{t-1} . Langkah ini mengambil *sampling* dari distribusi selanjutnya $p(x_t|u_t, x_{t-1})$. Set partikel dari iterasi ke M kali adalah $\overline{bel}(x_t)$.

Line 5 menghitung setiap partikel $x_t^{[m]}$ dengan sebutan *importance factor* (faktor penting) yang mendenotasikan $w_t^{[m]}$. faktor penting adalah digunakan

untuk mendapatkan pengukuran z_t dalam partikel. Probabilitas pengukuran z_t untuk partikel $x_t^{[m]}$. Jika kita menganggap $w_t^{[m]}$ sebagai bobot dari sebuah partikel, maka bobot dari setiap partikel menunjukkan pendekatan *posterior* $bel(x_t)$.

Langkah selanjutnya yaitu ditunjukkan dengan *line* 8 sampai dengan 11 dalam persamaan partikel filter. Langkah ini disebut *resampling*. Partikel filter melakukan *update* untuk sejumlah M partikel dari set partikel sementara \bar{X}_t . Partikel digambarkan kembali sesuai dengan bobotnya.

Langkah *resampling* mempunyai peranan penting untuk menyatakan sebuah *posterior* $bel(x_t)$. Partikel dengan probabilitas tinggi akan dipilih kembali atau digunakan kembali untuk melakukan pendekatan *posterior* $bel(x_t)$.

2.2 Robot Motion

Pada bagian ini dijelaskan tentang pengaruh pergerakan robot dalam proses partikel filter. Dua unsur utama dalam partikel filter adalah pergerakan dan pengukuran (*motion* dan *measurement*). Pada pembahasan ini fokus pada model pergerakan robot. Untuk model pergerakan robot maka pembahasan yang akan dilakukan adalah tentang kinematika robot.

Kinematik robot adalah perhitungan teknis tentang bagaimana robot bergerak dengan konfigurasi aktuator yang dipakai oleh robot. 6 variabel kinematik yang biasanya dipakai adalah tiga dimensi cartesian (x, y, z) dan 3 *euler angle* (*pitch, roll, yaw*). Konfigurasi yang dipakai adalah dalam bentuk *pose* yang dituliskan dalam $\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$. Untuk arah (*orientation*) disebut juga *bearing* atau *heading direction*). Untuk (x, y) disebut sebagai lokasi/posisi (*pose* tanpa arah).

Pembahasan selanjutnya adalah mengarah pada probabilitas kinematika robot. Karena dianggap semua unsur yang mempengaruhi yaitu sensor, kontrol, pengukuran, keadaan adalah bersifat tidak pasti, maka semuanya berpeluang untuk mempunyai probabilitas. Dalam probabilitas kinematika, model pergerakan dikenal dalam bentuk:

$$p(x_t | u_t, x_{t-1}) \quad (2.35)$$

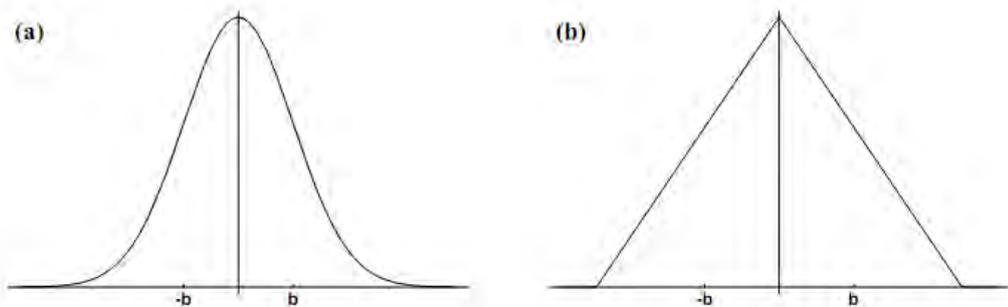
Dengan keterangan x_t dan x_{t-1} adalah robot *pose* dan u_t adalah aksi. Dalam probabilitas kinematika, robot *motion* dibagi menjadi 2 yaitu *velocity model* dan *odometry model*.

2.2.1 Velocity Model

Dalam *velocity* model diasumsikan kita dapat melakukan kontrol robot melalui dua kecepatan, sebuah rotasi dan sebuah translasi. Dengan beberapa tipe kendali yang banyak digunakan (*differential drive*, *synchro drive*, *Ackerman drive*, dan tipe kendali yang lainnya baik *holonomic* atau *non holonomic*). Kecepatan rotational v_t dan kecepatan translational ω_t dituliskan dalam bentuk:

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} \quad (2.36)$$

Dalam keadaan sebenarnya, pergerakan robot tidak dapat dianggap ideal karena pengaruh *noise*. Untuk itu dilakukan pendekatan dengan model distribusi normal. Ada dua model distribusi yaitu distribusi normal dan triangular distribusi.



Gambar 2.2 Distribusi normal (kiri) dan triangular distribusi (kanan)

Distribusi normal dengan *error* rata-rata nol dan *variance* b ditulis dalam bentuk

$$\varepsilon_b(a) = \frac{1}{\sqrt{2\pi \cdot b}} e^{-\frac{1a^2}{2b}} \quad (2.37)$$

Untuk distribusi triangulasi ditulis dalam bentuk

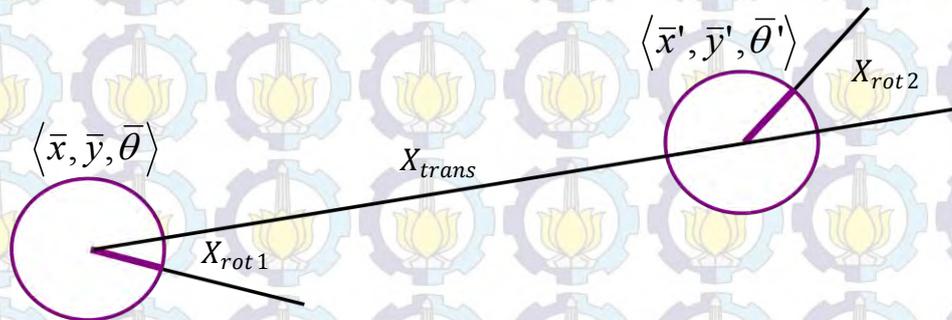
$$\varepsilon_b(a) = \begin{cases} 0 \\ \frac{\sqrt{6b-|a|}}{6b} \end{cases} \quad (2.38)$$

Bentuk akhir dari robot *motion* yang sesuai dengan kondisi robot sebenarnya yaitu:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega} \sin\theta + \frac{v}{\omega} \sin(\theta + \omega\Delta t) \\ \frac{v}{\omega} \cos\theta - \frac{v}{\omega} \cos(\theta + \omega\Delta t) \\ \omega\Delta t + \gamma\Delta t \end{pmatrix} \quad (2.39)$$

2.2.2 Odometry Model

Untuk penggunaan model odometry maka digunakan:



Gambar 2.3 Sistem Odometri robot

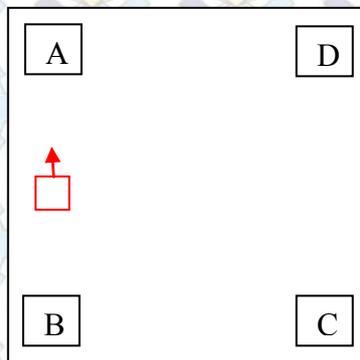
$$\begin{aligned} X_{trans} &= \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \\ X_{rot1} &= \text{atan}^2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \\ X_{rot2} &= \bar{\theta}' - \bar{\theta} - X_{rot1} \end{aligned} \quad (2.40)$$

2.3 Penentuan Posisi

Masalah penentuan posisi adalah menentukan letak dan orientasi (*pose*) dari robot di sebuah lingkungan. Penentuan lokasi atau posisi disebut juga *localization*. Dalam *localization* terbagi menjadi tiga jenis permasalahan yaitu *local*, *global*, dan *relokalisasi*. Lokalisasi lokal diterapkan pada kondisi ketika robot mengetahui posisi dan orientasi awal. Metode ini dapat mengestimasi posisi robot dengan menggunakan hasil pembacaan sensor robot secara terus menerus meskipun terdapat kesalahan odometri dan kesalahan pada observasi sensor. Sementara itu lokalisasi global diterapkan ketika robot tidak mengetahui posisi dan orientasi awalnya sehingga robot harus melakukan observasi pada lingkungannya agar dapat menentukan *pose*. Kondisi relokalisasi merupakan salah

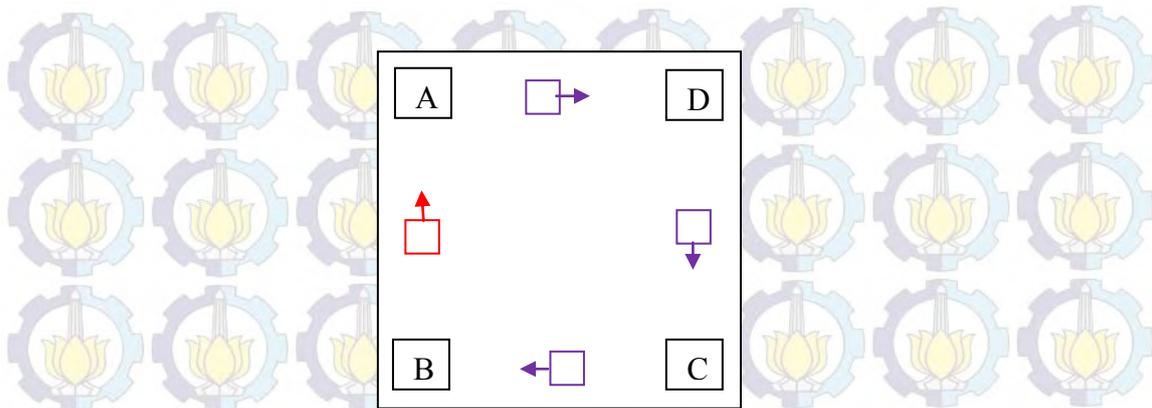
satu permasalahan yang lebih kompleks, dimana kondisi ini terjadi ketika robot menyadari bahwa hasil observasi lingkungan robot berbeda dengan hasil pengukuran yang dilakukan pada *sampel* robot dalam peta. Dengan kata lain, posisi robot dipindahkan tanpa ada informasi sebelumnya (*kidnapping*).

Contoh yang paling sederhana adalah mengenai *position-tracking* yaitu robot mengestimasi posisi setelah melakukan perintah gerakan dengan informasi posisi awal robot diketahui. Kemudian masalah yang lebih rumit adalah ketika robot tidak mengetahui posisi awal, robot harus melakukan prediksi posisi karena robot tidak diberi informasi apapun termasuk informasi titik awal robot. Sebagai contoh robot diletakkan pada lingkungan yang berbentuk kotak dengan titik awal berada diantara titik A dan B, robot menghadap ke titik A.



Gambar 2.4 Posisi awal robot

Dalam keadaan ini, jika robot mempunyai pemahaman bahwa robot berada di B, C, atau D (warna ungu) maka pemahaman tersebut tidak bisa dianggap salah. Meskipun secara posisi global posisi robot berada di B. Tetapi dari sisi lokal robot, titik B, C, D mempunyai kemungkinan yang sama untuk diketahui sebagai titik koordinat robot karena sama-sama mempunyai jarak terhadap sisi tepi (gambar 2.4).



Gambar 2.5 Kemungkinan Letak Robot

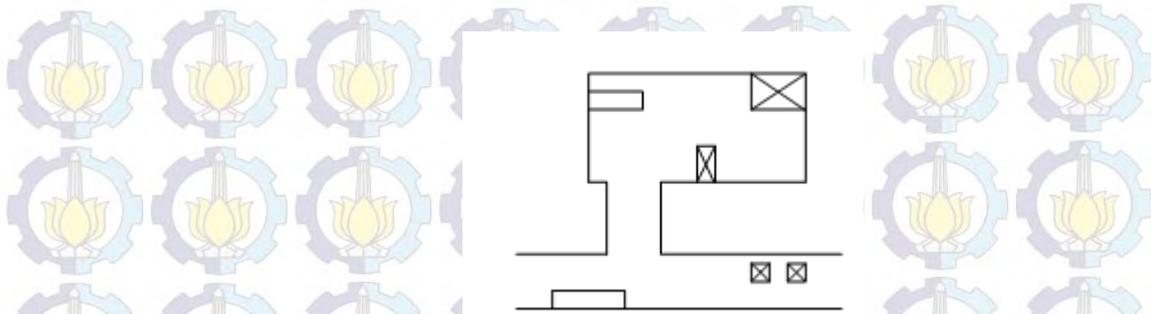
Masalah yang lebih kompleks adalah masalah *kidnap* yaitu kondisi robot yang dipindahkan secara tiba-tiba tanpa pemberitahuan terlebih dahulu, biasanya dilakukan dalam kompetisi. Ketika robot sedang *error* maka robot dipindahkan ke lokasi baru. Secara pemahaman robot, lokasi yang baru merupakan kelanjutan dari lokasi sebelumnya. Ini menjadi masalah karena dalam penempatan lokasi baru, robot harus menganggap bahwa itu adalah posisi baru / awal. Persepsi yang timbul atas pemahaman awal ini mengakibatkan semua pergerakan robot menjadi salah karena hasil pengukuran yang dihasilkan adalah terhadap posisi lanjutan dari titik akhir sebelum robot di *kidnap*. Untuk itu perlu adanya pengkondisian mengenai pemahaman (*belief*) robot pada saat terjadi *kidnap*.

2.4 Representasi Hasil *Mapping*

Beberapa bentuk peta yang digunakan untuk merepresentasikan hasil *mapping* yaitu:

1. *Geometric Feature Map* adalah merujuk pada robot untuk mengumpulkan informasi tentang persepsi (gambaran awal) lingkungan menggunakan garis dan kurva. *Geometric map* terdiri dari beberapa karakteristik yang berisi informasi lokasi untuk lingkungan yang sebenarnya dan posisi robot.

Keuntungan *geometric map* adalah memiliki ketelitian yang tinggi.



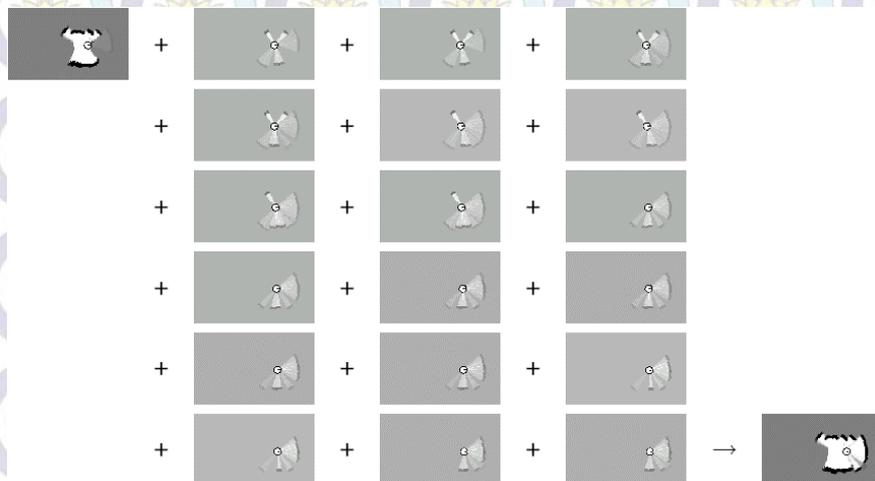
Gambar 2.6 Contoh *Geometric Feature Map* [12]

2. *Grid map* merepresentasikan lingkungan secara diskrit. Metode ini mudah diterapkan oleh komputer tetapi ketika skalanya besar maka perhitungan *grid* menjadi banyak membutuhkan proses komputasi yang kompleks dan tidak menjadi *realtime*.



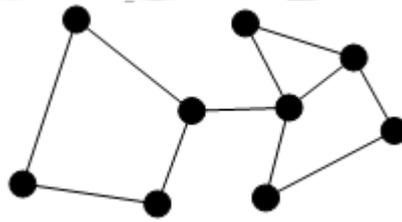
Gambar 2.7 Contoh *Grid Map* [12]

Contoh lain peta *grid* digunakan dalam *occupancy grid mapping* ditunjukkan pada gambar 2.7



Gambar 2.8 Proses update pada *Occupancy Grid* [13]

3. Selanjutnya yang digunakan adalah *topological map*. *Topological map* cocok untuk lingkungan yang terstruktur tapi tidak cocok untuk lingkungan yang tidak terstruktur. Penggunaan *topological map* didasarkan pada informasi dari setiap *node* yang didapat. Kelemahannya yaitu jika ada dua tempat yang sangat mirip maka akan sulit untuk *topological map* menentukan apakah keduanya merupakan titik yang sama atau berbeda.



Gambar 2.9 *Topological Map* [12]

2.5 Algoritma Partikel Filter

Dalam metode *particle filter*, posisi robot ditentukan oleh tiga parameter (x, y, θ) . Setiap *sampel* x, y, θ bersama dengan *weight factor* w_i , mendenotasikan lokasi hipotetik dimana robot sebenarnya berada dengan probabilitas $w_i, i = 1 \dots N$. N disebut sebagai ukuran *sampel set*. *Sampel* dan *weight factor* diperbarui secara rekursif dengan empat langkah berikut:

1. Evaluasi: untuk setiap *sampel*, probabilitas dimana robot sebenarnya berada dievaluasi secara proporsional berdasarkan perbedaan antara data yang diamati dari kamera dengan data yang diharapkan dari lokasi hipotetik menggunakan model sensor. Lalu, *weight factor* ditentukan berdasarkan normalisasi probabilitas.
2. Estimasi: *pose* dari robot dihitung dari sampel distribusi:

$$\begin{bmatrix} x'_r \\ y'_r \\ \theta'_r \end{bmatrix} = \begin{bmatrix} \sum_i^N w_i x_i \\ \sum_i^N w_i y_i \\ \tan^{-1} 2(\sum_i^N \sin \theta_i, \sum_i^N \cos \theta_i) \end{bmatrix} \quad (2.41)$$

3. Resample: sampel disebar ulang berdasarkan masing-masing *weight factor* untuk menghilangkan sampel yang tidak berprobabilitas. Lalu sampel akan disebar secara lokal berdasarkan persamaan:

$$\begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} + \begin{bmatrix} \Delta_{sx} (1 - w_i) g_{xi} \\ \Delta_{sy} (1 - w_i) g_{yi} \\ \Delta_{s\theta} (1 - w_i) g_{\theta i} \end{bmatrix} \quad (2.42)$$

Δ_{sx}, Δ_{sy} , dan $\Delta_{s\theta}$ adalah perbedaan langkah dalam jarak dan arah, dan g_{xi}, g_{yi} , dan $g_{\theta i}$ adalah angka Gaussian acak dalam nilai antara $[-1, 1]$.

