IMPLEMENTASI *HECTOR SLAM* PADA ROBOT PENCARI KORBAN GEMPA

THE IMPLEMENTATION OF HECTOR SLAM ON THE EARTHQUAKE VICTIMS FINDER ROBOT

Muliady[1], Erwani Merry Sartika[2], Kevin Reynaldo M[3]

[1,2,3] Jurusan Teknik Elektro
Universitas Kristen Maranatha
Jl. Suria Sumantri 65 Bandung , 022-2012186 ext 1239

Abstrak

Indonesia merupakan wilayah dimana gempa bumi banyak terjadi dan menelan banyak korban jiwa. Korban jiwa disebabkan karena terjebak atau terhimpit reruntuhan atau struktur bangunan dan tidak segera ditemukan. Oleh karena itu, diperlukan robot untuk menelusuri reruntuhan gempa dan juga memetakan kondisi medan yang terkena gempa, sehingga dapat memercepat proses pencarian korban yang terjebak di reruntuhan bangunan dan sekaligus tidak membahayakan keselamatan anggota tim SAR. Dalam penelitian ini, robot didesain menggunakan *controller* FitPC (Intel atom Z530), sensor Hokuyo URG04-LX 2D *laser scanner*, robot *rescue all-terrain*, OS *Ubuntu* 12.04, dan *Robot Operating System* (ROS) *Fuerte*. Untuk melakukan *localization* dan *mapping* digunakan metode *Hector-slam*. Proses pengolahan data dari sensor *laser scanner* akan dilakukan di Fit PC2i, sementara *Arduino Uno* digunakan sebagai antarmuka antara FitPC dengan *motor driver*. Berdasarkan parameter yang didapat dari sensor *laser scanner*, robot akan dapat bergerak secara *autonomous* tanpa menabrak rintangan dan secara bersamaan dapat melakukan pemetaan kondisi sekitar kemudian menampilkannya dalam bentuk dua dimensi. Dari hasil percobaan yang dilakukan diperoleh bahwa *localization* dan *mapping* secara *autonomous* telah berhasil diimplementasikan dengan menggunakan Hokuyo URG04-LX 2D *laser scanner*, FitPC2i, *Arduino Uno* dan Robot *Rescue All-Terrain*.

Keywords: *ROS, FitPC2i, Hokuyo URG04-LX, Arduino Uno, Hector-SLAM*

Abstract

Indonesia is a region where many earthquakes take place inflicting heavy casualties. Getting buried in the debris, stuck in building ruins, and too late for rescue are causes for the increase of death toll. Robots are required to explore the earthquake ruins and chart the terrain affected by the earthquake so that the search for survivors can be speeded up at the same time maintain the rescue team members' safety. In this study, the robot is designed by using FitPC controller (Intel atom Z530), sensor - LX URG04 Hokuyo 2D laser scanner, all- terrain rescue robot, OS Ubuntu 12:04, and ROS (Robot Operating System) Fuerte. The experiment's result show that autonomous localization and mapping can be successfully implemented by using URG04 - LX Hokuyo 2D laser scanner, FitPC2i, Arduino Uno, and the All-Terrain Robot Rescue .

Keywords: *ROS , FitPC2i , Hokuyo URG04 - LX , Arduino Uno , Hector - SLAM*

Tanggal Terima Naskah : 05 Juni 2015
Tanggal Persetujuan Naskah : 29 Juli 2015

1. PENDAHULUAN

Saat ini teknologi semakin berkembang pesat, salah satunya adalah teknologi robotika. Robot mengalami perkembangan pesat, baik *hardware* maupun *software*. Perkembangan tersebut membuat robot semakin canggih. Robot yang semula bergerak hanya menggunakan *remote control*, saat ini sudah mampu bergerak secara *autonomous* tanpa perlu bantuan manusia. Agar robot dapat bergerak secara *autonomous*, perlu diciptakan sistem yang membuat robot dapat bergerak secara bebas di dalam lingkungan yang tidak diketahui dan menghindari rintangan secara otomatis. Kunci dari hal ini adalah agar robot dapat memetakan kondisi lingkungan sekitar dan mengidentifikasi posisi robot dalam peta. Solusi permasalahan ini dirangkum dalam *Simultaneous Localization and Mapping* (SLAM). SLAM adalah proses dimana sebuah robot dapat membuat sebuah peta berdasarkan sensor yang ada dan secara bersamaan mengidentifikasi posisinya dalam peta.

