IMPLEMENTATION AND TESTING MONTE CARLO LOCALIZATION METHOD ON V-REP SIMULATOR

Tony Purba

STT Sapta Taruna, Jakarta Timur, Indonesia
E-mail: tony.purba@gmail.com

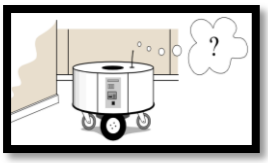
Abstract

For a robot in order to be able to do navigation in a closed environment requires a way to find out information from the robot's location itself, this is often referred to as localization. Among the many approaches and methods in solving localization problems, there is an approach that involves probabilistic elements in the field of robotics known as robotics probabilistic. One of the techniques for solving the localization of the robot is to use the Monte Carlo Localization (MCL) technique which is based on the particle filter technique. The MCL technique represents the robot's confidence level with a number of sample weights (particles), with posterior probability estimates derived from the Bayesian formulation of the localization problem. Considering that the easier way to implement the model and concept of robots is through existing robot simulators, one of which is the V-REP simulator. It is also necessary to implement and implement a probabilistic robotics model through a simulator, and one of the techniques implemented in this research is the MCL technique.

Keywords: *localization, probabilistic robotics, monte carlo localization, particle filter*

1. Pendahuluan

Di dalam bidang displin ilmu Robotika, pada dasarnya terbagi menjadi beberapa aspek yaitu Locomotion, Kinematics, Perception, Localization, dan Navigation serta Planning (Siegwart & Nourbakhsh, 2004). Pertama adalah aspek Locomotion merupakan aspek yang terkait dengan kemampuan robot dalam mekanisme lokomosi atau aktuator geraknya yang memungkinkan robot bergerak menjelajahi lingkungannya. Kedua adalah aspek Kinematics merupakan aspek yang terkait dengan kemampuan robot dalam memodelkan bagaimana perilaku dari pergerakan mekanikal robot tersebut. Keempat adalah aspek preception merupakan aspek yang terkait dalam kemampuan robot dalam memperoleh informasi terkait dari kondisi lingkungan sekitarnya melalui sensor. Keempat adalah aspek Localization merupakan aspek yang terkait dengan kemampuan robot dalam bagaimana menentukan posisinya terhadap lingkungan sekitarnya, seperti yang terlihat pada gambar 1. Terakhir adalah aspek navigation dan planning, merupakan aspek yang terkait dengan kempuan robot dalam bagaimana membuat rencana dan navigasi terhadap goal yang dituju. Kesemua aspek ini memampukan robot agar dapat beroperasi untuk mengerjakan task sesuai dengan goal yang telah ditentukan sebelumnya pada robot tersebut.



Gambar.1.1: Permasalahan Localization dari Robot

Dari kesemua aspek-aspek di dalam bidang ilmu robotika, aspek Localization merupakan aspek yang mendapatkan perhatian terbesar oleh penelitian-penelitian di dalam dekade terkahir ini (Siegwart & Nourbakhsh, 2004). Berdasarkan penelitian-penelitian tersebut, keseluruhan metode dapat dibagi secara umum menjadi tiga macam berdasarkan survey literatur yang dilakukan oleh DeSouza dan Kak, 2002. Ketiga macam jenis kategori metodenya adalah Absolute (atau Global) Localization, Incremental Localization, dan Localization yang diturunkan dari Landmark Tracking. Diantara ketiga kategori tersebut, pendekatan Inceremental Localization merupakan pendekatan dengan lokasi inisial yang diketahui melalui pendekatan; kemudian menggunakan algoritma localization dengan tetap melacak ketidakpastian dari posisi robot ketika mengeksekusi motion; dan ketika ketidakpastian tersebut melewati batas tertentu, digunakan sensor untuk mengkoreksi posisinya. Incremental Localization umumnya dibangun berdasarkan teori dan teknik probabilitas dan merupakan salah satu teknik yang menjanjikan dalam menyediakan solusi yang

komprehensif dan real-time terhadap permasalahan localization (Dellaert et al, 1999). Pada Incremental Localization atau Probabilistic Localization sendiri terdapat beberapa algoritma seperti Kalman Filter, Extended Kalman Filter, Information Filter, Histogram Filter, dan Particle Filter (dan salah satu turununannya adalah teknik Monte Carlo Localization).