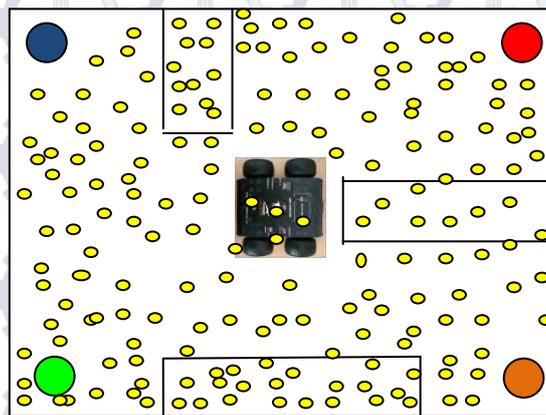
4. Prediksi: posisi selanjutnya dari setiap sampel didapatkan dengan mengaplikasikan model probabilistik pergerakan odometri.

$$\begin{bmatrix} x'_i \\ y'_i \\ \theta'_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} + \begin{bmatrix} \Delta_x \\ \Delta_y \\ \Delta_\theta \end{bmatrix} + \begin{bmatrix} g_{xi} \\ g_{yi} \\ g_{\theta i} \end{bmatrix} \quad (2.43)$$

dengan Δ_x, Δ_y , dan Δ_θ adalah *offsets* dari *pose* robot dalam koordinat lapangan yang didapatkan dari data odometri.

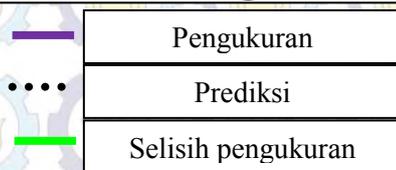
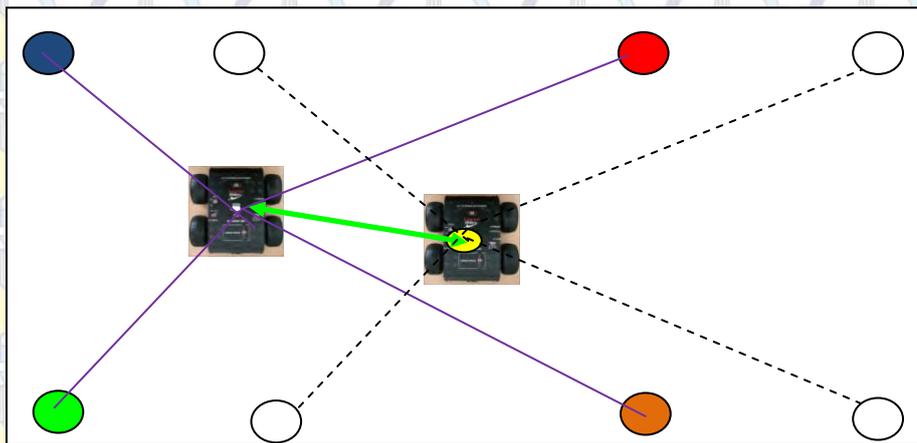
Untuk tahapan proses *particle filter* sebagai berikut:

1. Pada awalnya sejumlah partikel diinisialisasi secara acak (*random*) pada lapangan.



Gambar 2.10 Penyebaran partikel secara acak

2. Kemudian robot mendeteksi sebuah penanda, bobot (*weight*) dari setiap partikel dihitung berdasarkan perbandingan jarak antara robot dengan penanda yang diperoleh dari kamera dan jarak partikel dengan penanda. Dengan kata lain bobot diartikan sebagai perbedaan pengukuran sebenarnya (*actual*) dengan prediksi pengukuran (*predict*) oleh sebuah partikel.



Gambar 2.11 Penentuan *weight*

- Selanjutnya dilakukan proses *resampling* yaitu pemilihan partikel-partikel secara acak berdasarkan bobot setiap partikel sehingga didapat komposisi partikel baru yang konvergen pada posisi mendekati posisi robot sebenarnya. Adapun *pseudocode* untuk *resampling* adalah sebagai berikut:

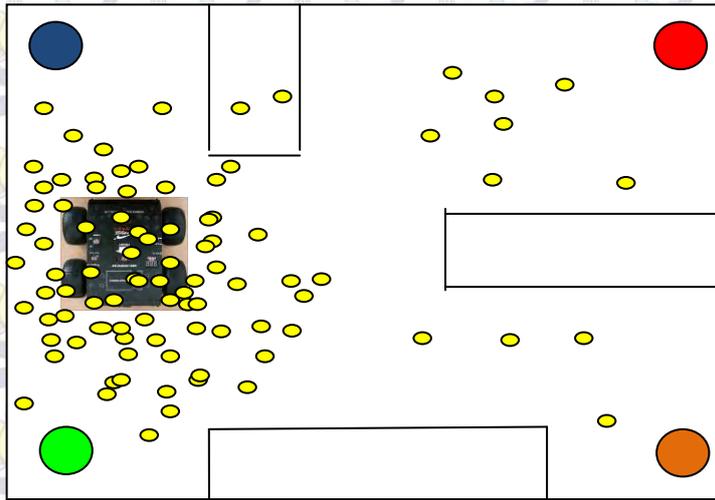
```

Masukan: partikel[n], bobot[n], hasil[n]
index = rand(0, 1) * n
beta = 0.0
for ( i = 0; i < n; i++ )
    beta += rand(0, 1) * 2.0 *
max(bobot)
while ( beta > bobot[index] ) do
    beta -= bobot[index]
    index = (index + 1) % n
    hasil[i] = partikel[index]
return(hasil)

```

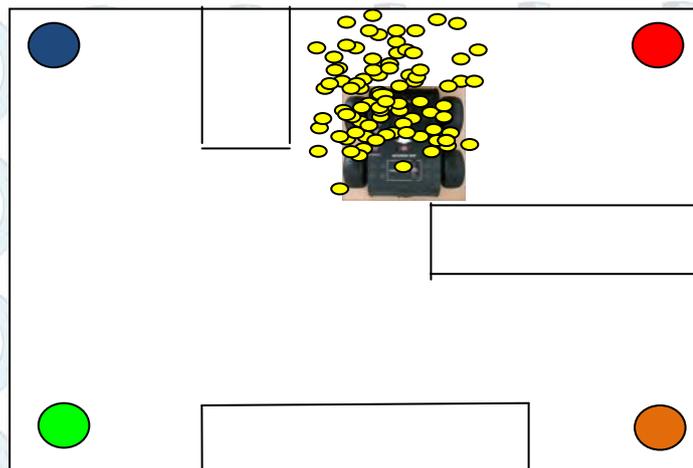
Gambar 2.12 *Pseudocode resampling process*

- Setelah itu posisi setiap partikel akan diperbarui berdasarkan informasi odometri robot terakhir. Partikel dengan bobot terbesar akan berada disekitar robot.



Gambar 2.13 Hasil *Resampling*.

5. Kemudian proses *update* dan *resampling* akan dilakukan berulang-ulang bersamaan dengan *update* pergerakan dan data *update* dari sensor robot. Partikel yang mempunyai bobot yang paling banyak dan besar akan menjadi solusi terbaik dan memiliki *pose* (x, y, θ) yang mewakili posisi robot sebenarnya.



Gambar 2.14 Konvergen partikel.

2.6 Algoritma Dinamik A-Star

Perencanaan jalur mempunyai peranan yang penting untuk proses navigasi robot. Navigasi diartikan sebagai proses atau aktivitas untuk merencanakan atau menuju jalur secara langsung dalam sebuah misi yang diberikan pada sebuah *autonomous mobile robot* dari satu tempat ke tempat yang

lain tanpa kehilangan arah atau mengalami tabrakan dengan *object* yang lain [21]. Dalam perencanaan jalur yang ideal diharapkan dapat memberikan solusi ketidakyakinan (*uncertainties*) robot dalam menemukan jalur optimum dengan waktu yang singkat serta terhindar dari penghalang atau tabrakan.

Penelitian sebelumnya tentang perencanaan jalur diantaranya adalah menggunakan algoritma A*. Standar pencarian jalur terdekat menggunakan algoritma A* adalah menggunakan persamaan:

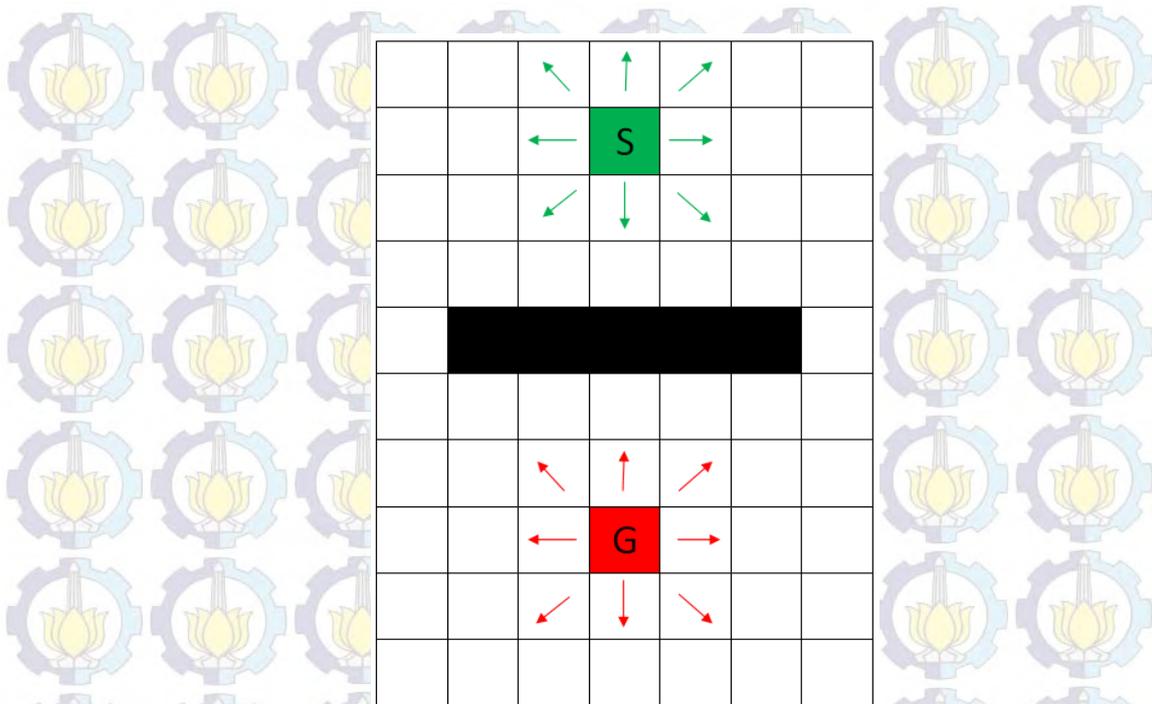
$$f(n) = g(n) + h(n) \quad (2.44)$$

Pada persamaan yang dipakai untuk algoritma A* tersebut, digunakan $g(n)$ untuk mewakili harga (*cost*) rute dari titik awal ke *node n*, lalu $h(n)$ mewakili perkiraan harga dari *node* awal ke *node goal*, yang dihitung dengan fungsi *heuristic*. A* menjumlahkan kedua nilai ini dalam mencari jalur dari *node* awal ke *node goal*. Algoritma ini akan memilih *node* dengan nilai $f(n)$ yang paling kecil.

Pada sistem ini diimplementasikan A* dengan perulangan perhitungan harga yang dilakukan setiap kali robot mendeteksi adanya penghalang atau *obstacle* dalam mencapai *goal*. Algoritma A* mengkombinasikan data yang diperoleh dari *global camera* berupa data koordinat dan arah dengan data yang diperoleh dari pembacaan sensor yang ada di robot (*local*). Sensor meliputi *rotary encoder* yang digunakan untuk menghitung jarak tempuh robot menggunakan hubungan pulsa yang dihasilkan dan hubungan radius roda dari robot. Sensor ultrasonik digunakan untuk sistem penghindar halangan dan sebagai informasi dari sisi lokal robot yang memberitahukan kepada sistem untuk mencari jalan terpendek yang lain dengan mempertimbangkan harga minimal dari penjumlahan $g(n)$ dan $h(n)$.

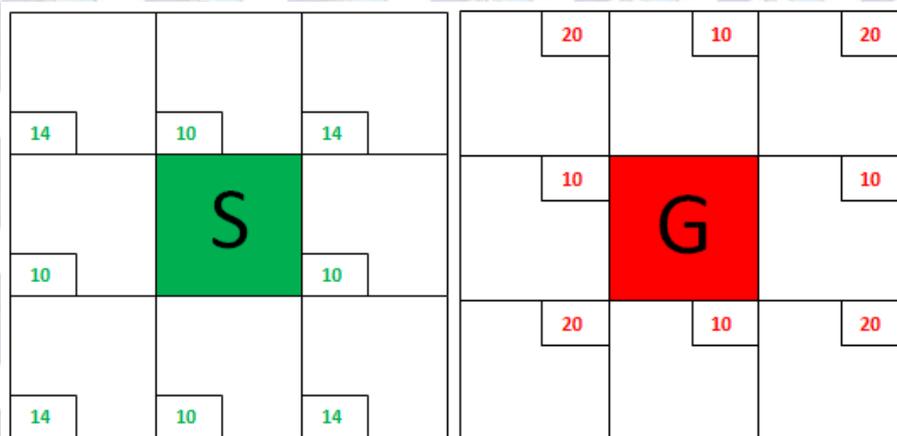
Adapun tahapan untuk menggunakan A* dalam sistem ini adalah sebagai berikut:

1. Sebagai contoh lintasan atau area yang akan dilewati adalah berawal dari start (kotak warna hijau) dan titik tujuan akhir (kotak warna merah). Kemudian terdapat penghalang (*obstacle*) dengan warna hitam. Ditunjukkan pada gambar 2.14.



Gambar 2.15 Posisi titik *start* dan titik *goal*.

- Langkah awal adalah memberikan nilai untuk $g(n)$ dan $h(n)$. Nilai $g(n)$ adalah nilai untuk setiap arah yang mempunyai kemungkinan untuk menuju ke *goal*. Nilai $h(n)$ adalah nilai *heuristic* yang menunjukkan jarak dari *start* ke *goal* tanpa melihat adanya penghalang. Sebagai contoh pada gambar 2.15 digunakan nilai $h(n)$ sebesar 10 untuk *grid* dengan arah *horizontal* atau *vertical*. Dan untuk *diagonal* diberi nilai 14. Kemudian untuk $h(n)$ diberikan nilai sebesar 10 untuk setiap *grid* yang mengandung jarak dari titik *goal* ke titik *start*.



Gambar 2.16 Pemberian nilai $g(n)$ dan $h(n)$ untuk setiap arah pada *start* dan *goal*

3. Untuk mencari jarak terdekat maka digunakan harga $f(n)$ yang paling kecil. Nilai $f(n)$ adalah hasil penjumlahan $g(n)$ dan $h(n)$. Nilai $h(n)$ adalah harga dari *start* ke *goal* dengan mempertimbangkan penghalang yang ada. Nilai $g(n)$ ditunjukkan dengan warna hijau. Selanjutnya yaitu $h(n)$ adalah nilai *heuristic* dari titik *goal* ke *start* dengan memberi nilai pada setiap kotak (termasuk penghalang). Nilai $h(n)$ ditunjukkan dengan warna merah. Untuk $f(n)$ adalah penjumlahan dari $g(n)$ dan $h(n)$ ditunjukkan dengan warna ungu. Untuk langkah awal yaitu menghitung nilai untuk setiap *grid* yang berada di sekitar *start*. Terdapat beberapa nilai yaitu 80, 94, 60 dan 74. Dari nilai-nilai ini dipilih nilai $h(n)$ yang paling kecil yaitu 60.

94	80	80	70	94	80
14		10		14	
80	70	S	80	70	
10				10	
74	60	60	50	74	60
14		10		14	

Gambar 2.17 Nilai $f(n)$ untuk grid sekitar titik start

4. Dengan dipilihnya *grid* dengan nilai terkecil pada langkah 3 yaitu 60 maka kotak di sekitar *start* awal akan dianggap sebagai *close*. Dan kotak 60 dianggap sebagai *open* yaitu perhitungan *grid* selanjutnya adalah terhadap titik baru yang terpilih ini. Setiap terpilih nilai paling kecil diantara *grid* yang berada di sekitar kotak maka terjadi perulangan perhitungan dengan menjadikan setiap *grid* yang terkecil nilainya menjadi *grid open* dan *grid* yang berada disekitar $f(n)$ yang baru akan menjadi *close*. Secara keseluruhan perhitungan untuk $f(n)$, $g(n)$ dan $h(n)$ dari titik awal (*start*) ke titik akhir (*goal*) ditunjukkan pada gambar 2.17. Jalur terdekat ditunjukkan dengan warna biru untuk sederetan nilai $f(n)$ yang kecil yaitu jalur terdekat dalam menuju ke titik *goal*.

setiap m_i berisi nilai biner yang menyatakan *grid* tersebut sudah di *occupied* (ditinjau) atau belum ditinjau (*free*). Ditulis nilai “1” untuk *grid* yang *occupied* dan “0” untuk nilai *grid* yang bebas. Jika untuk probabilitas sebuah *grid* yang sudah di *occupied* atau belum dituliskan dengan notasi $p(m_i = 1)$ atau $p(m_i)$.