Berdasarkan hal tersebut, dalam penelitian ini dibutuhkan sebuah robot yang mampu mengimplementasikan SLAM. Untuk realisasinya, digunakan robot *rescue all-terrain* dan sensor *laser scanner*. Sensor ini dipilih karena keakuratan dan banyaknya data jarak yang mampu diolah dalam selang waktu yang singkat. Untuk membantu proses SLAM ini digunakan *Robot Operating System* (ROS) sebagai pemroses data dan media komunikasi agar dapat menghubungkan sensor dengan robot, sehingga diharapkan robot dapat membuat sebuah peta dan melokalisasi posisinya dalam peta secara *autonomous*.

2. *Simultaneous Localization and Mapping*

Simultaneous localization and mapping (SLAM) adalah teknik yang digunakan untuk mengkonstruksi sebuah peta dari lingkungan yang tidak diketahui (atau memperbarui sebuah peta dalam lingkungan yang sudah diketahui sebelumnya) dan secara bersamaan mengetahui posisi mesin dalam lingkungan fisik. *Mapping* adalah permasalahan dalam mengintegrasikan informasi yang didapatkan oleh sekumpulan sensor menjadi sebuah model yang konsisten dan menggambarkan informasi tersebut dalam sebuah representasi. *Localization* adalah permasalahan dalam mengestimasi posisi (dan pose) dari robot terhadap sebuah peta.

SLAM berkaitan dengan masalah membangun peta lingkungan yang tidak diketahui oleh robot, sementara pada saat yang sama menavigasi lingkungan menggunakan peta. Untuk melakukan SLAM dibutuhkan *mobile robot* dan perangkat pengukuran jangkauan. Perangkat pengukuran jangkauan yang digunakan antara lain *laser scanner*, sonar, dan lain-lain [1].

Hector merupakan singkatan dari *Heterogeneous Cooperating Team of Robots*. Algoritma ini dibuat sebagai modul yang bersifat *open source*, yang menyediakan blok sistem yang mampu melakukan eksplorasi secara *autonomous* dalam lingkungan *Urban Search and Rescue* (USAR) [2]. ROS digunakan sebagai perantara untuk modul *software* yang digunakan [3]. Untuk melakukan proses *localization* dan *mapping* digunakan algoritma *Adaptive Monte Carlo* (AMCL) dan *Occupancy Grid Mapping*.

2.1 *Adaptive Monte Carlo Localization* (AMCL)

Adaptive Monte Carlo Localization yang dikenal juga sebagai lokalisasi partikel *filter* adalah algoritma untuk robot agar dapat melokalisasi dengan menggunakan partikel *filter*. Algoritma ini memperkirakan posisi dan orientasi dari robot ketika bergerak dan mendeteksi lingkungan sekitar. Tiap partikel merepresentasikan posisi robot yang mungkin (hipotesis posisi robot). Partikel ini merupakan hasil *sampling* yang telah diberikan bobot.

MCL terdiri atas dua fasa, yaitu fasa prediksi dan fasa *update*. Pada fasa prediksi, model gerakan robot akan diaplikasikan kepada seluruh *sample* sehingga pergerakan yang dilakukan oleh robot sama dengan pergerakan yang dilakukan oleh *sample* pada peta. Pada fasa *update*, tingkat keyakinan robot tentang posisi dan orientasi diperbaharui dengan menggunakan hasil bacaan sensor. Setelah itu, dilakukan proses *resampling* untuk menghasilkan kumpulan *sample* baru berdasarkan bobot dari kumpulan *sample* yang lama. Tahap terakhir adalah tahap ekstraksi hasil estimasi posisi dan orientasi robot.