Dengan semakin banyaknya metode serta teknik-teknik yang digunakan dalam permasalahan localization, tentu saja diperlukan pengujian-pengujian terhadap robot agar dapat dihasilkan data-data agar dapat mengevaluasi metode atau teknik tersebut. Akan tetapi, pengujian dengan menggunakan robot dan environment sesungguhnya sungguh sangat memakan waktu dan biaya yang tidak sedikit untuk menguji suatu metode atau teknik tertentu terutama dengan teknik metode atau teknik baru yang belom pernah diterapkan sama sekali. Oleh karena itu saat diperlukan pengujian di tingkat simulasi terlebih dahulu ketimbang melakukan pengujian menggunakan robot sungguhan. Dengan menggunakan simulator, perubahan dan tuning dalam parameter dapat dilakukan lebih mudah dan dapat dibuat simulasi dengan environment yang menyerupai kondisi sesungguhnya. Salah satu dari sekian banyak simulator robot yang ada saat ini, simulator V-REP dari Copella robotics merupakan salah satu yang terbaik karena bersifat multiplatform dan open-source. Diharapkan dengan penggunaan simulator robot V-REP, dapat dilakukan penelitian dalam memodelkan teknik MCL dan mendapatkan hasil evaluasi teknik tersebut.

2. Dasar Teori

2.1 Introduksi Probablistic Robotics

Probabilistic Robotic adalah suatu pendekatan relatif baru untuk robotika yang memfokuskan kepada ketidakpastian persepsi dan tindakan robot. Ide utama dari probabilistic robotic adalah untuk mewakili ketidakpastian secara eksplisit, dengan menggunakan perhitungan teori probabilitas. Dengan kata lain, ketimbang hanya mengandalkan satu "tebakan terbaik" untuk sesuatu yang akan mngkin terjadi, algoritma probabilistik merepresentasikan informasi dengan distribusi probabilitas atas keseluruhan ruang kemungkin dari hipotesis. Dengan demikian, keseluruhan hipotesis tersebut dapat merepresentasikan ambiguitas dan tingkat kepercayaan secara matematis, sehngga memungkinkan semua sumber dari ketidakpastian dapat terus dicek. Selain itu, dengan mendasarkan kendali keputusan dengan informasi probabilistik, algoritma ini dapat diturunkan dengan baik dalam mengikuti berbagai model atau distribusi ketidakpastian yang ada, yang mengarah ke solusi untuk masalah robotika seperti halnya localization.

2.2 Konsep Dasar dalam Probabilitas

Sub-bab ini membahas notasi dan konsep dasar dari probabilistik. Dalam robotika probabilistik, kuantitas metric seperti pengukuran sensor, kontrol, dan state-state dari robot dan lingkungannya dianggap sebagai variabel acak atau random variable. Variabel acak dapat berupa kumpulan nilai (multiple-value) berdasarkan aturan probabilistik tertentu. Inferensi dari probabilistik (Probabilistic Inference) sendiri adalah proses mengkalkulasikan aturan-aturan dari variabel acak yang berasal atau diturunkan dari variabel acak lainnya, misalkan dari pemodelan data sensor.