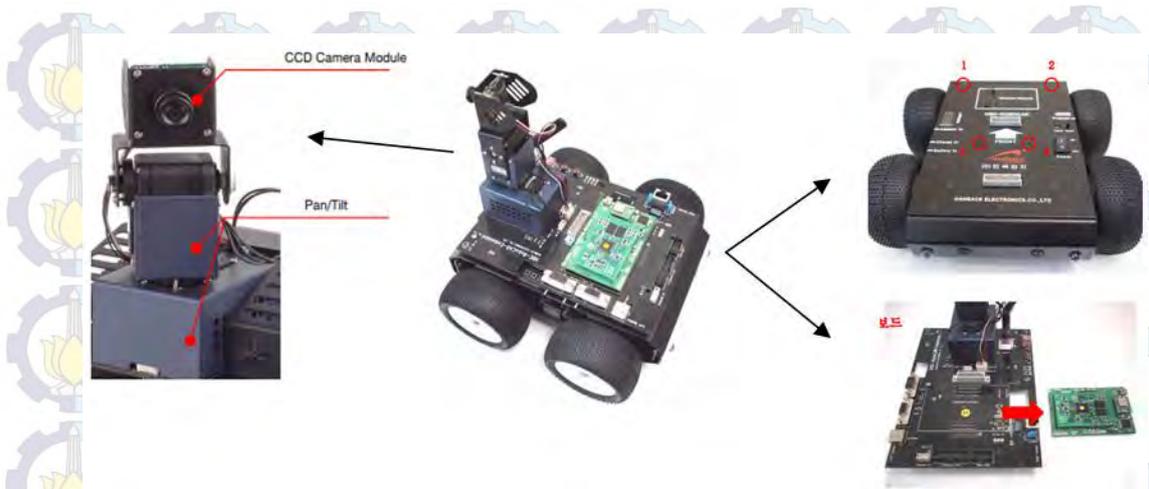
2.8 HBE Robocar Embedded

Robot yang digunakan untuk mengimplementasikan SLAM ini adalah HBE Robocar yang mempunyai beberapa perangkat *hardware* dan pendukung untuk keperluan SLAM seperti sensor jarak (ultrasonik), rotary encoder, kamera untuk pengenalan penanda. Berikut adalah bentuk fisik robot HBE-Robocar Embedded



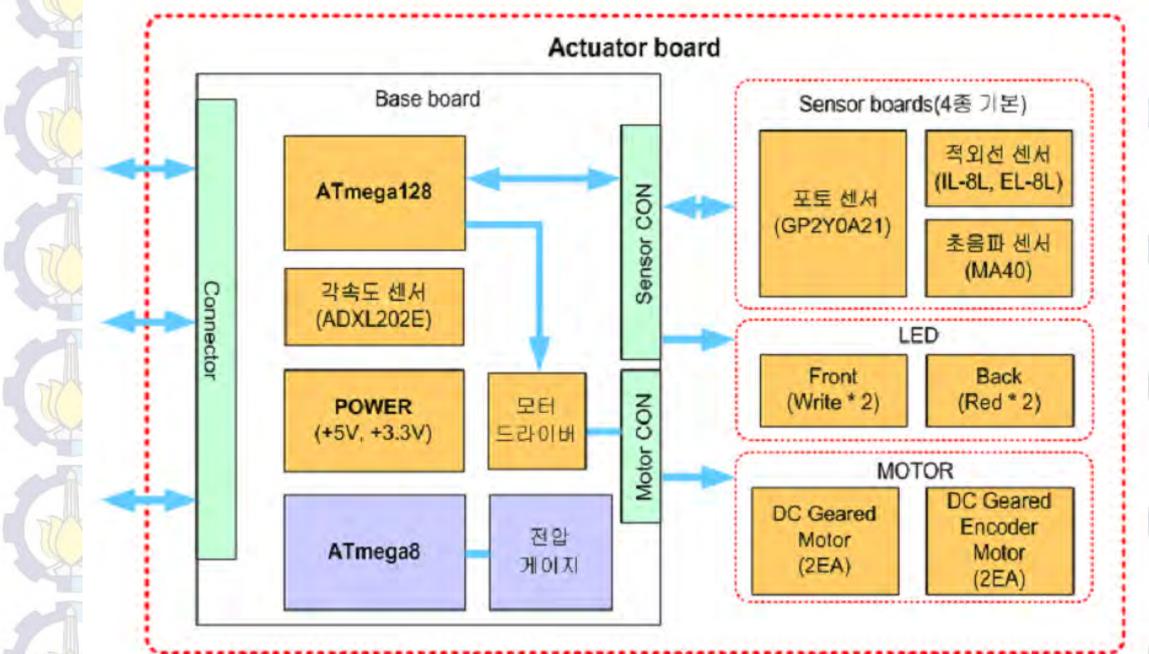
Gambar 2.19 HBE Robocar Embedded [14]

Pada HBE-robocar Embedded adalah terdiri dari 2 bagian yaitu *mobile robot actuator* dan *module embedded camera* yang merupakan *expansion module* yang ditancapkan diatas HBE robocar. Pada *Robocar Embedded* terdapat *camera CCD* dengan penggerak berupa *motor servo* dengan gerakan *pan / tilt*.



Gambar 2.20 HBE Robocar dan *Embedded Camera* [14]

Fitur yang dimiliki oleh HBE-Robocar diantaranya ATmega 128, *sensor board*, motor, *encoder*, LED dan beberapa modul yang lain. Adapun diagram blok sistem dari HBE Robocar:



Gambar 2.21 Bagan sistem HBE Robocar [14]

Pada HBE-Robocar terdapat beberapa blok spesifikasi sistem meliputi *processor*, sensor dan *actuator* yang terintegrasi didalam badan robot.



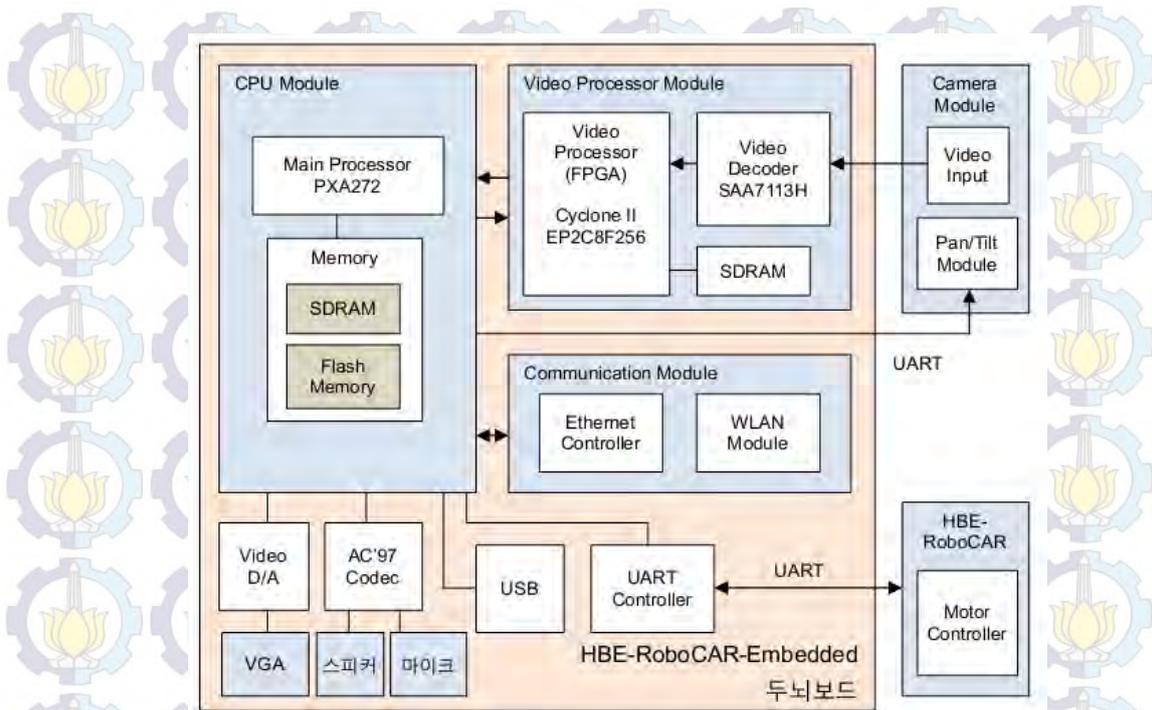
Gambar 2.22 *Body* HBE Robocar [14] dan *board* di dalamnya

Untuk informasi sensor dan actuator dalam HBE Robocar, detail spesifikasi sebagai berikut:

Tabel 2.1 Detail sistem perangkat pada HBE-Robocar [14]

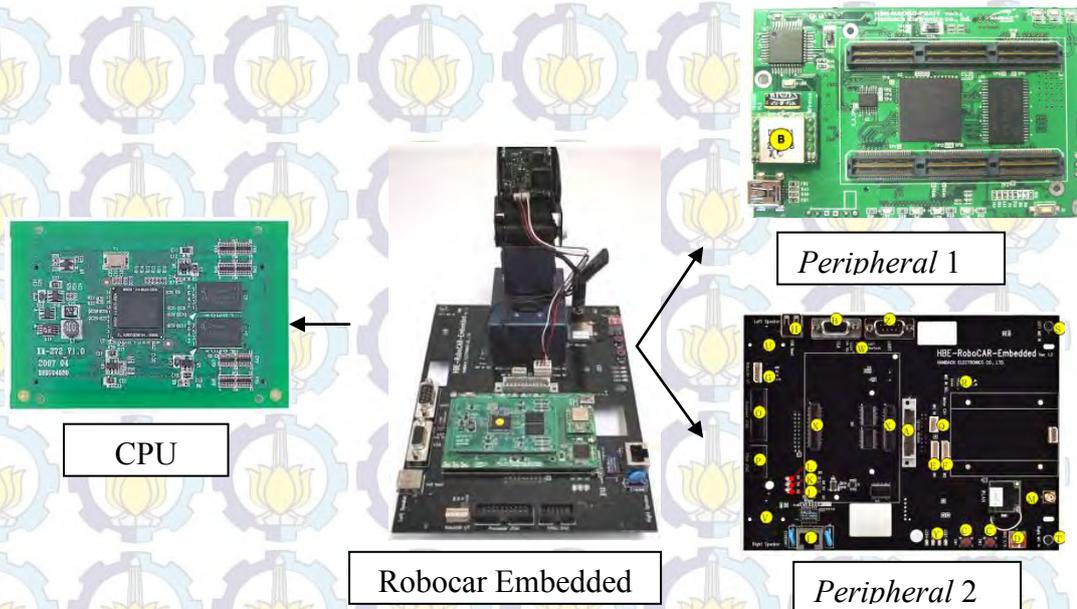
구분	사양
ATmega128L	8-bit AVR, Microcontroller with 128K Bytes, main control
ATmega8L	8-bit AVR, Microcontroller with 8K Bytes, Voltmeter control
L298P	Up to 4A DC Motor Driver 2EA
Ultrasonic sensor	40.0 ± 0.5KHz Frequency, 2.0KHz Bandwidth, 2EA
Accelerometer sensor	Dual-Axis Accelerometer sensor 1EA, Duty Cycle
PSD sensor	Distance Measuring Sensors 1EA, 10-80cm
Phototransistors	8-groups Infrared rays sensor
Motors	DC geared motor 2EA, DC geared encoder motor 2EA
Buzzer	5V input Buzzer 1EA
LED	10mm high brightness LED, White 2EA, RED 2EA
7-Segment	Voltmeter Display, 3-Digit 1EA
Regulator	+11.1V DC Input, DC output : +5V, +3.3V
Battery	+11.1V, 5200mA Lithium Ion 1EA
Charge & Adapter	+12.6V 1.2A Battery Charger 1EA

Selanjutnya adalah bagan sistem untuk HBE-Robocar Embedded adalah ditunjukkan pada gambar 2.23:



Gambar 2.23 Bagan sistem Robocar Embedded [14].

Dalam HBE Robocar Embedded terdapat tiga buah modul yaitu CPU, *peripheral1*, dan *peripheral 2*.



Gambar 2.24 Beberapa *peripheral* dalam Robocar Embedded [14].

Adapun Spesifikasi *peripheral 1*, *peripheral 2* dan CPU ditunjukkan pada tabel 2.2, tabel 2.3, dan tabel 2.4

Tabel 2.2 Spesifikasi CPU [14].

■ CPU 모듈

Items	Hardware Specifications
CPU	Intel XScale PXA272 MCP, 520MHz
Memory	Flash Memory : 64MByte SDRAM : 64MByte
Connector	192pin * 2ea (Samtec, QSH-060-010-F-D-A)
PCB Size	70 x 45 x 1.0 mm (L x W x H)

Tabel 2.3 Spesifikasi *Peripheral 1* [14].

■ Peri-1 모듈

Items	Hardware Specifications
FPGA	Altera Cyclone-II EP2C8F256C8, LE : 8,256
Config Device	Altera EPCS4SI8N, 4Mbit
SRAM	Samsung K6R4016V1C, 256Kx16 Bit
Video Decoder	Philips SAA7113H
UART	Bluetooth UART 1ea : UART0 TTL Level UART 1ea : UART1
I/O Connector	3p Header 1 port: UART1 2p Header 4 port: Video IN1, Video IN2, Power IN, Power OUT
PCB Size	95 x 60 x 1.6 mm (L x W x H)

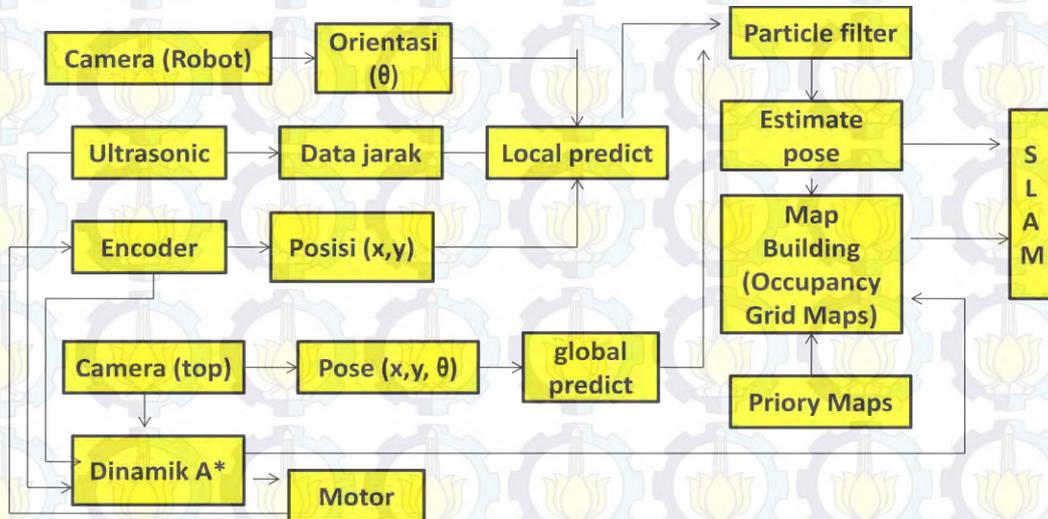
Tabel 2.4 Spesifikasi *Peripheral 2* [14].

■ Peri-2 모듈

Items	Hardware Specifications
Ethernet Controller	10/100 Non-PCI Ethernet Single Chip, SMSC LAN91C111
WLAN	IEEE 802.11b/g CF Wireless LAN Module
Video DAC	10Bit, 240MSPS Video D/A Converter
Audio	AC'97 Audio Codec 2 port speaker out 2 port MIC IN
UART	RS232 Level 1 port : UART0 TTL Level UART 3 port: UART1, UART2, UART3
USB	USB 1.1 Host
Peripheral Device	4pin DIP Switch 1ea, Push Switch 2ea, LED 4ea Compact size speaker 2ea Compact size Microphone 2ea
I/O Connector	9p D-SUB Connector 1ea: UART0 USB 1.1 Connector 1ae: B type RJ-45 Connector 1ea: 10/100Base-T Ethernet 4p Molex Connector 2ea: Video IN 15p D-SUB Connector 1ea: VGA out 20p Extension Connector 1ea: UART3/ GPIO 2pin/ FPGA IO 2pin
PCB Size	160 x 125 x 1.6 mm (L x W x H)

BAB 3 DESAIN DAN IMPLEMENTASI SISTEM

3.1 Diagram Blok Sistem

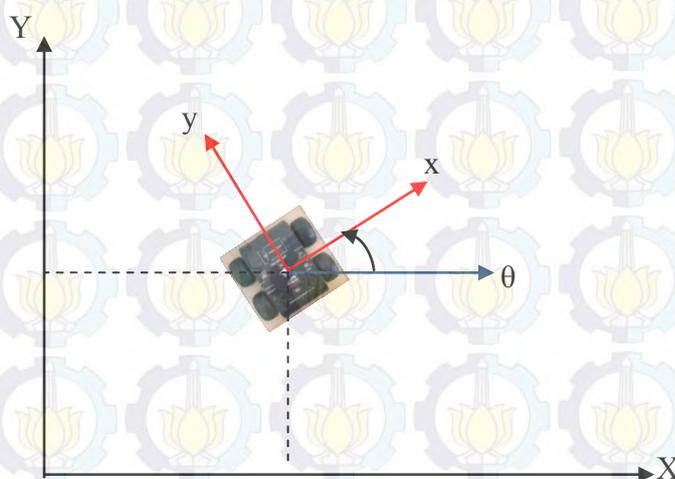


Gambar 3.1 Diagram Blok sistem SLAM

Untuk membangun SLAM dalam sistem ini digunakan beberapa metode yaitu *particle filter*, *dynamic A** dan *occupancy grid maps*. Dalam *particle filter* dibutuhkan input berupa *predict phase* dan *update phase*. *Predict phase* adalah fasa prediksi yang didapatkan dengan cara menyebar partikel yang mewakili dugaan robot. Kemudian *phase update* adalah fasa pada saat robot membaca data dari kinematika (*motion*) yang akan membentuk *pose* robot dan dibandingkan dengan *pose* dari setiap partikel. *Predict* yang dibutuhkan dalam sistem ini adalah *local* dan *global*. Bersifat *local* adalah dari sisi robot (pembacaan data sensor), *global* adalah dari sisi global yang diberikan (dalam sistem ini berasal dari kamera atas yang memantau jalannya robot). Untuk mendapatkan *pose local*, dalam sistem ini koordinat robot didapatkan dengan mengolah data dari rotary encoder (sensor putaran roda). Kemudian untuk orientasi (arah) adalah berasal dari kamera robot dengan melakukan pengolahan penanda-penanda (*landmark*) yang dipasang di lapangan atau area yang akan dilalui robot. Penggunaan *landmark* pada penelitian ini adalah karena keterbatasan sensor yang dipunyai HBE Robocar. Jadi terdapat dua solusi yang bisa digunakan yaitu menggunakan persepsi visual yang

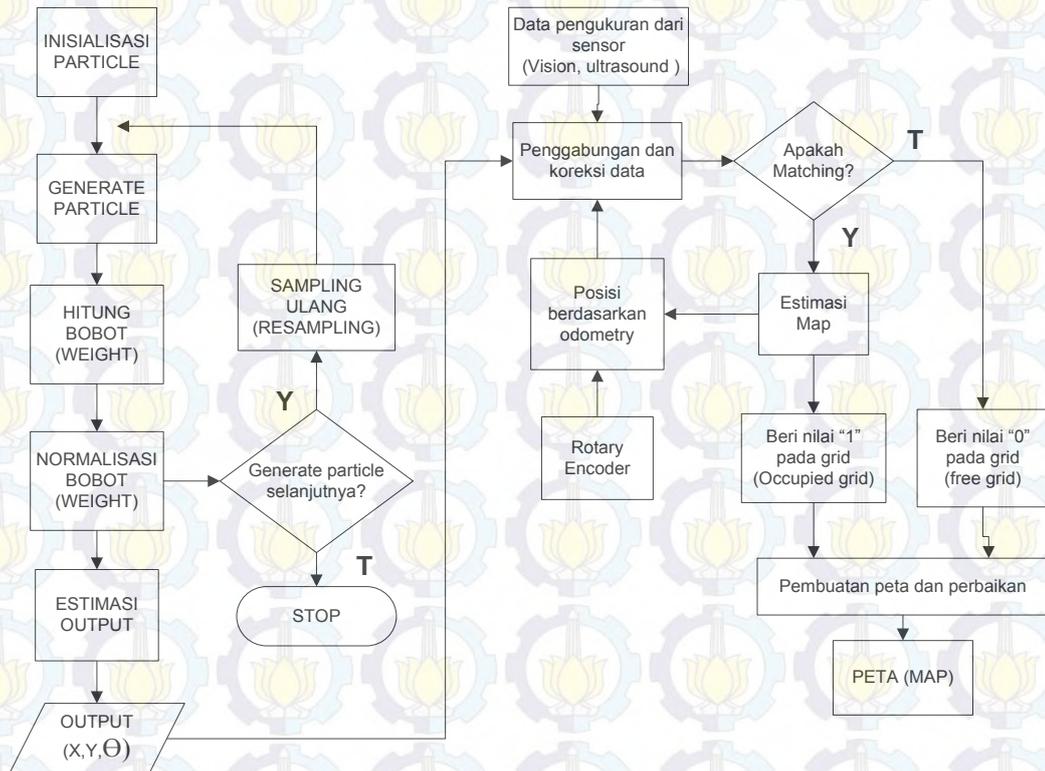
berasal dari kamera dan yang kedua adalah menggunakan kompas. tapi pada penelitian ini menggunakan sistem persepsi kamera karena pada robot juga terdapat kamera yang bisa dimanfaatkan.