Algoritma ini bermula dengan penyebaran partikel secara acak. Hal ini berarti robot tidak memiliki informasi posisi robot dan mengasumsikan bahwa robot bisa berada dimana saja di dalam ruang yang ada. Ketika robot bergerak, partikel yang ada ikut bergerak untuk memprediksi keadaan yang baru setelah terjadi pergerakan robot. Ketika robot mendeteksi sesuatu, partikel ini akan di *resampling* berdasarkan pada estimasi Bayes. Estimasi Bayes menggabungkan informasi yang dikandung dalam *sample* dengan infomasi lain yang telah tersedia sebelumnya, hingga pada akhirnya partikel akan berkumpul pada posisi robot yang sebenarnya [4].

2.2 *Occupancy Grid Mapping*

Algoritma ini telah digunakan oleh banyak robot *autonomous*. Aplikasi *occupancy grid map* yang paling baik sejauh ini menggunakan sensor jarak, seperti sensor sonar dan *laser range finder*. Kedua sensor dipengaruhi oleh *noise*. Sonar mencakup area berbentuk kerucut di depannya, dari pengukuran satu sonar adalah tidak mungkin untuk mengatakan dimana posisi objek dalam kerucut tersebut. Kedua sensor sensitif terhadap sudut dari permukaan objek relatif terhadap sensor dan pemantulan dari permukaan (penyerapan dan pemancaran).

Occupancy grid map memecahkan masalah tersebut dengan menghasilkan *map* probabilitas. Seperti namanya, *occupancy grid map* direpresentasikan oleh *grid*, berbentuk dua dimensi. Algoritma *occupancy grid map* yang standar adalah versi dari Bayes *filter*, seperti algoritma *mapping* pada umumnya. Bayes *filter* digunakan untuk mengkalkulasikan *occupancy* dari tiap *grid cell*.

Occupancy grid map mudah untuk digunakan, terbukti dari banyaknya penggunaan. Kekurangannya antara lain ketidakmampuan untuk menghasilkan ketidakpastian *pose* dan ketika ada *noise* sensor yang dikorelasikan dengan data dari sensor, maka *map* yang dihasilkan akan menjadi *error* [5].

3. PERANCANGAN DAN REALISASI

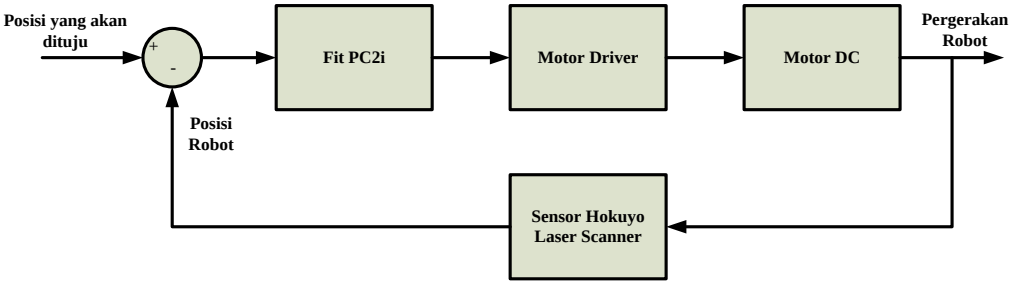
Proses diawali dengan perancangan robot *rescue all-terrain*, kemudian dilanjutkan dengan realisasinya.

3.1 Perancangan Sistem Robot *Rescue All-Terrain*

Robot yang digunakan untuk implementasi *localization* dan *mapping* adalah robot *Rescue All Terrain*. Robot ini dirancang untuk dapat melewati rintangan dalam medan gempa, menggunakan roda berbentuk *caterpillar*, yang terbuat dari bahan karet dengan ukuran 5 cm x 27 cm yang dapat bergerak bebas 360^{o}, bergerak maju dan mundur sesuai dengan medan yang dilalui. *Body* robot bagian dasar terbuat dari *allumunium* dengan tebal 8 mm. *Body* atas robot menggunakan plat *alluminium* setebal 2,5m. Gambar 1. menunjukkan realisasi Robot *Rescue All Terrain* yang digunakan untuk *localization* dan *mapping* [6].