Misalkan terdapat sebuah variabel acak X yang menyatakan sebuah even spesifik yang dapat terjadi. Sebuah contoh standar dari variabel acak adalah pelemparan koin, dimana X dapat menjadi salah satu dari kedua sisi koin. Jika ruang kemungkinan dari X adalah ruang diskrit, seperti halnya contoh dari kasus pelemparan koin, maka dapat dinyatakan persamaan berikut :

$$p(X = x) \tag{2.1}$$

Untuk menyatakan kemungkinan bahwa variabel acak X memiliki nilai x. Misalnya, sebuah pelembaran koin yang adil akan ditandai dengan $p(\mathrm{X} = \text{kepala}) = p(\mathrm{X} = \text{ekor}) = 0.5$. Dan jumlah probabilitas diskritnya berjumlah satu, yaitu,

$$\sum_x p(X = x) = 1, \tag{2.2}$$

dan tentu saja, probabilitas selalu non-negatif, yaitu, $p(\mathrm{X} = \mathrm{x}) \geq 0$. Untuk menyederhanakan notasi, biasanya akan variabel random tidak dinotasikan secara eksplisit, dan sebagai gantinya menggunakan singkatan umum $p(\mathrm{x})$ bukan ketimbang menulis $p(\mathrm{X} = \mathrm{x})$.

Sebagian besar teknik yang ada dalam probabilistik dalam pengambilan keputusan dimodelkan ruang kontinyu. Diasumsikan bahwa semua variabel acak kontinu memiliki Probability Density Function (PDF). PDF adalah Sebuah fungsi densitas satu dimensi yang merupakan distribusi normal dengan mean μ dan varian σ^2. Distribusi ini dinotasikan dengan fungsi Gaussian sebagai berikut :

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

Dan persamaan ini dipersingkat dengan notasi $N(\mathrm{x};\ \mu,\ \sigma^2)$ yang menspesifikan variabel acak (dengan mean dan varian-nya).

Distribusi Normal (persamaan 2.3) mengasumsikan bahwa x adalah nilai skalar. Seringkali, x akan menjadi
vektor multi-dimensi. Distribusi normal melalui vektor disebut dengan *multivariate*. Distribusi normal multivariat dengan fungsi densitas sebagai berikut:

$$p(x) = \det\left(2\pi\Sigma\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{2.4}$$

Dimana disini μ adalah vektor dari mean dan Σ a (semidefinite positif) adalh matriks simetris yang disebut matriks kovarians. Sedangkan T menandai transpos dari vektor.

Persamaan (2.3) dan (2.4) adalah contoh dari PDF. Seperti halnya distribusi probabilitas diskrit yang selalu berjumlah 1, maka sebuah PDF jika diintegrasi akan bernilai 1.

$$\int p(x)\,dx = 1. \tag{2.5}$$

Namun, tidak seperti probabilitas diskrit, nilai dari PDF tidak dibatasi atas oleh nilai 1. Kemudian nilai dari joint distribution dari dua variabel acak X dan Y dapat diformulasikan dengan persamaan berikut.

$$p(x,y) = p(X = x \text{ and } Y = y). \tag{2.6}$$

Persamaan ini menggambarkan probabilitas dari event yang variabel acak X ambil dari nilai x dan Y mengambil nilai y. Jika X dan Y adalah independen, maka didapatkan.

$$p(x,y) = p(x)\,p(y). \tag{2.7}$$

Seringkali, variabel acak membawa informasi tentang variabel acak lainnya. Misalkan diketahui bahwa nilai Y adalah y, dan ingin diketahui probabilitas bahwa X bernilai x dengan kondisi yang diberikan. Maka probabilitas yang dimaksud tersebut dapat direpresentasikan dengan notasi conditional probability sebagai berikut.

$$p(x \mid y) = p(X = x \mid Y = y) \tag{2.8}$$

Sehingga jika $p(y) > 0$, maka conditional probability akan didefinisikan sebagai berikut.

$$p(x \mid y) = \frac{p(x,y)}{p(y)}. \tag{2.9}$$

Jika X dan Y independent maka dihasilkan.