Untuk menentukan jalur terdekat yang akan dilalui robot digunakan algoritma *dynamic A**. Data yang berasal dari kamera global (kamera atas) digunakan untuk menentukan jalur terdekat dari titik awal robot (*start*) ke titik tujuan akhir (*goal*). Karena dari sisi lokal robot belum mengetahui penghalang dalam mencapai titik *goal*. Maka digunakan ultrasonik sensor untuk mendeteksi penghalang yang berada disekitar lingkungan dalam melintasi ke titik akhir (*goal*). Data lokal *predict* akan diolah menggunakan algoritma partikel filter untuk mendapatkan *estimate pose* (perkiraan koordinat dan arah robot). Estimasi posisi ini akan dikombinasikan dengan *prior maps* (peta yang sudah diketahui) sebagai koreksi hasil. Hasil estimasi *pose* dan jangkauan sensor ultrasonik direpresentasikan dengan *grid maps*. Dalam *grid maps*, *grid* yang dijangkau oleh sensor akan di *occupied* (diberi nilai “1”) dan untuk *grid* yang tidak terjangkau sensor maka tidak *teroccupied* (diberi nilai “0”). Untuk mendeskripsikan masalah penentuan posisi, sistem koordinat secara global dan koordinat robot digambarkan seperti terlihat pada gambar 3.2. *Pose* dari robot merupakan posisi relatif robot terhadap koordinat global lapangan (x, y) dan orientasi robot terhadap sumbu x koordinat lapangan (θ).



Gambar 3.2 Sistem koordinat *global* dan sistem koordinat robot.

3.2 Diagram Alir Sistem



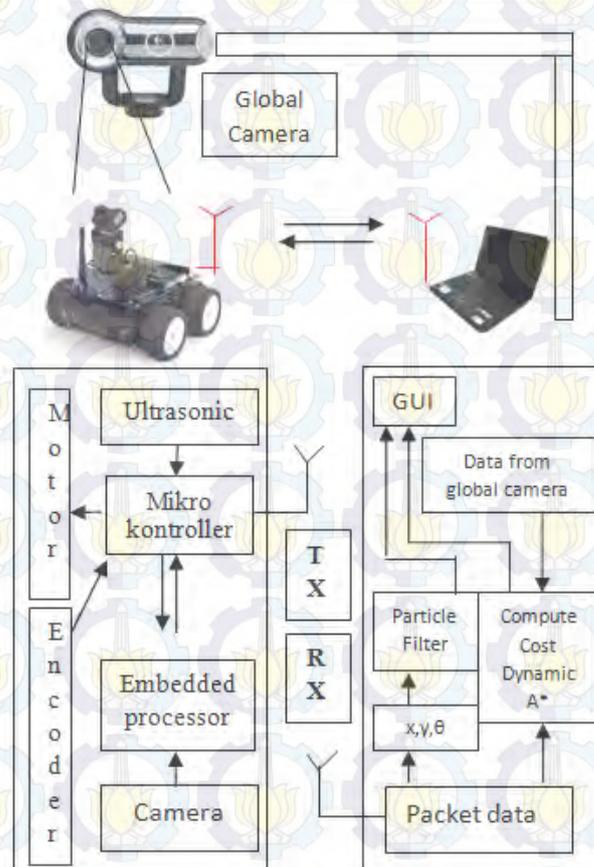
Gambar 3.3 Diagram alir proses SLAM

Dari diagram blok sistem yang sebelumnya disebutkan, dapat dijelaskan lebih rinci secara diagram alir mengenai proses yang berjalan pada robot, meliputi cara kerja partikel filter sampai dengan proses peta yang dihasilkan. Langkah pertama yang dilakukan adalah menginisialisasi partikel yaitu dengan menyebar beberapa partikel yang mewakili dugaan awal posisi robot. Selanjutnya setiap partikel yang disebar akan dihitung bobotnya (*weight*) dengan cara membandingkan perbedaan *actual measurement* (pengukuran yang dilakukan robot) dengan *predict measurement* (pengukuran *landmark* yang dilakukan oleh partikel). Setelah didapatkan selisih pengukuran antara partikel dengan robot, akan dilakukan normalisasi bobot agar untuk mengetahui bobot tertinggi dari suatu partikel. Partikel dengan bobot yang tinggi akan dijadikan acuan untuk persebaran partikel selanjutnya di daerah sekitar partikel dengan bobot tinggi tersebut.

Estimasi *pose* hasil partikel filter akan dikoreksi ulang dengan data yang berasal dari kamera dan ultrasonik. Data dari ultrasonik akan menentukan grid peta yang teroccupied atau tidak. Untuk *grid* yang terjangkau oleh data sensor ultrasonik akan diberi nilai “1” dan untuk *grid* yang tidak terjangkau akan diberi nilai “0”. Susunan dari *grid* yang diberi nilai “1” akan membentuk serangkaian *grid* yang mewakili bentuk peta yang dijangkau oleh robot.

3.3 Perangkat Keras Sistem

Untuk mengimplementasikan beberapa metode yang digunakan dalam sistem ini maka dibutuhkan beberapa perangkat keras. Sistem perangkat keras dan konfigurasi sistem ditunjukkan dengan gambar 3.4 dibawah ini:



Gambar 3.4 Setting Konfigurasi Sistem

Dalam konfigurasi sistem pada Gambar 3.4 terdapat beberapa bagian diantaranya data dari sensor yang digunakan oleh robot yaitu sensor kamera, sensor ultrasonik, *rotary encoder*. Kamera dibagi menjadi dua bagian yaitu global

dan lokal. Untuk global position, kamera yang digunakan adalah kamera USB Logitech. Sedangkan kamera yang digunakan di lokal robot yaitu kamera *embedded* yang diproses oleh mini PC yang ada pada robot. Kemudian digunakan sensor ultrasonik untuk mendapatkan data jarak. *Rotary Encoder* digunakan untuk mendapatkan data jarak tempuh robot melalui radius roda yang digunakan,

Ketiganya akan memberikan informasi berupa *local predict* (persepsi letak dan koordinat yang dipahami robot lewat pengolahan data sensor). Biasanya disebut informasi *sensing model*. Kamera (atas) memberikan informasi posisi dan koordinat robot dengan mengolah citra yang didapatkan (*tracking color*).

Data dari kamera akan menjadi *global predict*. Data dari kamera global digunakan untuk menginformasikan tujuan akhir robot. Semua data robot dikirimkan melalui modul *transceiver* berupa *bluetooth* dan *wifi* ke PC. Dari PC dilakukan *parsing* data untuk mendapatkan satu persatu data dari data paket. Untuk perencanaan jalur menggunakan algoritma dinamik A* dengan mengolah data dari kamera global (tampak atas) dengan data dari rotary encoder. Kamera atas digunakan untuk memberitahukan robot tentang jalur terdekat yang dapat dilalui robot dengan menerapkan perhitungan harga $g(n)$ dan $h(n)$ dari titik awal menuju ke titik akhir dengan mempertimbangkan *obstacle* atau penghalang yang ada.

Output dari dinamik A* yaitu gerakan pada motor yang memandu robot untuk menuju *goal* dengan tidak melupakan informasi lokal yang diperoleh robot melalui sensor ultrasonik. Setiap robot mendeteksi penghalang, maka robot melakukan perhitungan ulang harga $g(n)$ dan $h(n)$ yang terdekat untuk memandu robot mencapai titik goal yang diinginkan.

Output dari dinamik A* juga berguna untuk *increment maps building* dalam pembangunan peta. Kombinasi data dilakukan antara estimasi *pose* robot dengan *priory maps* (peta yang sudah disediakan) membentuk SLAM. Jadi secara hasil terjadi *increment* (peningkatan) secara *step by step* dari data lokal sensor untuk memberikan pemahaman hasil jelajah robot dalam mengenali lingkungan yang belum diketahui.

3.4 Penentuan Posisi menggunakan algoritma Partikel Filter

Penggunaan partikel filter untuk penentuan posisi pada penelitian ini adalah sebagai berikut:

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

Gambar 3.5 Pseudo code particle filter

Pada *line* 4 berlaku untuk setiap partikel mulai dari $m = 1$ sampai dengan M . M adalah banyaknya partikel. Maka dilakukan persebaran sampel yang mewakili posisi robot sebenarnya yang tersebar di luasan lapangan. Setiap *pose* (x, y, θ) sampel ditentukan oleh $p(x_t | u_t, x_{t-1})$ dengan arti bahwa *pose* didapatkan dari u_t yaitu kontrol pergerakan terhadap x_{t-1} yaitu *pose* sebelumnya. Kemudian pada *line* 5, setiap partikel dihitung bobotnya w_t dengan cara $p(z_t | x_t)$ yang mempunyai arti z_t adalah pengukuran terhadap x_t yaitu *pose* sekarang (*pose* yang telah sebelumnya sudah digerakkan terhadap x_{t-1}).

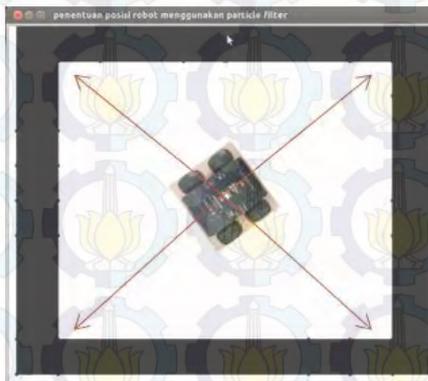
Tampak pada *line* 4 dan 5 bahwa metode partikel filter membutuhkan dua masukan utama yang dimodelkan dengan model sensor dan model gerak. Model sensor memberi masukan berupa informasi jarak robot dengan penanda yang bisa dijadikan acuan. Model gerak memberi masukan berupa informasi odometri robot. Informasi ini digunakan sebagai perbaruan posisi dari setiap partikel. Dalam sistem ini, model sensor didapat dari persepsi visual robot. Sedangkan untuk model gerak didapat dari informasi kinematik robot.

Kemudian partikel-partikel tersebut dipilih secara acak berdasarkan bobot setiap partikel sehingga didapat komposisi partikel baru yang konvergen

dan mendekati posisi robot sebenarnya. Proses ini disebut dengan *resampling*. Setelah itu posisi setiap partikel akan diperbarui berdasarkan informasi odometri robot terakhir. Partikel dengan bobot terbesar dipilih sebagai solusi terbaik.

3.4.1 Sensoring Model

Untuk *sensing model* pada penelitian ini digunakan dua sistem yang berbeda yaitu untuk simulasi dan untuk robot. Pada simulasi digunakan sistem persepsi sensor jarak yang selalu bisa dijangkau oleh robot. Atau dengan kata lain selalu bisa didapatkan *pose* dimanapun robot berada.



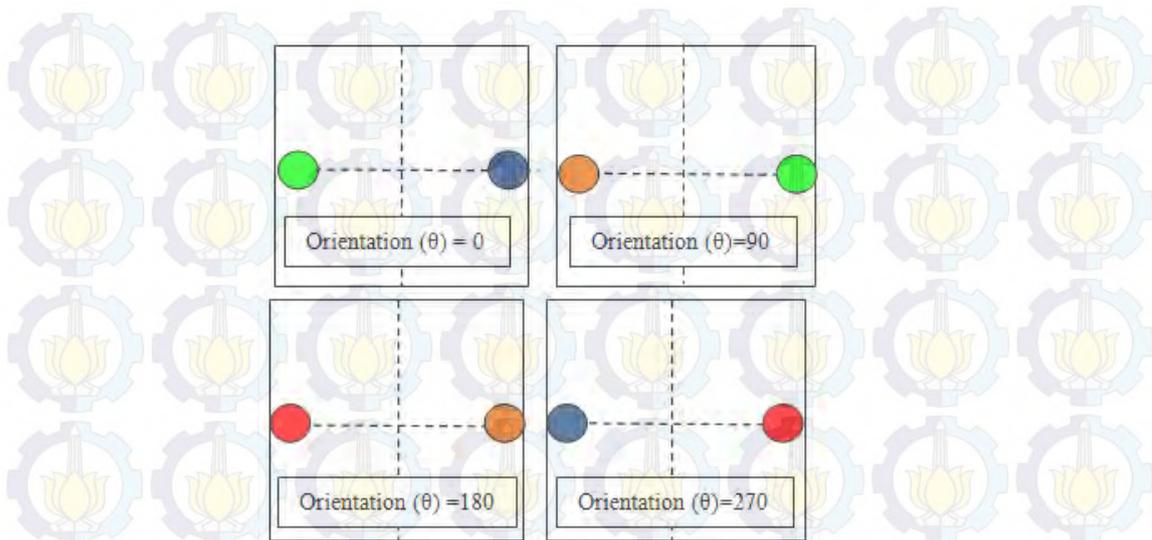
Gambar 3.6 Sistem persepsi yang digunakan dalam simulasi

Selanjutnya *sensing model* untuk robot yang sebenarnya menggunakan sistem persepsi visual kamera terhadap *landmark* yang dipasang di sudut lapangan. Persepsi sudut 0 dibentuk dari robot saat melihat bola warna hijau dan biru. Kemudian sudut 90 dibentuk saat robot mendeteksi bola warna orange dan hijau.

Selanjutnya untuk sudut 180 didapatkan saat robot mendeteksi bola berwarna merah dengan orange. Dan untuk sudut 270 didapatkan dari warna biru dan merah.



Gambar 3.7 Tracking color pada robot.



Gambar 3.8 Sistem persepsi yang digunakan dalam robot.

3.4.2 Motion Model

Untuk melakukan perbaruan sampel *pose* yang disebar maka dibutuhkan *motion model*. *Motion model* pada sistem partikel filter didapatkan dari model Odometri. Odometri adalah penggunaan data dari pergerakan aktuator untuk memperkirakan perubahan posisi dari waktu ke waktu. odometri digunakan untuk memperkirakan posisi relatif terhadap posisi awal.

Untuk memperkirakan posisi relatif robot, digunakan perhitungan jumlah pulsa yang dihasilkan oleh sensor rotary encoder setiap satuan ukuran yang kemudian dikonversi menjadi satuan millimeter. Untuk mendapatkan jumlah pulsa setiap satu kali putaran roda digunakan rumus sebagai berikut:

$$K \text{ Roda} = 2 \pi r$$

$$\text{pulsa per mm} = \text{resolusi encoder} / K \text{ Roda} \quad (3.1)$$

Pada sistem gerak diferensial terdapat dua roda, yaitu roda kanan dan roda kiri dan dimisalkan jumlah pulsa_per_mm untuk roda kanan adalah *right_encoder* dan roda kiri adalah *left_encoder* dan jarak antara dua roda adalah *wheel_base* maka didapatkan jarak tempuh (*distance*) dan sudut orientasi (θ). Rumusnya adalah sebagai berikut.

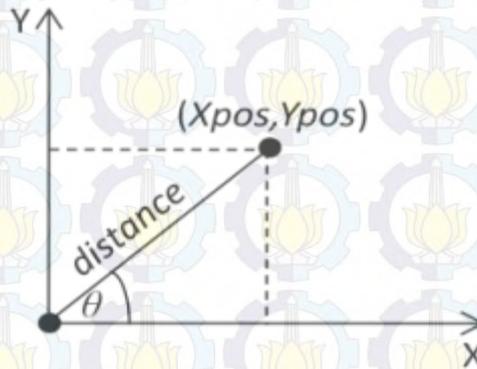
$$\text{distance} = (\text{left enc} + \text{right enc}) / 2$$

$$\theta = (\text{left enc} - \text{right enc}) / \text{wheel_base} \quad (3.2)$$

Karena θ adalah sudut dalam radian maka untuk mengetahui sudut dalam derajat (*heading*) digunakan rumus sebagai berikut :

$$heading = \theta \times \frac{180}{\pi} \quad (3.3)$$

Dari ketentuan diatas didapatkan bahwa nilai *heading* akan bernilai negatif (-) ketika robot berputar melawan arah jarum jam dan akan bernilai positif (+) ketika robot berputar searah dengan jarum jam. Dengan mengetahui jarak dan sudut (*distance* dan θ) maka kita dapat mengetahui koordinat X dan koordinat Y dengan persamaan trigonometri.