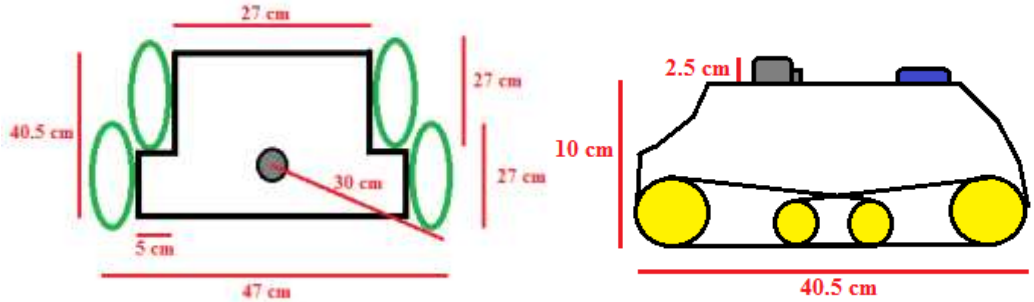
Gambar 1. Robot *rescue all-terrain*

Agar robot *Rescue All Terrain* mampu melakukan *localization* dan *mapping*, maka ditambahkan sensor Hokuyo *laser scanner* [7] pada robot tersebut. Sensor *laser scanner* tersebut harus diposisikan pada bagian yang tidak terhalang dan pada bagian tengah robot. Hal ini karena posisi *laser scanner* akan merepresentasikan pose dari robot *rescue all-terrain*. Fit PC2i dipilih sebagai *controller* dalam penelitian ini, yaitu *controller* yang menggunakan OS Linux Ubuntu 12.04 dan ROS Fuerte. Gambar 2 menunjukkan diagram blok sistem kontrol robot *rescue all-terrain* menggunakan *laser scanner* sebagai sensornya.



Gambar 2. Diagram blok sistem kontrol robot *rescue all-terrain*

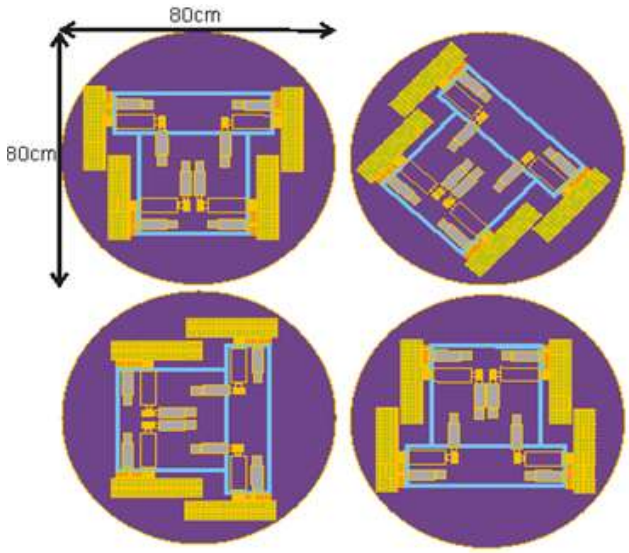
Gambar 2 menunjukkan diagram blok sistem kontrol robot *rescue all-terrain*. *Laser scanner* melakukan *scanning* area sekitar kemudian mengirim data (berupa informasi jarak) ke Fit PC2i. Data tersebut akan diproses oleh Fit PC2i, untuk menganalisis ada atau tidaknya halangan, kemudian Fit PC2i akan mengirimkan sinyal PWM ke *motor driver* untuk menggerakkan motor DC sehingga robot dapat menghindari halangan.



Gambar 3. Desain posisi *laser scanner* pada robot *rescue all-terrain*

Gambar 3 adalah desain perancangan posisi sensor *laser scanner*, warna abu-abu adalah Hokuyo *laser scanner*, sedangkan warna biru adalah *controller* Fit PC2i. Dalam perancangannya, robot dapat menghindari halangan berdasarkan bacaan dari sensor *laser scanner*. Untuk dapat bermanuver dengan baik robot membutuhkan ruang berbentuk

lingkaran dengan diameter ±80 cm. Bentuk area yang dibutuhkan robot *rescue all-terrain* ditunjukkan pada Gambar 4 [8].

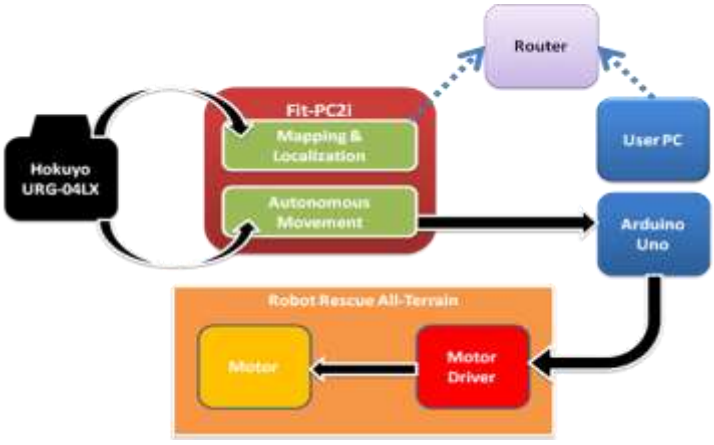


Gambar 4. Area *manuver* robot *rescue all-terrain*

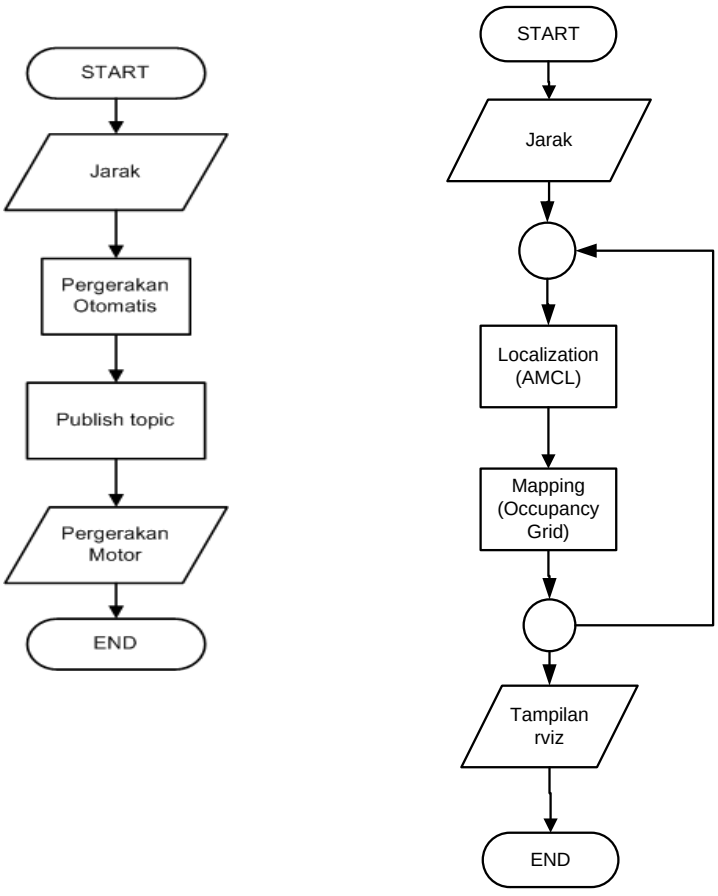
Sesuai Gambar 4, karena diameter area yang dibutuhkan adalah 80 cm, hal ini berarti jari-jari area manuver adalah 40 cm. Karena jari-jari area manuver adalah 40 cm maka batas pembacaan sensor untuk menghindari rintangan diatur 50 cm. Nilai 10 cm ditambahkan untuk memberi toleransi apabila waktu respon robot terlambat.