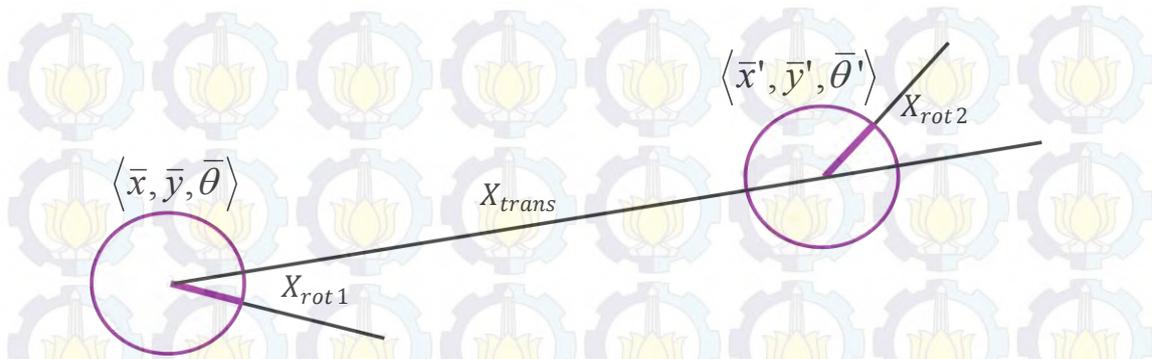
Gambar 3.9 Sistem Koordinat pose

Dari ilustrasi diatas maka koordinat dari robot dapat kita ketahui dengan rumus :

$$\begin{aligned} X_{pos} &= distance \times \sin(\theta) \\ Y_{pos} &= distance \times \cos(\theta) \end{aligned} \quad (3.4)$$

Robot yang bergerak dari $\bar{x}, \bar{y}, \bar{\theta}$ ke $\bar{x}', \bar{y}', \bar{\theta}'$ maka berlaku translasi dan rotasi dari x_{t-1} ke x_t antara dua titik keadaan robot adalah

$$\begin{aligned} X_{trans} &= \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \\ X_{rot1} &= atan^2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \\ X_{rot2} &= \bar{\theta}' - \bar{\theta} - X_{rot1} \end{aligned} \quad (3.5)$$



Gambar 3.10 Odometri robot

Adapun syarat penggunaan dari atan^2

$$\text{atan}^2(y, x) = \begin{cases} \text{atan}(y/x) & \text{jika } x > 0 \\ \text{sign}(y)(\pi - \text{atan}(|y/n|)) & \text{jika } x < 0 \\ 0 & \text{jika } x = y = 0 \\ \text{sign}(y)\pi/2 & \text{jika } x = 0, y \neq 0 \end{cases} \quad (3.6)$$

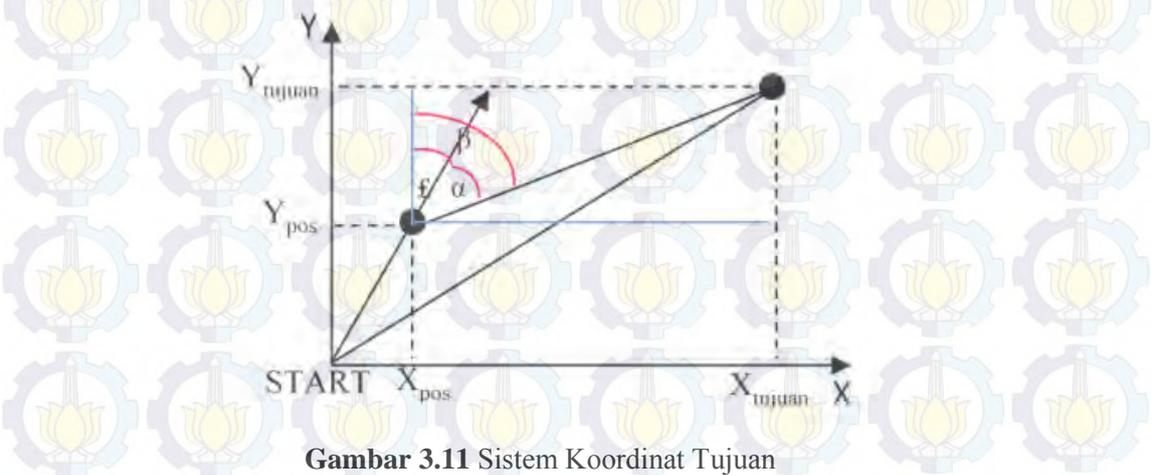
Untuk menentukan *error* arah hadap dari robot terhadap titik tujuan maka digunakan *teorema pythagoras* yang akan menghasilkan posisi saat ini dan jarak terhadap titik tujuan, berikut perhitungannya:

$$x = X_{\text{tujuan}} - X_{\text{pos}}$$

$$y = Y_{\text{tujuan}} - Y_{\text{pos}}$$

$$\text{target distance} = \sqrt{x^2 + y^2} \quad (3.8)$$

Arah hadap dari robot yang telah diketahui sehingga kita dapat menghitung *error* arah hadap (*heading error*) robot terhadap titik tujuan.



Gambar 3.11 Sistem Koordinat Tujuan

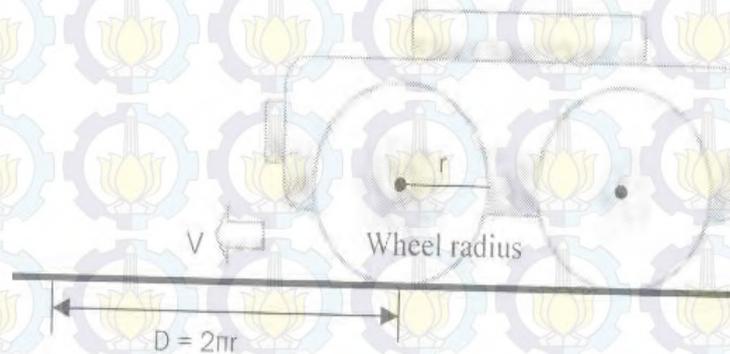
Pada gambar 3.11. menunjukkan ilustrasi untuk mencari *heading error* (α) dimana β adalah *target bearing* yaitu sudut antara posisi robot saat ini terhadap titik tujuan. Sedangkan garis berwarna biru adalah garis bantu yang masing-masing sejajar dengan sumbu X dan sumbu Y . Untuk mendapat nilai dari β , maka digunakan rumus sebagai berikut :

$$\beta = \arctan \frac{(Y \text{ tujuan} - Y \text{ Pos})}{(X \text{ tujuan} - X \text{ pos})} \quad (3.9)$$

dan

$$\alpha = \beta - \epsilon \quad (3.10)$$

Berikut ini adalah *motion model* pada HBE Robocar



Gambar 3.12 Rotasi roda untuk perhitungan jarak [14]

Diameter roda robot r berputar dengan kecepatan sudut ω maka berlaku:

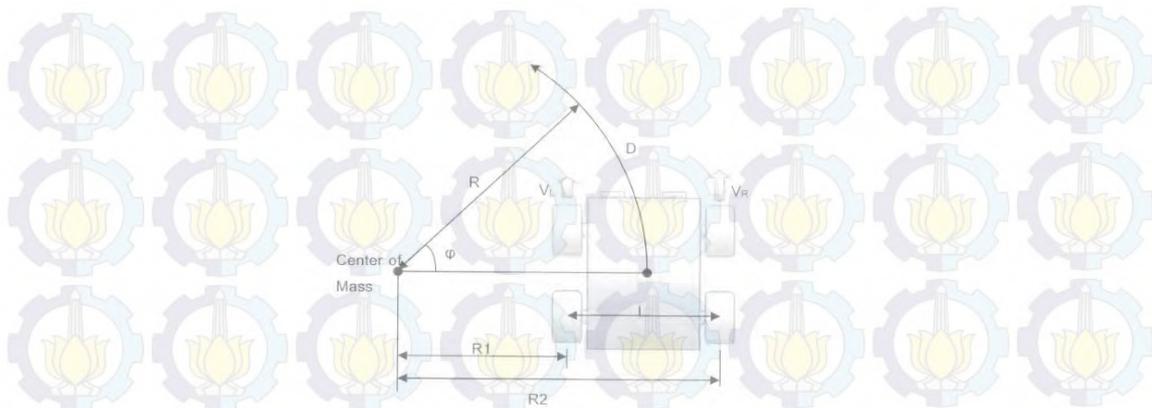
$$V = r \cdot \omega \quad (3.11)$$

Dan jika robot bergerak selama t dengan kecepatan V maka jarak D didapatkan dengan

$$D = V \cdot t \quad (3.12)$$

Jarak D adalah sama dengan keliling roda robot, maka berlaku

$$D = 2\pi r \quad (3.13)$$



Gambar 3.13 Perputaran robot pada sumbu *center of mass* [14]

Kemudian untuk radius titik tengah robot R yaitu jarak yang dibentuk oleh R_1 dan R_2 . Dengan kecepatan kiri disimbolkan dengan V_L dan kecepatan kanan disimbolkan dengan V_R . Berlaku perbandingan:

$$V_L : R_1 = V_R : R_2 \quad (3.14)$$

Dengan persamaan diatas maka

$$\frac{V_L}{R - \frac{L}{2}} = \frac{V_R}{R + \frac{L}{2}} \quad (3.15)$$

Kemudian untuk mendapatkan R radius putaran maka:

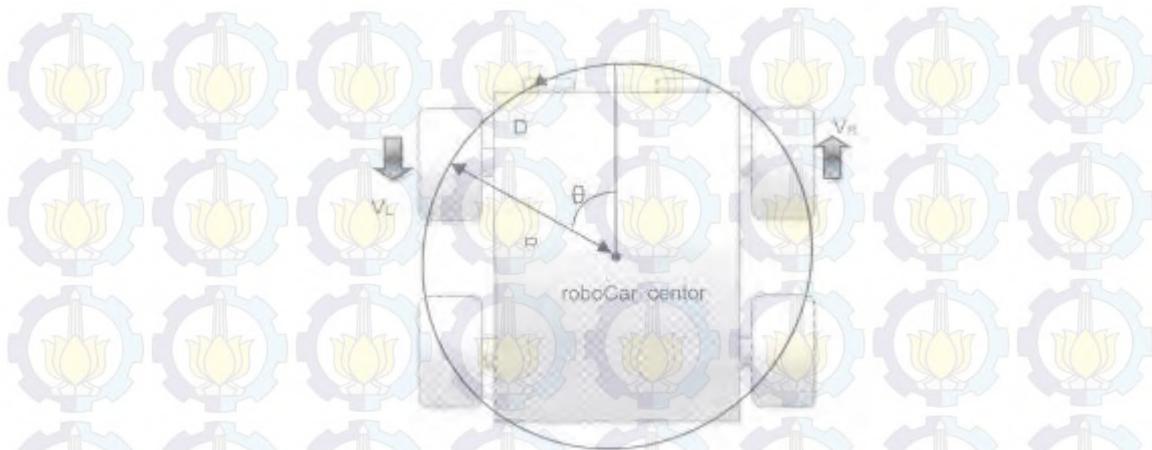
$$R = \frac{L}{2} \cdot \frac{V_R + V_L}{V_R - V_L} \quad (3.16)$$

Saat $R = \infty$ dan $V_R = V_L$ robot bergerak maju dan ketika $R = 0$ dan $V_R = -V_L$ maka robot berputar di tempat. Maka D bisa didapatkan dengan asumsi kecepatan tidak berubah selama robot bergerak, maka berlaku:

$$D = \frac{V_L + V_R}{2} \cdot t \quad (3.17)$$

Dan kita dapatkan sudut putar sebesar

$$\varphi = \frac{D}{R} \quad (3.18)$$



Gambar 3.14 Robot berputar di tempat [14]

Untuk gerakan robot berputar ditempat maka berlaku:

$$\theta = \frac{D}{R} (\text{radian}) \quad (3.19)$$

3.4.3 Resampling

Setelah dilakukan perbaruan *pose* yang didapatkan dari *motion* dan *sensing model*, maka langkah *resampling* yang digunakan adalah sebagai berikut:

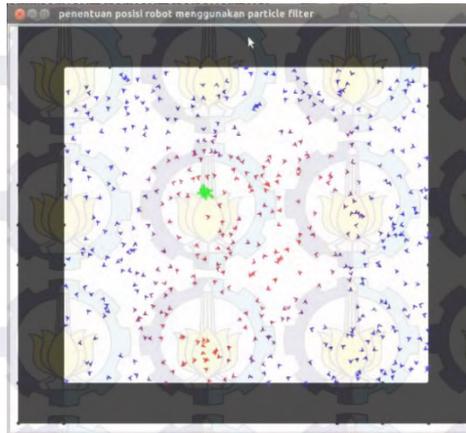
```

Masukan: partikel[n], bobot[n], hasil[n]
index = rand(0, 1) * n
beta = 0.0
for ( i = 0; i < n; i++ )
    beta += rand(0, 1) * 2.0 *
max(bobot)
while ( beta > bobot[index] ) do
    beta -= bobot[index]
    index = (index + 1) % n
    hasil[i] = partikel[index]
return (hasil)

```

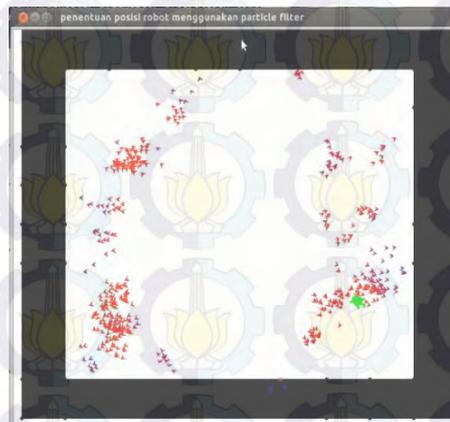
Setelah bobot dari setiap partikel diperbarui berdasarkan model sensor dan model gerak, dilakukan proses *resampling* untuk mendapatkan kumpulan partikel baru yang dipilih berdasarkan bobot setiap partikel. Partikel yang memiliki parameter bobot tinggi mempunyai kemungkinan lebih besar untuk dipilih, sedangkan partikel dengan bobot rendah memiliki probabilitas rendah untuk dipilih dalam kumpulan partikel yang baru. Sehingga persebaran partikel

baru berada di sekitar partikel dengan bobot yang tinggi. Pada gambar 3.15 dilakukan persebaran partikel sebanyak 200 secara acak ke area luasan lapangan.



Gambar 3.15 Penyebaran partikel secara acak

Pada gambar 3.16 menunjukkan hasil resampling ke 11 yaitu partikel yang mulai terpilih berdasarkan bobotnya. Beberapa partikel berdekatan dengan letak yang berbeda. Hal ini dikarenakan kemungkinan *belief* yang sama.



Gambar 3.16 Resampling ke 11 proses penentuan bobot

Pada gambar 3.17 adalah hasil *resampling* ke 17 dengan partikel yang memiliki bobot yang tinggi dan mewakili posisi robot yang sebenarnya.



Gambar 3.17 Resampling ke 17 partikel dengan bobot tinggi

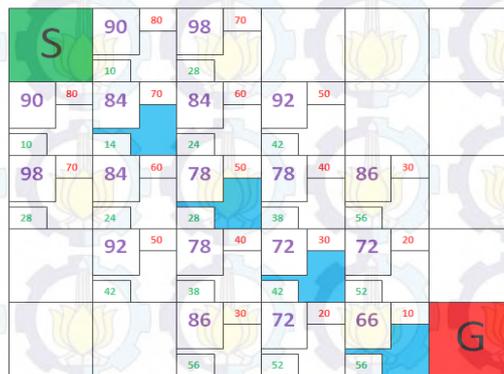
Pada gambar 3.18 robot bergerak dengan *update* pergerakan menuju penanda berikutnya. Dan persebaran partikel adalah disekitar bobot yang tinggi.



Gambar 3.18 Resampling ke 39 penyebaran partikel di sekitar partikel bobot tinggi.

3.5 Perencanaan jalur menggunakan Algoritma Dinamik A*

Seperti yang telah dijelaskan pada bab II point 2.6 tentang proses perhitungan dinamik A*, langkah penentuan jalur robot dilakukan dengan cara membuat *grid* pada area lapangan seperti pada gambar 3.19.



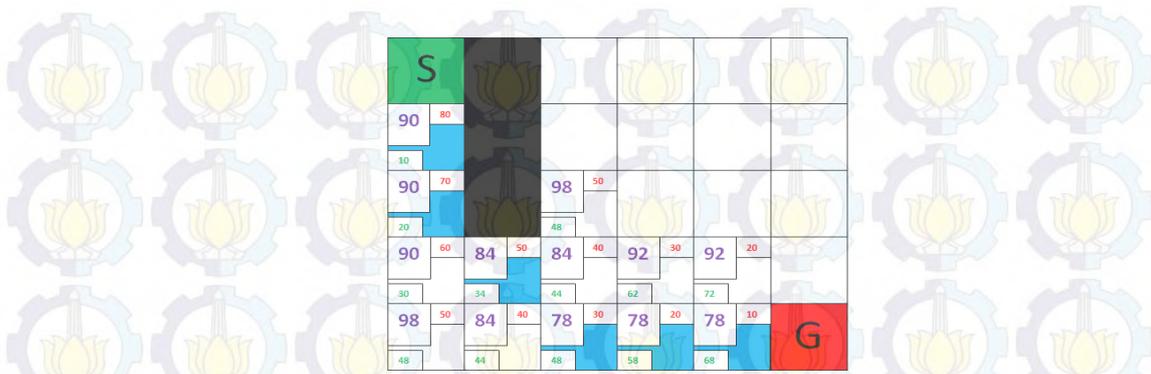
Gambar 3.19 Perhitungan $g(n)$ dan $h(n)$ dari start ke goal.

Pada proses awal dilakukan perhitungan setiap nilai *grid* yang akan dilalui robot dari *start* ke *goal*. Setelah didapatkan nilai dari setiap *grid*, kemudian dipilih *grid* dengan nilai $f(n)$ yang paling kecil diantara *grid-grid* di sekitarnya. Setiap didapatkan *grid* yang terpilih, maka dilakukan perhitungan selanjutnya untuk *grid* yang mempunyai nilai $f(n)$ yang kecil. Pada gambar 3.19 *grid* yang mempunyai nilai $f(n)$ yang kecil berwarna biru muda.



Gambar 3.20 Perhitungan ulang $f(n)$

Setelah ditemukan jalur terdekat dari *start* ke *goal* (berwarna biru) maka robot harus menuju ke jalur tersebut. Pada saat robot menuju ke satu *grid* berwarna biru, robot mendeteksi bahwa *grid* tersebut tidak bisa dilalui karena *grid* tersebut adalah *obstacle* (penghalang). Dalam gambar 3.20 diberi warna hitam. Akibatnya proses perhitungan ulang dilakukan untuk mencari jalur terdekat lainnya.



Gambar 3.21 Perhitungan ulang untuk setiap grid yang diketahui sebagai obstacle $f(n)$

Proses perhitungan terjadi berulang pada saat robot menemukan satu *grid* yang harus dilalui akan tetapi *grid* tersebut merupakan *obstacle* (penghalang) yang mengakibatkan *grid* tersebut tidak bisa dilalui.