3.2 Realisasi Sistem Robot *Rescue All-Terrain*

Pada Gambar 5 ditunjukkan mengenai diagram blok sistem secara keseluruhan. Pada awalnya sensor *laser scanner* hokuyo URG-04LX akan mengirimkan data ke Fit PC2i menggunakan koneksi USB. FitPC2i akan melakukan dua proses, yaitu *mapping* dan *localization*, serta *autonomous movement*. Kedua proses ini dilakukan oleh ROS menggunakan *topic* sebagai media komunikasi. Hasil *mapping* dan *localization* ini akan ditampilkan pada monitor di *user* PC secara *wireless* menggunakan perantara *router*. Setelah data *laser scanner* diproses, kemudian Fit PC2i akan mengirimkan data ke *Arduino Uno* untuk pergerakan otomatisnya. *Arduino* menerima data tersebut kemudian mengirimkan sinyal PWM ke *motor driver*, setelah itu *motor driver* akan menggerakkan motor.



Gambar 5. Diagram blok sistem robot *rescue all-terrain*



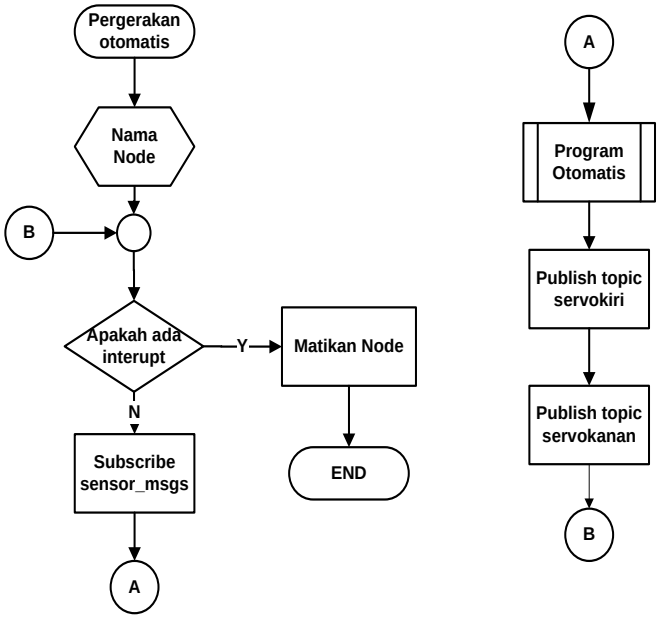
(a) (b)

Gambar 6. Diagram alir sistem robot *Rescue All-Terrain*

Untuk diagram alir *autonomous movement* seperti ditunjukkan Gambar 6 (a), pada awalnya data dari sensor *laser scanner* berupa jarak akan diolah di dalam *node* yang mengeksekusi pergerakan *autonomous*. *Node* tersebut kemudian akan mem-*publish topic*, selanjutnya *topic* tersebut akan di-*subscribe* oleh *node* untuk menggerakkan motor. Akhirnya akan didapatkan pergerakan motor agar robot dapat bergerak secara *autonomous*.

Untuk diagram alir *mapping* dan *localization* seperti ditunjukkan Gambar 6 (b), data dari *laser scanner* berupa jarak akan diproses di dalam *node* yang berfungsi untuk melokalisasi posisi robot yang disebut *pose estimation*. Di dalam *node* tersebut akan diproses lokalisasi robot menggunakan metode *Adaptive Monte Carlo Localization* (AMCL), sehingga akan didapatkan posisi robot di dalam peta. Setelah didapatkan lokalisasi dari robot tersebut, posisi robot akan dimasukkan ke dalam *node* untuk *mapping*. *Node mapping* ini menggunakan metode *Occupancy Grid* untuk mendapatkan hasil *mapping* dari kondisi lingkungan sekitar robot dan akhirnya hasilnya akan ditampilkan dalam rviz.

Gambar 7 menjelaskan mengenai pengiriman *topic* dari *node* pergerakan *autonomous*. Pada awalnya nama *node* akan diinisialisasi, kemudian jika tidak ada *error* atau *interrupt*, maka *node* tersebut akan melakukan proses *subscribe topic* sensor_msgs, sedangkan jika salah, maka *node* tersebut akan berhenti. Setelah *topic* tersebut diterima, maka akan dipanggil program otomatis untuk melakukan proses *publish topic* servo kiri dan servo kana.[9].