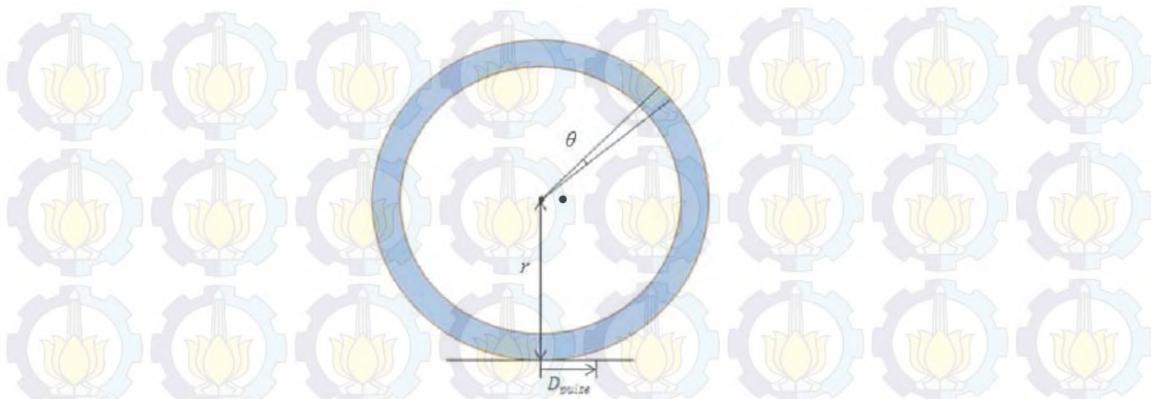


Gambar 3.22 Robot mencapai titik goal.

Sampai pada titik terakhir goal dicapai oleh robot. Jalur yang dilalui robot ditunjukkan dengan warna biru muda dan warna hitam adalah *obstacle* yang dideteksi oleh robot. Untuk bergerak ke *grid* yang mempunyai $f(n)$ yang kecil, dalam implementasinya digunakan perintah *serial communication* untuk menggerakkan robot sejauh 1 satuan *grid* dari data *rotary encoder*. Dengan memakai hubungan *pulse rotary* dan radius roda robot maka

$$D_{pulsas} = \frac{1}{h} \left(2\pi r \times \frac{n\theta}{360^\circ} \right) \quad (3.11)$$

r adalah radius roda robot (42mm), n = jumlah pulsa yang dihasilkan, θ =derajat sudut (1.8°), h = factor untuk half step atau full step. 1 untuk full step dan 2 untuk half step [22] Hubungan notasi dapat digambarkan sebagai berikut:



Gambar 3.23 Pengukuran radius putaran roda robot [22].

Untuk perintah *serial communication* yang digunakan adalah memakai beberapa huruf dengan keterangan sebagai berikut:

Tabel 3.1 Perintah Komunikasi Serial

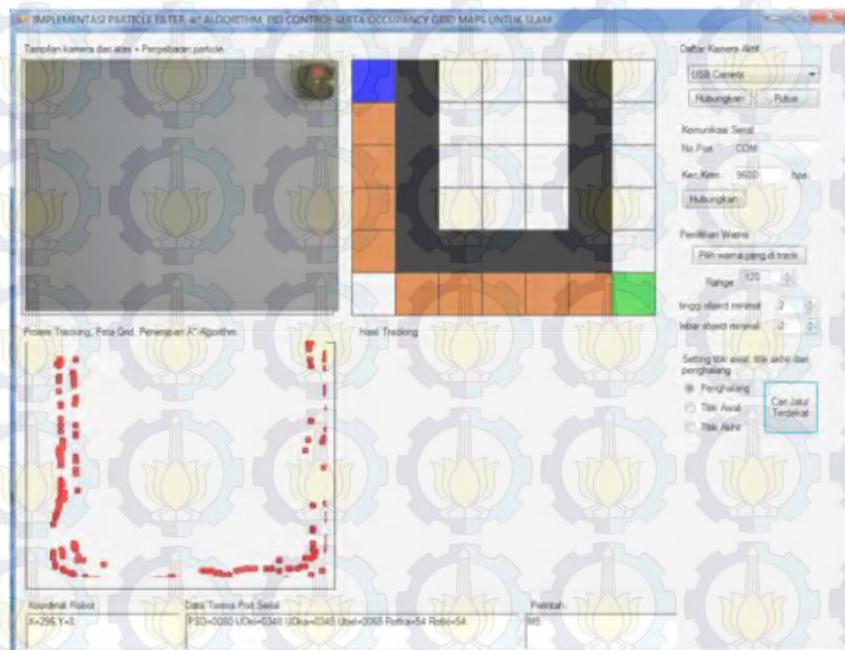
Perintah	Keterangan
M<n>	Maju sejauh n <i>grid</i> (kotak)
Kr<d>	Putar Kiri sampai d derajat
Kn<d>	Putar Kanan sampai d derajat
B	Gerak Belakang/Mundur
S	<i>Stop</i> (berhenti)
F	<i>Finish</i> (akhir instruksi / <i>command</i>)

Contoh perintah yang dikirimkan komputer untuk gerak maju sejauh 4 kotak, maka computer mengirimkan M4. Untuk gerak putar kanan dan kiri digunakan perintah Kr dan Kn diikuti dengan derajat putar robot. Contohnya Kr45 (kiri 45°). Data ini diolah oleh mikrokontroller untuk diubah ke besaran tegangan yang diumpankan ke *driver* motor dengan umpan balik atau koreksi hasil berupa pembacaan putaran roda oleh *rotary encoder*.

3.6 *Graphical User Interface (GUI)*

GUI dalam sistem ini menggunakan *software Visual Studio 2010* yang diintegrasikan dengan *OpenCV* untuk akses kamera *eksternal* dan melakukan pengolahan citra. Dalam *GUI* terdapat beberapa informasi yang ditampilkan yaitu tampilan kamera global, hasil *track* robot, perintah komunikasi serial, data terima *port serial*, setting komunikasi serial, koordinat robot dan hasil A*. *Software* ini mengolah data kamera dan menjalankan fungsi A*. Output dari sebuah pemandu

gerak berupa data komunikasi serial yang dikirim ke robot menggunakan akses kontrol melalui *serial port*. Kemudian untuk pengolahan data partikel filter dan *occupancy grid maps* adalah dengan melakukan rekonstruksi ulang data yang di *record* oleh robot. Untuk pengujian *particle filter* dan *occupancy grid maps* tidak dilakukan secara *online*. Karena kebutuhan komputasi yang cukup besar untuk mengolah jumlah partikel serta melakukan pembangunan peta hasil estimasi posisi dan orientasi robot.



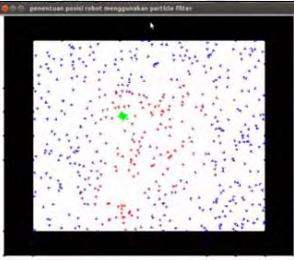
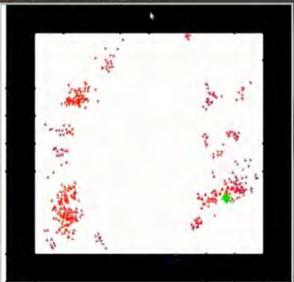
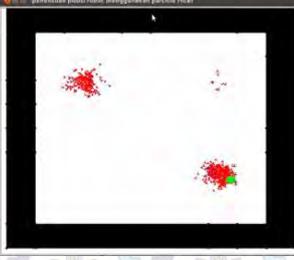
Gambar 3.24 Tampilan User Interface

BAB 4 PENGUJIAN SISTEM DAN ANALISA

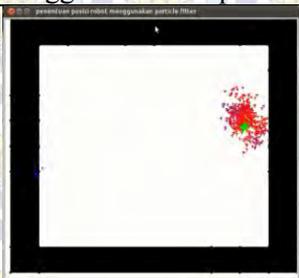
4.1 Pengujian penentuan posisi menggunakan partikel filter (simulasi)

Pada pengujian ini dilakukan dengan mengujicobakan beberapa partikel dengan jumlah yang berbeda-beda yaitu 200, 400 dan 600 partikel. Untuk *sensing model* digunakan sistem persepsi yang telah dijelaskan pada bab III point 3.41. Berikut ini hasil pengujian dengan menggunakan jumlah partikel yang berbeda.

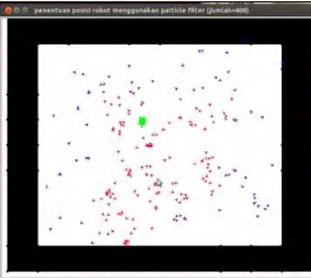
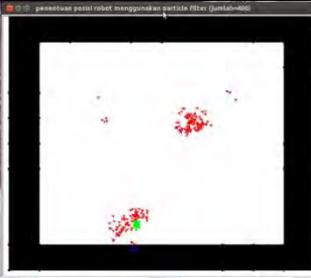
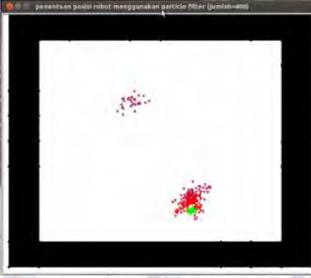
Tabel 4.1 Hasil penentuan posisi menggunakan 600 partikel.

No.	Resampling ke	Hasil simulasi
1	0	
2	7	
3	16	

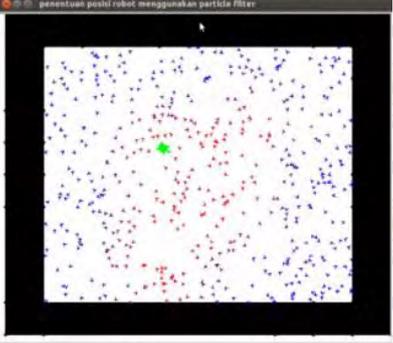
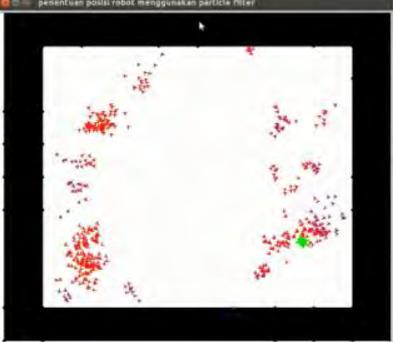
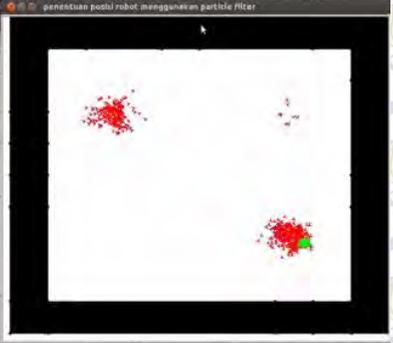
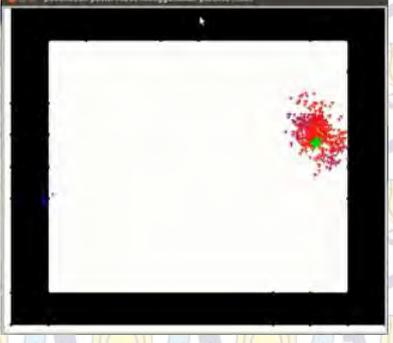
Lanjutan Tabel 4.1 Hasil penentuan posisi menggunakan 600 partikel.

4	19	
---	----	--

Tabel 4.2 Hasil penentuan posisi menggunakan 400 partikel.

No	Resampling ke	Hasil simulasi
1	0	
2	17	
3	21	
4	24	

Tabel 4.3 Hasil penentuan posisi menggunakan 200 partikel.

No	Resampling ke	Hasil simulasi
1	0	
2	11	
3	28	
4	31	

Dari semua data hasil pengujian simulasi menggunakan 200, 400, dan 600 partikel didapatkan data sebagai berikut:

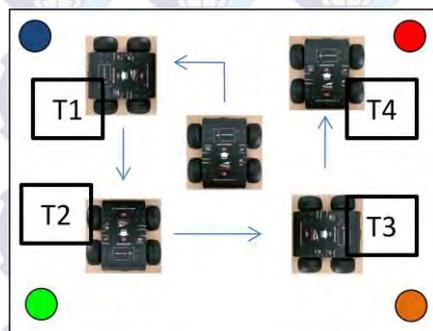
Tabel 4.4 Data perbandingan estimasi posisi dengan jumlah partikel.

Point	Robot Position			200 particles			400 particles			600 particles		
	x	y	θ (rad)	x	y	θ_p (rad)	x	y	θ_p (rad)	x	y	θ_p (rad)
A-B	36	10	0.017	32.4	9.4	0.019	35.3	7.8	0.019	40.2	6.8	0.018
B-C	214	216	0.031	222.8	199	0.024	197.3	231.1	0.028	196	225	0.041
C-D	304	146	0.024	346	122	0.029	312.1	158.2	0.029	331	161	0.026
D-A	178	8	0.069	202	30	0.059	156.8	19.7	0.057	176	4.6	0.052
Average error = $\frac{X - X_p}{X} \times 100\%$				10.35	26.98	17.41	3.41	4.03	65.79	7.45	15.5	20.6
Iterasi ke				118			42			37		
Waktu				50detik			86detik			219detik		

Dari data diatas dapat dianalisa bahwa semakin banyak jumlah partikel maka waktu yang dibutuhkan semakin lama tapi jumlah iterasi untuk mencapai konvergen partikel lebih sedikit. Dan sebaliknya jika digunakan partikel dalam jumlah kecil maka waktu yang diperlukan lebih singkat tapi dibutuhkan jumlah iterasi yang lebih banyak.

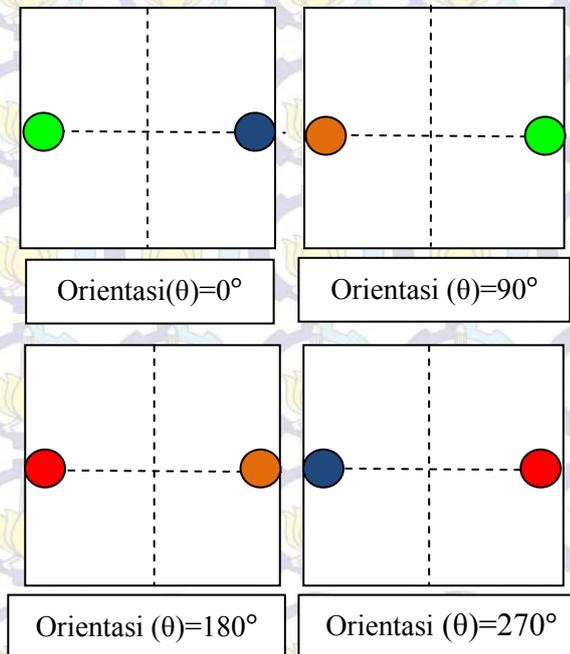
4.2 Pengujian penentuan posisi menggunakan partikel filter (pada robot)

Pada pengujian ini dilakukan pada lapangan berukuran 350cm x 250cm dengan 4 titik percobaan dan alur pergerakan robot dari satu titik ke titik yang lain adalah sebagai berikut:



Gambar 4.1 Titik-titik pengujian

Digunakan persepsi *landmark* dengan beberapa kondisi untuk sudut orientasi (θ) 0° pada saat bola warna hijau dan biru dideteksi. Kemudian untuk sudut orientasi (θ) 90° pada saat bola warna orange dan hijau dideteksi. Selanjutnya untuk sudut orientasi (θ) 180° pada saat bola warna merah dan orange dideteksi. Dan untuk sudut orientasi (θ) 270° pada saat bola warna biru dan merah dideteksi.



Gambar 4.2 Persepsi orientasi robot

Pada tabel 4.5 tentang data hasil pengujian estimasi posisi berdasarkan *landmark* menunjukkan hasil estimasi posisi T1 untuk selisih data posisi x dengan posisi yang sebenarnya mempunyai galat sebesar 6%, kemudian untuk selisih data posisi y dengan posisi yang sebenarnya mempunyai galat sebesar 13.8 %. Untuk rata-rata galat atas semua posisi x di titik uji T1 sampai T4 sebesar 4.7%. Kemudian untuk data posisi y di titik uji T1 sampai T4 sebesar 2.6%. Untuk data orientasi terdapat galat sebesar 10.8%. Galat yang besar untuk data orientasi adalah disebabkan karena sistem persepsi yang diberikan pada robot untuk orientasi adalah menggunakan data orientasi dengan jangkauan yang cukup jauh. Robot mengartikan data orientasi menggunakan derajat sudut dengan besaran 0° , 90° , 180° , 270° .

Tabel 4.5 Pengujian estimasi *pose* robot.

Titik Pengujian	Estimasi Posisi			Posisi sebenarnya			Deteksi citra
	x	y	θ	x	y	θ	
T1	6.7	3.1	44	0	0	65	
T2	01.8	4.2	88	00	0	15	
T3	04.9	89.4	71	00	00	40	
T4	4.7	00.5	3	0	00	5	

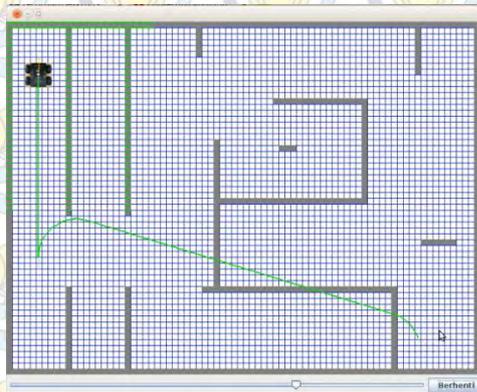
Dalam kenyataan, robot seharusnya mempunyai data untuk orientasi sebesar 0-360 °. Pada gambar 4.3 menunjukkan posisi robot terhadap penanda saat pengambilan data uji untuk posisi x , y dan orientasi.



Gambar 4.3 Posisi robot saat melakukan pengamatan *landmark* bola orange (kiri) bola biru (kanan)

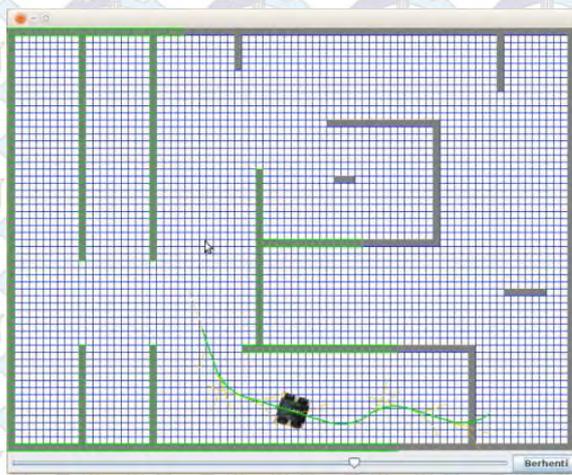
4.3 Pengujian navigasi menggunakan algoritma dinamik A*(simulasi)

Tahapan awal pengujian sistem yang dilakukan adalah membangun simulasi *dynamic A**. Simulasi *dynamic A** yang digunakan adalah bentuk modifikasi program dengan source asli pada [16]. Adapun hasil simulasi dinamik A* untuk menentukan jalur terdekat ditunjukkan dengan kondisi mula-mula informasi global yang diberikan berupa koordinat awal robot yaitu di titik S dan koordinat akhir yang harus dituju robot yaitu di titik G. Dalam kondisi ini robot belum mengetahui apakah ada penghalang dalam menuju titik akhir tersebut. Hasil jelajah robot terhadap *obstacle* ditandai dengan warna hijau, untuk penghalang yang belum diketahui diberi warna abu-abu. Pada kondisi ini hanya diketahui sebagian penghalang dari keseluruhan.