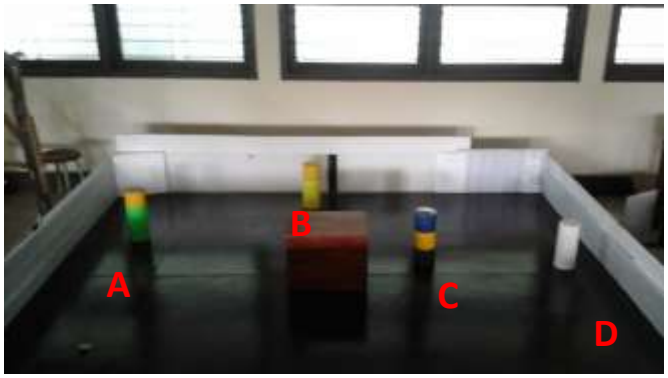
Gambar 7. Diagram alir pergerakan otomatis

4. DATA PENGAMATAN DAN ANALISIS

Bab ini berisi data pengamatan dan analisis data yang telah diperoleh dari percobaan *localization* dan *mapping* secara manual dan *autonomous*.

4.1 Uji Coba *Localization*

Percobaan *localization* dilakukan sebanyak lima kali pada area berbentuk kotak dan diberikan empat buah benda yang ditunjukkan huruf A, B, C, D pada Gambar 8. Percobaan *localization* dilakukan secara manual menggunakan *remote control*.

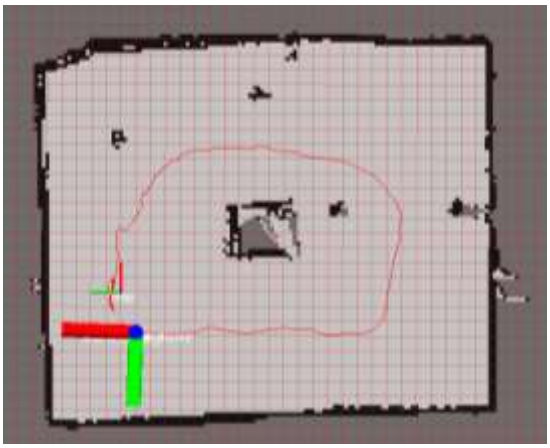


Gambar 8. Area percobaan *localization*

Robot melakukan satu kali putaran mengelilingi area percobaan secara manual untuk membentuk satu peta yang utuh, kemudian dilakukan lima kali percobaan untuk mengetahui lokasi robot dan hasilnya dibandingkan dengan lokasi pada area yang sebenarnya. Tabel 1 menunjukkan posisi benda pada area sebenarnya dan dalam peta setelah robot mengelilingi area uji sebanyak satu kali. Dari hasil yang didapat menunjukkan nilai koordinat dalam peta mendekati nilai koordinat dalam area yang sebenarnya.

Tabel 1. Tabel pengamatan *localization* setelah dilakukan satu putaran

Benda	Posisi(cm)		Posisi dalam Peta(cm)	
	X	Y	x	y
Titik Kuning	0	0	0	0
Huruf A	49	194	50	193,33
Huruf B	148	225,5	150	226,67
Huruf C	205	142	203,36	140,9
Huruf D	289	142	287,3	140,9

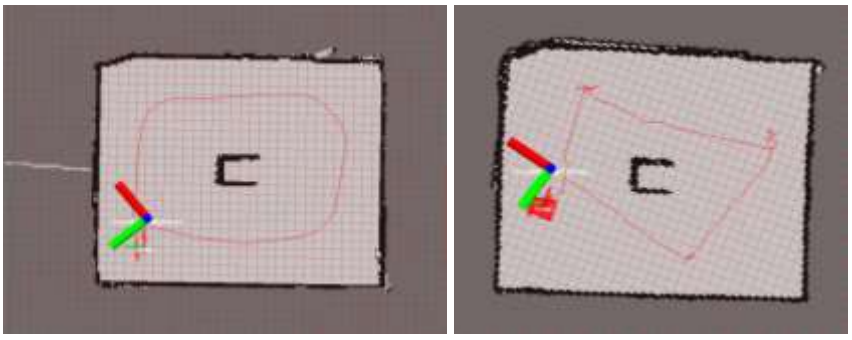


Gambar 9. Percobaan *localization* yang pertama

Untuk pengamatan *localization* setelah lima kali percobaan menunjukkan hasil *localization* dalam peta yang mirip dengan kondisi sebenarnya, robot harus mendekati objek atau melewati objek lebih dari satu kali untuk mendapatkan hasil lokalisasi yang lebih baik [9].