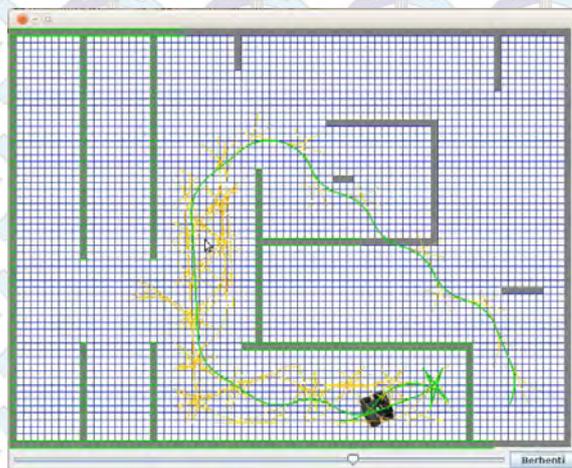
Gambar 4.4 Start awal robot

Selanjutnya robot bergerak mengikuti jalur *track* yang diberikan. Pada data jarak di *step* yang berikutnya, informasi lokal robot menginformasikan bahwa terdapat penghalang yang mengakibatkan robot tidak bisa menuju ke titik *goal*. Terjadi perhitungan ulang untuk menentukan jarak terdekat yang mungkin dicapai robot.



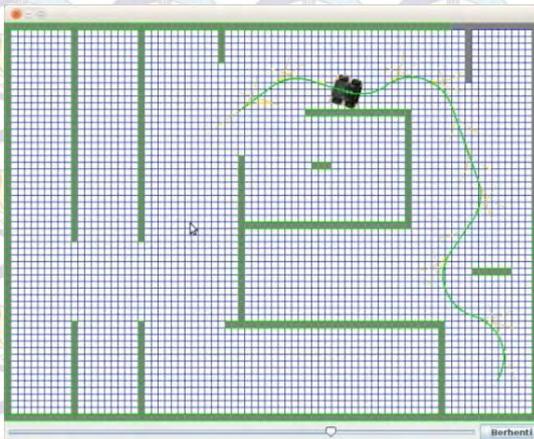
Gambar 4.5 Robot saat mengikuti *track*.

Setelah robot menemukan *update* jalur terbaru maka robot bergerak menuju titik *goal*.

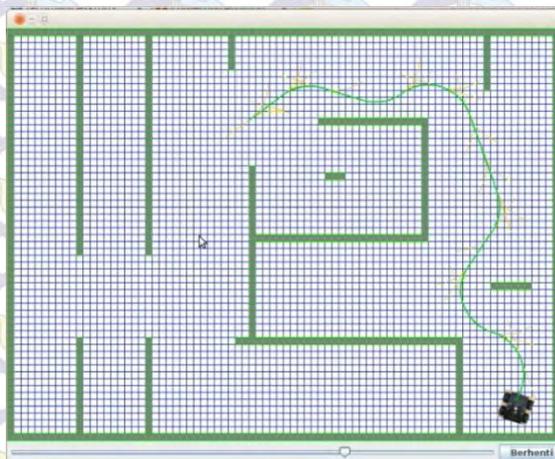


Gambar 4.6 Robot mencari jalur alternative.

Kondisi serupa berulang karena dari sisi lokal robot, robot membawa misi berupa penghindar halangan. Jadi dalam kondisi ini robot tidak bisa secara langsung menuju titik akhir karena *obstacle* baru diketahui bersamaan dengan penelusuran yang dilakukan. Setiap kali robot menemukan penghalang, maka robot selalu melakukan perhitungan ulang jarak terdekat untuk mencapai *finish*. Ditunjukkan dengan gambar 4.7 dan gambar 4.8.



Gambar 4.7 Robot kembali mencari jalur alternatif



Gambar 4.8 Robot mencapai titik *finish*.

Dilanjutkan dengan pengujian berulang sebanyak 10 kali dengan area dan *obstacle* yang sama. Sampai dengan pengujian ke 10, *dynamic A** mampu menentukan jalur terdekat dari titik *start* ke titik *goal*.

4.4 Pengujian navigasi menggunakan algoritma dinamik A*(Robot)

Konfigurasi sistem yang digunakan untuk pengujian navigasi adalah dijelaskan pada bab III poin 3.3. Menggunakan *view camera* atas sebagai *global position*. Dari *global position* dilakukan perhitungan $f(n)$ untuk mendapatkan jarak terdekat. Dari sisi lokal robot, robot harus melalui jalur hasil perhitungan *Dynamic A** tersebut. Gerakan robot untuk melalui *grid-grid* terdekat dijelaskan dalam bab III poin 3.5 yaitu menggunakan perintah komunikasi serial. Sebelum

dilakukan *running Dynamic A** secara *online* terlebih dahulu dilakukan pengujian derajat translasi dan rotasi robot.



Gambar 4.9 Pengujian putaran 0 derajat (kiri) dan putaran 45 derajat (kanan)



Gambar 4.10 Pengujian putaran 90 derajat (kiri) dan putaran 270 derajat (kanan)

Pengujian dilakukan sebanyak 10 kali. Berikut ini adalah data pengujian ke 1,

Tabel 4.6 Data pengujian ke-1. Translasi dan rotasi robot saat perintah serial diberikan

Perintah	Jarak/sudut sebenarnya	Hasil pengukuran jarak	Prosentase kesalahan
M1	30cm	29.2cm	2.6%
M2	60cm	61.4cm	2.3%
M3	90cm	88.8cm	1.3%
M4	120cm	118.2cm	1.5%
M5	150cm	150.8cm	0.5%
Kr45	45°	41,8°	7.1%
Kr90	90°	94.2°	4.6%
Kn45	45°	46°	2.2%
Kn90	90°	92.4°	2.6%

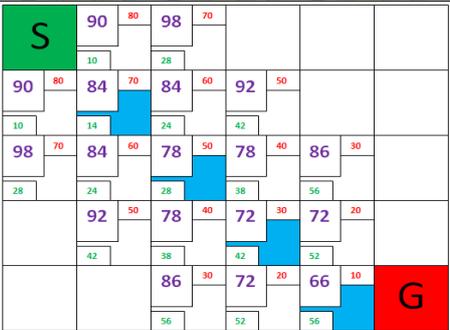
Tabel 4.7 Data pengujian ke-10. Translasi dan rotasi robot saat perintah serial diberikan

Perintah	Jarak/sudut sebenarnya	Hasil Pengukuran Jarak	Prosentase Kesalahan
M1	30cm	28.1cm	6.3%
M2	60cm	60.8cm	1.3%
M3	90cm	92.1cm	2.3%
M4	120cm	122.2cm	1.8%
M5	150cm	150cm	0%
Kr45	45°	43.4°	5.3%
Kr90	90°	94.5°	5%
Kn45	45°	46°	2.2%
Kn90	90°	92.4°	2.6%

Dari keseluruhan prosentase *error* yang didapat di setiap percobaan diambil rata-rata untuk data orientasi terdapat *error* sebesar 4.34 % dan *error* jarak tempuh robot sebesar 4.8 %. Galat yang terjadi pada data orientasi dan jarak tempuh robot disebabkan karena kesalahan odometri robot yang diambil dari data *rotary encoder*. Untuk perintah belok sudut 45 memiliki galat dengan presentasi yang lebih kecil dibandingkan dengan data derajat belok 90. Perintah serial yang diberikan adalah menggunakan satuan per *grid*.

Selanjutnya dilakukan pengujian secara online dengan data yang diperoleh dari kamera atas (*top view*)

Tabel 4.8 Data hasil pengujian *dynamic A** pada robot.

No	Posisi grid	Posisi grid robot	<i>error</i>
1			0 grid

Lanjutan Tabel 4.8 Data hasil pengujian *dynamic A** pada robot.

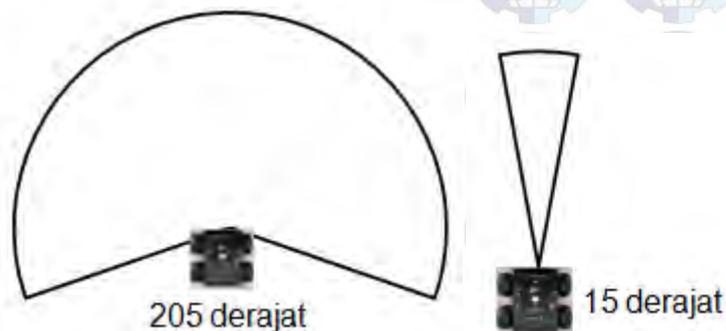
2			0 grid
3			1 grid
4			1 grid
5			1 grid

Pada data hasil pengujian *dynamic A** terdapat galat *grid* pada saat *grid* ke 3, ke 4 dan ke 5. Hal ini disebabkan karena kesalahan data pengukuran odometri yang berasal dari hasil pembacaan *encoder*. Kesalahan atau *error*

odometri juga dibuktikan dalam data pengujian derajat putar dan translasi sebelumnya. Untuk itu hal ini dapat diselesaikan dengan menggunakan *rotary encoder* yang mempunyai resolusi tinggi. Perlu diketahui dalam penerapan *Dynamic A** ini, gerakan robot yang dipakai untuk mendeteksi adanya penghalang di sekitar robot adalah dengan memutar robot dengan 360 putaran setiap *grid*. Hal ini dikarenakan tidak adanya sensor pada sisi kanan dan kiri robot. Jika robot kita tempatkan menghadap lurus pada posisi *start*, maka jangkauan sensor jarak hanya berfokus pada sisi depan robot, sedangkan robot harus mengetahui penghalang yang ada di sisi kanan dan kiri robot guna melakukan kembali perhitungan $f(n)$ yang baru dalam sistem *Dynamic A**.

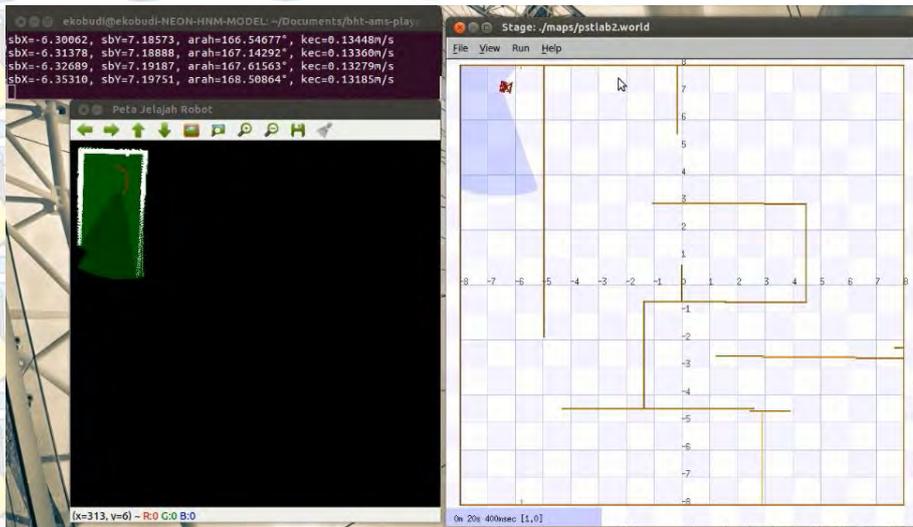
4.5 Pengujian *mapping* menggunakan Occupancy Grid Maps (simulasi)

Pengujian *mapping* secara simulasi ini dilakukan pada lapangan dengan pembagian *grid* menjadi 16 x 16 dengan koordinat pusat 0,0. Simulasi yang digunakan adalah player stage [17]. Robot bergerak dari kordinat awal -8,8 sampai ke titik finish 8,-8. Kemudian dalam pengujian ini menggunakan asumsi sensor ideal yang biasanya digunakan untuk proses SLAM. Sensor yang digunakan adalah *hokuyo* dengan lebar *beam* sebesar 205 derajat dan jarak pancar sejauh lebih dari 5m. Tapi dalam pengujian navigasi pada real robot menggunakan sensor ultrasonik dengan derajat *beam* maksimal adalah 15 derajat dan jarak pancar kurang dari 5m. Dapat diustrasikan dengan gambar dibawah ini:



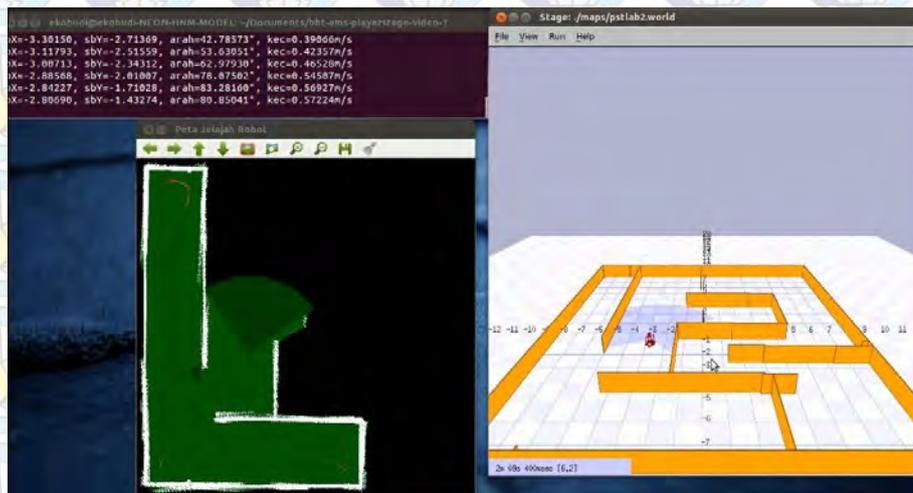
Gambar 4.11 Perbandingan lebar *beam* pada sensor simulasi (kiri) dan pada robot (kanan)

Kemudian untuk *scan* awal sensor dan hasil *mapping* ditunjukkan pada gambar 4.12.



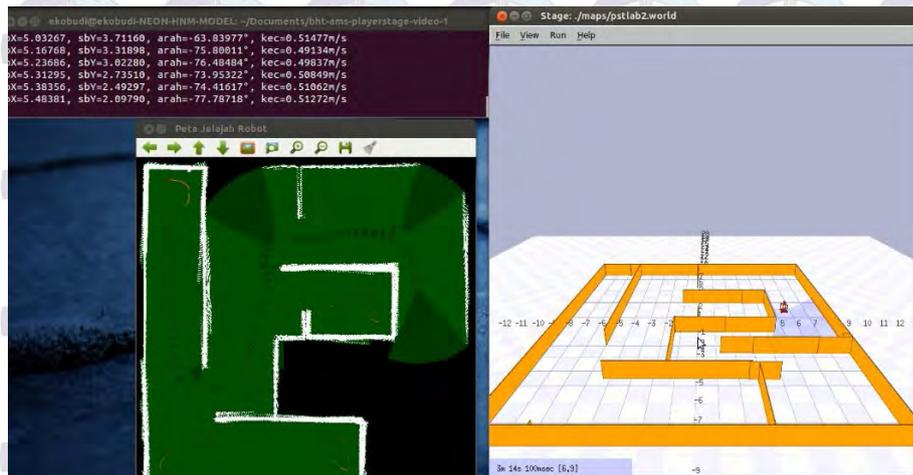
Gambar 4.12 Robot melakukan *scan* awal.

Robot bergerak dengan gerakan penghindar halangan terhadap penghalang yang ada menuju titik finish dengan pemandu dari sisi global yaitu menggunakan algoritma A*.



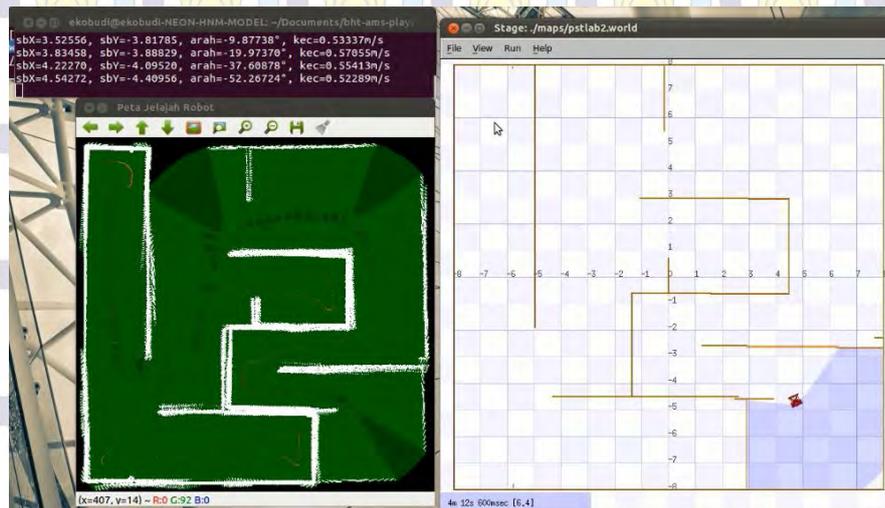
Gambar 4.13 Robot melakukan penelusuran.

Pada terminal ditampilkan data posisi, arah dan kecepatan robot saat menelusuri lingkungan. Hasil sementara peta lingkungan yang dijelajah robot ditunjukkan pada gambar 4.14.