4.2 Uji Coba *Mapping*

Percobaan untuk menguji hasil *mapping* dilakukan dalam dua metode yang berbeda, yaitu pergerakan robot secara *manual* dan *autonomous*. Hasil *mapping* dibagi dalam dua model. Gambar 10 (a) adalah gambar hasil *mapping* secara *manual*, sedangkan Gambar 10 (b) adalah gambar hasil *mapping* secara *autonomous*.



(a) (b)

Gambar 10. Percobaan *mapping*

Pada percobaan pergerakan secara *manual*, hasil percobaan dipengaruhi oleh orang yang mengontrol pergerakan robot untuk membentuk peta sesuai dengan yang sebenarnya, diperlukan gerakan robot yang cukup stabil. Untuk hasil percobaan pergerakan yang *autonomous*, pergerakan robot yang tidak terlalu banyak membelok

membuat peta yang dihasilkan lebih baik. Hal ini menandakan bahwa untuk mendapatkan hasil peta yang akurat, diperlukan pergerakan robot yang cukup stabil [9].

5. KESIMPULAN

Berdasarkan hasil percobaan dan data yang diperoleh, dapat disimpulkan bahwa pengaplikasian ROS untuk mengendalikan robot *rescue all-terrain* berhasil dilakukan menggunakan 2D *Laser Scanner localization* dan *mapping* pada robot *rescue all-terrain* telah berhasil diimplementasikan, serta robot telah berhasil menghindari rintangan secara *autonomous*. Beberapa hal yang dapat disimpulkan adalah sebagai berikut:

1. Untuk mendapatkan *map* yang sesuai dengan yang area sebenarnya diperlukan pergerakan robot yang stabil
2. Untuk mendapatkan hasil *localization* yang lebih akurat, sensor harus mendeteksi benda lebih dari satu kali atau mendekati benda tersebut.
3. Dari hasil percobaan disimpulkan bahwa untuk mendapatkan hasil *mapping* yang lebih baik, pergerakan robot diusahakan tidak terlalu banyak berbelok. Untuk mendapatkan hasil *mapping* yang optimal, kecepatan robot diatur sebesar ±9 m/menit.

REFERENSI

[1]. Lee, Yong Jae. Sung, Sangkyung. 2010. Vision Based SLAM for Mobile Robot Navigation Using Distributed Filters. Intech.
[2]. Kohlbrecher, Stefan., Et al. Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robot. Germany: TU Darmstadt.
[3]. Martinez,Aaron., Fernandez,Enrique. 2013. Learning ROS for Robotics Programming. Packt Publishing Ltd.: Birmingham.
[4]. Fox, Dieter., et. Al. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. Pittsburgh.
[5]. Thrun, Sebastian. 2002. Robotic Mapping: A Survey. Pittsburgh.
[6]. Tjahyadi, Hendra. 2013. Robot Penyelamat Korban Gempa. Penelitian: Bandung.
[7]. Kneip, Laurent et al. Characterization of the Compact Hokuyo URG-04LX 2D Laser Range Scanner. Autonomous System Laboratory, ETH Zurich.
[8]. D.A. Gunadi. Realisasi Robot Rescue All Terrain Mengacu pada Robocup Rescue Robot League Competition. Jurusan Teknik Elektro UKM, Bandung, Laporan Tugas Akhir, Maret 2015
[9]. K. Reinaldo. Implementasi *Localization & Mapping* menggunakan 2D *Laser Scanner* pada Robot Rescue All-Terrain. Jurusan Teknik Elektro UKM, Bandung, Laporan Tugas Akhir, Januari 2015