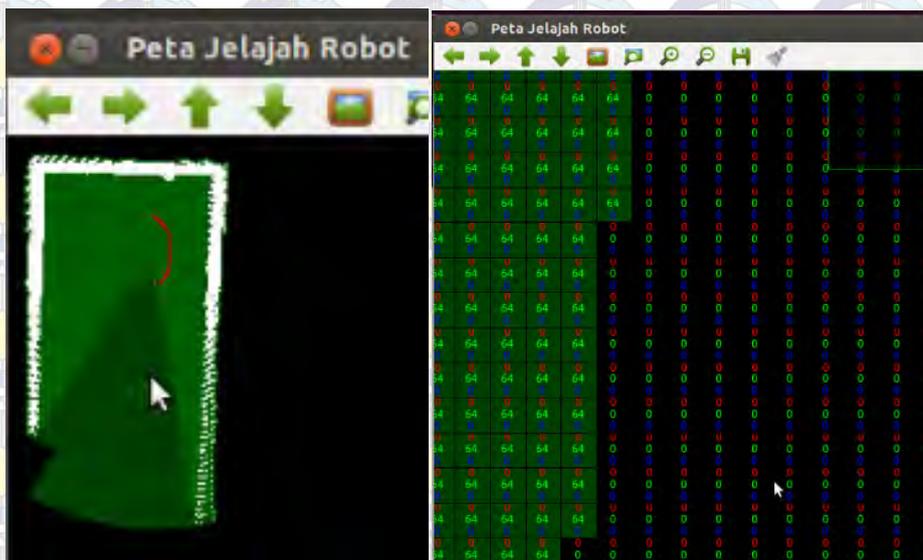
Gambar 4.14 Hasil *mapping* sementara

Pada saat robot mencapai garis *finish*, hasil pemetaan ditunjukkan pada gambar 4.15 dengan beberapa bagian masih berwarna hitam (tidak teroccupied). Hal ini dikarenakan robot melintasi *grid* dengan nilai $f(n)$ yang terkecil sehingga untuk *grid* yang tidak menjadi solusi jarak terpendek yang dilalui robot masih berwarna hijau.



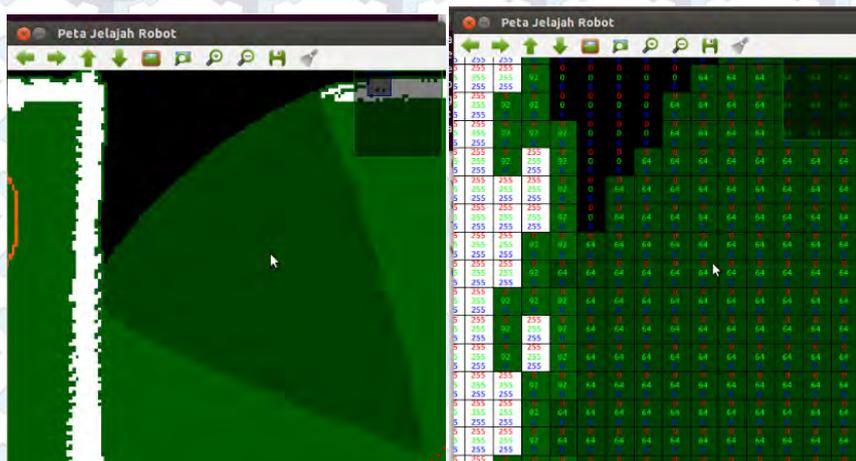
Gambar 4.15 Hasil *mapping* untuk warna hijau (tidak occupied) warna putih (occupied)

Selanjutnya untuk *detail* nilai *grid* setiap *cell* yang akan digunakan untuk menyatakan sebuah *grid* yang tidak teroccupied, diberi nilai 0. Untuk nilai sensor yang melintas di atas *grid* ditandai dengan warna hijau dengan derajat RGB sebesar 64.



Gambar 4.16 Nilai grid untuk peta yang tidak teroccupied (hitam) dan derajat probabilitas pengukuran sensor (warna hijau)

Untuk nilai batas jangkauan sensor akan ditandai sebagai *cell* yang sudah teroccupied dan diberikan nilai RGB 255, 255, 255 atau warna putih. Pada gambar 4.17 menunjukkan keadaan hasil *mapping* antara *grid* yang sudah *occupied*, belum *occupied* dan derajat jangkauan sensor.



Gambar 4.17 Nilai grid untuk peta yang teroccupied, tidak teroccupied dan penghalang

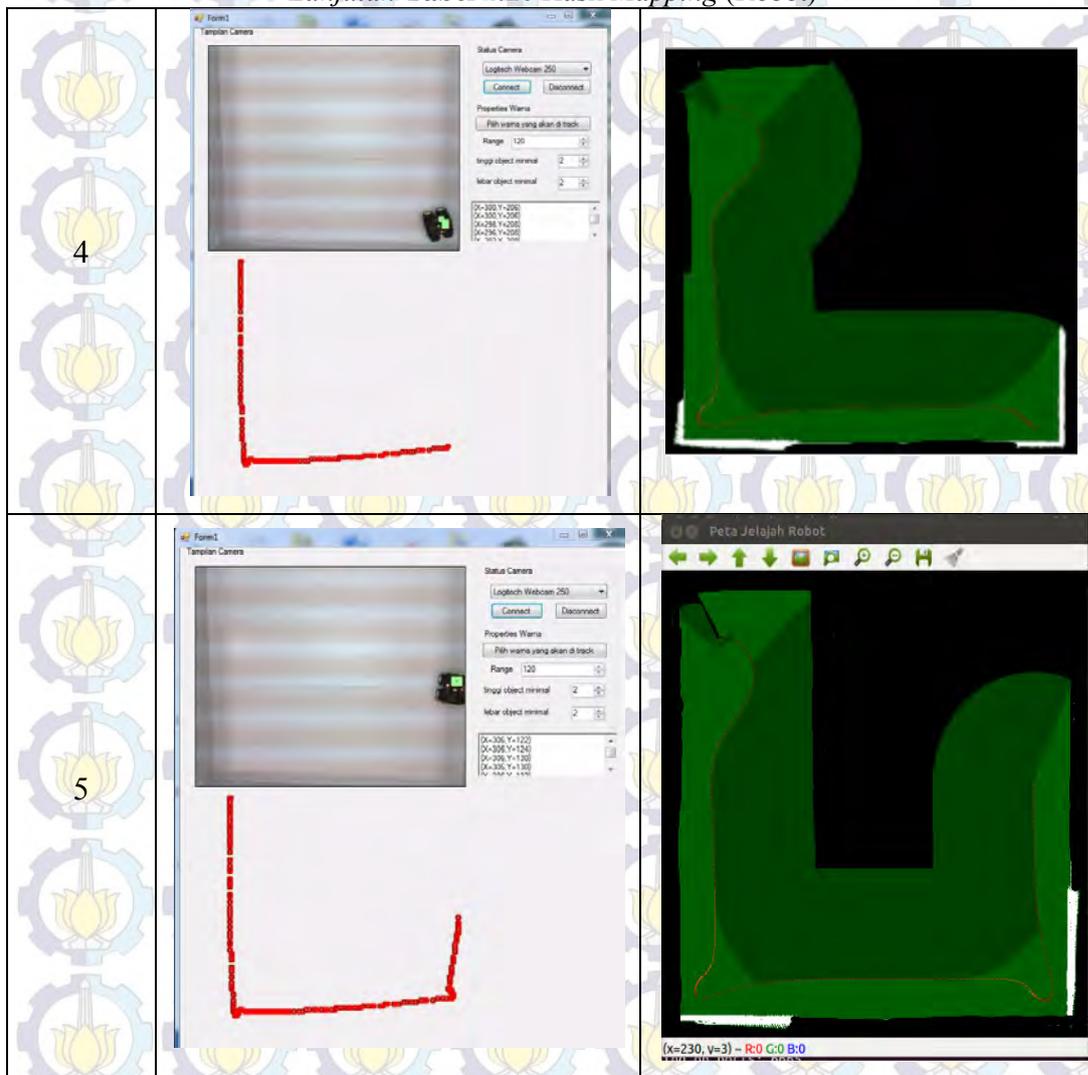
4.6 Pengujian *mapping* menggunakan Occupancy Grid Maps (Robot)

Untuk pengujian *mapping* pada robot menggunakan sensor ultrasonik dengan jangkauan kurang 5 mm dengan lebar *beam* sebesar 15 derajat. Berikut ini hasil *mapping* pada robot.

Tabel 4.9 Hasil Mapping (Robot)

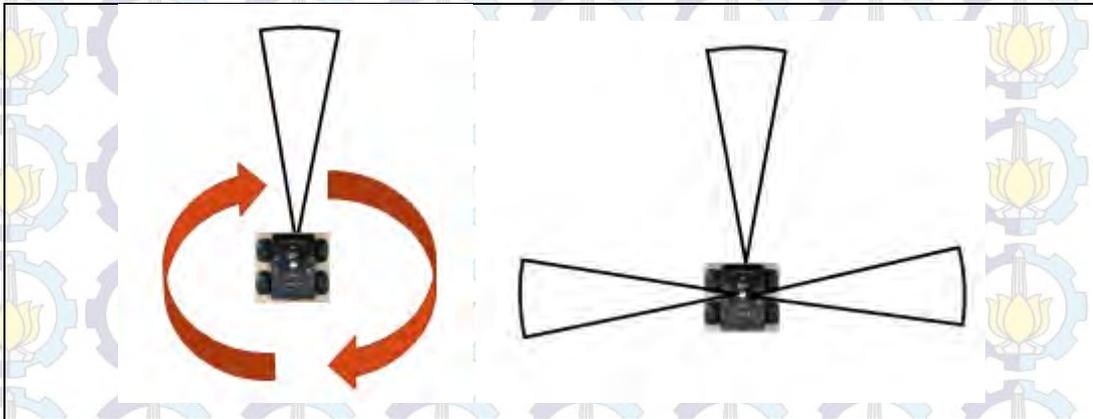
No	Data ke / Grid ke	Hasil Mapping
1		
2		
3		

Lanjutan Tabel 4.10 Hasil Mapping (Robot)



Dari hasil *mapping* menggunakan robot menunjukkan *cell occupied* (warna putih) tidak berada di tepi peta. Hanya berada di data pengujian ke 3, 4, dan 5. Hal ini disebabkan karena sensor ultrasonik pada robot hanya berada di bagian depan robot. Sedangkan di sisi kanan dan kiri pada badan robot tidak terdapat sensor ultrasonik. Oleh karena itu hasil pembacaan sensor ultrasonik ditunjukkan seperti pada hasil *mapping* diatas. Pergerakan robot pada pengujian tersebut adalah maju lurus tanpa adanya gerakan tambahan. Jika dibandingkan dengan hasil simulasi pemetaan yang dibahas sebelumnya,. Terdapat *cell* yang ter *occupied* pada setiap sisi peta. Ini dikarenakan simulasi tersebut menggunakan

persepsi sensor dengan derajat 205, sedangkan pada ultrasonik hanya 15 derajat. Untuk bisa membuat hasil *mapping* seperti pada simulasi, cara yang bisa dilakukan adalah dengan melakukan penambahan sensor ultrasonik disisi kanan dan kiri robot. Jika tidak diinginkan penambahan sensor maka diperlukan gerakan berputar untuk setiap gerakan maju yang dilakukan.



Gambar 4.18 Solusi gerakan berputar (kiri) dan penambahan sensor (kanan)

BAB 5 KESIMPULAN

5.1 Kesimpulan

Dari hasil implementasi dan pengujian yang dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. Penerapan unsur pembangun SLAM pada HBE robocar memerlukan penyesuaian untuk SLAM ideal dengan tambahan
 1. Sensor jarak pada sisi kanan dan kiri robot untuk :
 - a. Kasus navigasi : mendeteksi penghalang saat nilai $f(n)$ pada dinamik A* dijalankan
 - b. Kasus pemetaan : menjadikan penghalang yang ada di kanan kiri robot sebagai *cell occupied*
 - c. Kasus penentuan posisi : membentuk persepsi yang digunakan *sensing model* dalam menghitung *pose* relatif robot
 2. Pemakaian kamera atas (camera global) untuk *global reference*
 3. Penambahan sensor kompas untuk data orientasi robot.
2. Solusi agar HBE Robocar dapat digunakan untuk navigasi, penentuan posisi dan pemetaan adalah dengan membuat gerakan berputar di tempat untuk mendapatkan data jarak disekitar robot.
3. Semakin kecil jumlah partikel, diperlukan iterasi yang lebih banyak dalam mencapai konvergen partikel (estimasi *pose* (x, y, θ))
4. Semakin banyak jumlah partikel, waktu komputasi yang diperlukan semakin lama.
5. Hasil konvergen partikel terbaik selama pengujian yaitu pada iterasi ke 42 dengan jumlah partikel yang disebar adalah 400. Dengan rata-rata *error* untuk nilai $x = 3.41\%$, $y = 4.03\%$, dan $\theta = 65.79\%$
6. Penggunaan algoritma A* dapat membantu robot menemukan jalur terpendek dengan *error pose* robot untuk data arah sebesar 4.34 % dan *error* jarak tempuh robot sebesar 4.8 %.

5.2 Penelitian Selanjutnya

Dengan ditambahkan penyesuaian berupa penambahan sensor jarak pada sisi kiri dan kanan robot atau dengan solusi yang diberikan berupa gerakan berputar maka *HBE Robocar* maka dapat digunakan untuk membangun SLAM (*Simultaneous Localization And Mapping*). Dalam SLAM, robot diletakkan atau *running* pada lingkungan yang belum diketahui. Diletakkan di arena atau lingkungan yang baru. Kemudian peta dari lingkungan tersebut juga tidak diberitahukan pada robot. Berlaku penentuan posisi terhadap pembacaan lingkungan yang *partial* (sebagian). Dari hasil yang *partial* tersebut maka robot membuat sistem persepsi baik secara visual kamera, *ultrasonic perception* atau sensor-sensor lain yang dapat digunakan sebagai sumber informasi untuk mengestimasi posisi robot tersebut.

DAFTAR PUSTAKA

- [1] J. Leonard and H.Durrant Whyte. "Simultaneous map building and Localization for an autonomous mobile robot", IEEE. Workshop on Intelligent Robots and Systems, pp. 1442-1447, 1991.
- [2] Leonard J. J., Durrant-whyte F. H. "Mobile robot localization by tracking geometric beacons[J]". IEEE Trans on Robotics and Automation, 1991, 7(3): 376-382.
- [3] Fox D., Burgard W., Dellaert F. "Monte carlo localization: efficient position estimation for mobile robots[c]". Proceedings of the National Conference on Artificial Intelligence. 1999, 343-349.
- [4] Wang H., Liu H. Y., Ju H. H., Li X. Z. "Improvement for the rao-blackwellized particle filters SLAM with MCMC resampling [C]". International Conference on Computational Intelligence and Software Engineering. 2009.
- [5] Sheng Fu, Hui ying Liu, Lu-fang Gao, Yu-xian Gai. "SLAM For Mobile Robots using Laser Range Finder and Monocular Vision". IEEE. 2007.
- [6] Sebastian Zug., Felix Penzlin., Andre Dietrich., Tran T.N."Are Laser Scanners replaceable by kinect sensor in robotic applications". IEEE. 2012.
- [7] Ren C.Luo, Wei-Lung Hsu. "Autonomous Mobile Robot Localization Based On Multisensor Fusion Approach". IEEE. 2012.
- [8] Tauseef Gulrez,Rami Al-Hmouz,Zenon Chaczko. "Unmanned Autonomous Vehicle Control &SLAM Problem in 2D environment". IEEE. 2004.
- [9] Yusuke Misono, Yoshitaka Goto, Yuki Tarutoko, Kazuyuki Kobayasi. "Development of Laser Rangefinder-based SLAM algorithm for mobile robot Navigation.". IEEE. 2007.
- [10] Yusuke Misono, Yoshitaka Goto, Yuki Tarutoko, Kazuyuki Kobayasi. "VorSLAM: A New Solution to Simultaneous Localization and Mapping.". Proceeding of the International Conference on information and automation.IEEE. June, 2007.

- [11] Thrun, S., Burgard W. dan Fox, D. (2005) : Probabilistic Robotics, The MIT Press, Cambridge, England
- [12] Jie Li, Lei Cheng, Huaiyu Wu, Ling Xiong, Dongmei Wang, "An Overview of the Simultaneous Localization and Mapping on Mobile Robot," Proceedings of 2012 International Conference on Modelling, Identification and Control, Wuhan, China, June 24-26, 2012.
- [13] <http://robots.stanford.edu/papers/thrun.occ-journal.html> (diakses pada tanggal 6 januari 2014 pukul 21.31 BBWI)
- [14] Manual Book of HBE Robocar and Robocar Embedded.2007
- [15] Demetris Stavrou, Christos panayiotou. "Localization of a simple robot with low computational - power using a single short range sensor" Proceedings of International Conference on Robotics & biomimetics December 11-14, 2012.
- [16] Philipp Reit CarSim- a simple simulator udacity , <http://forums.udacity.com/questions/1013970/carsim-a-simple-simulator>
- [17] Richard T. Vaughan, Brian P. Gerkey, and Andrew Howard, "The Player/Stage Project: Tools for Multi-Robot". Proceeding of the Intelligent Conference on Advanced Robotics (ICAR), Portugal, July 2003.
- [18] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. "Particle filters for mobile robot localization". *Sequential Monte Carlo Methods in Practice*, pages 499-516. Springer Verlag, 2001.
- [19] Dae hee Won., Young Jae lee, Sangkyung Sung, Taesam Kang. "Integration of vision based SLAM and Nonlinear Filter for simple Mobile Robot Navigation," 2008.
- [20] Jianming G., liang liu, Qing Liu,"An improvement of D* for mobile robot path planning in partial unknown environment". Proceedings of International Conference on Intelligent Computation Technology, 2009.
- [21] Pearsall, J., ed. Concise Oxford Dictionary. Tenth Edition, Revised ed. 2001, Oxford University Press.
- [22] Wong Yuen Loong, Liew Zhen Long and Lim Chot Hun, "A Star Path Following Mobile Robot", 2011 4th International Conference on Mechatronics (ICOM), May 2011.

BIOGRAFI PENULIS



Eko Budi Utomo. Anak pertama dari pasangan Sugiarto dan Nanik Rachmawati, lahir pada tanggal 20 Mei 1990 di Banyuwangi, Jawa Timur. Memulai pendidikan tahun 1995 di TK Aisyiyah II Banyuwangi. Tahun 1996 melanjutkan di SDN.Tukang Kayu II Banyuwangi. Tahun 2002 melanjutkan di SMPN 1 Banyuwangi, tahun 2005 penulis melanjutkan studi di Jurusan Teknik Audio Video SMKN 1 Glagah Banyuwangi. Setelah lulus tahun 2008, penulis melanjutkan studi di D4 Teknik Elektronika PENS (Politeknik Elektronika Negeri Surabaya) ITS melalui jalur PMDK Berprestasi dan lulus pada tahun 2012 dengan gelar Sarjana Sains Terapan. Setelah itu tahun 2012 penulis melanjutkan studi di Magister Bidang Studi Jaringan Cerdas Multimedia - Konsentrasi Teknologi Permainan Institut Teknologi Sepuluh Nopember Surabaya. Penulis telah melaksanakan ujian tesis pada bulan Januari 2015. Alamat email penulis yang bisa dihubungi : ekobudi_u@pens.ac.id atau di eko.pensrobotics@gmail.com