$$p(x \mid y) = \frac{p(x)\,p(y)}{p(y)} = p(x). \tag{2.10}$$

Dengan kata lain, jika X dan Y independent, maka Y tidak akan dapat memberitahukan nilai X sama sekali dan tidak ada gunanya mengetahui nilai Y. Fakta lainnya yang menarik adalah, dengan definisi dari conditional probability dan aksioma dari pengukuran probabilitas sebelumnya sering disebut sebagai teorema dari total probabilitas sebagai berikut.

$$p(x) = \sum_y p(x \mid y)\,p(y) \quad \text{(discrete case)} \tag{2.11}$$

$$p(x) = \int p(x \mid y)\,p(y)\,dy \quad \text{(continuous case)} \tag{2.12}$$

Jika $p(x \mid y)$ atau $p(y)$ adalah nol, dapat didefinisikan perkalian dari $p(x \mid y)$ adalah nol, walaupun nilai dari faktor lainnya. Hal lain yang sama pentingnya adalah Bayes rule, yang mengkaitkan conditional dari $p(x \mid y)$ dengan inverse nya yaitu $p(y \mid x)$. Aturan dari bayes rule membutuhkan syarat yaitu $p(y) > 0$.

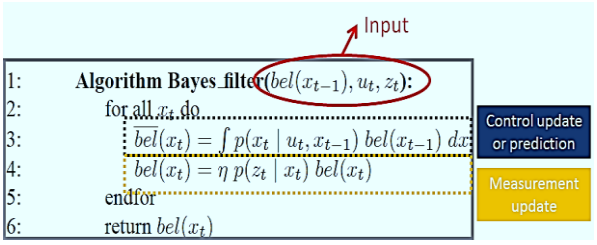
$$p(x \mid y) = \frac{p(y \mid x)\,p(x)}{p(y)} = \frac{p(y \mid x)\,p(x)}{\sum_{x'} p(y \mid x')\,p(x')} \quad \text{(discrete)} \tag{2.13}$$

$$p(x \mid y) = \frac{p(y \mid x)\,p(x)}{p(y)} = \frac{p(y \mid x)\,p(x)}{\int p(y \mid x')\,p(x')\,dx'} \quad \text{(continuous)} \tag{2.14}$$

Persamaan dari Bayes rule memainkan peran dominan dalam probabilistic robotics. Jika x adalah kuantitas yang ingin disimpulkan dari y, maka probabilitas $p(x)$ akan disebut probabilitas sebagai distribusi probabilitas sebelumnya atau *prior probability distribution*, dan y disebut sebagai data (misalnya, pengukuran sensor).

Distribusi dari $p(x)$ merangkum knowledge yang dimiliki terkait dengan X sebelum menggabungkannya dengan data y. Probabilitas $p(x \mid y)$ sendiri disebut sebagai distribusi probabilitas sesudah X atau *posterior probability distribution*. Seperti yang persamaan (2.14) tunjukan, Bayes rule menyediakan cara untuk menghitung *posterior probability* $p(x \mid y)$ menggunakan "inverse" dari *conditional probability* $p(y \mid x)$ bersama dengan *prior probability* $p(x)$.

Dengan kata lain, jika ingin disimpulkan kuantitas atau nilai x dari data sensor y, maka Bayes rule memungkinkanhal tersebut melalui probabilitas inverse, yang menspesifikkan probabilitas dari data y dengan asumsi bahwa x telah terjadi. Dalam robotika, probabilitas inverse ini sering disebut dengan istilah "generative mode", karena menggambarkan beberapa tingkatan abstraksi dalam bagaimana variabel state X menghasilkan pengukuran sensor Y. Salah satu penerapan dari Bayes rule adalah penerapannya melalui bayer filter. Bayes filter adalah algoritma yang paling umum dalam menghitung tingkat kepercayaan atau belief (distribusi belief yang dinotasikan , bel) dari pengukuran dan kontrol data.



Gambar.2.1: Algoritma Bayes Filter

2.3 Particle Filter

Particle Filter merupakan alternatif implementasi nonparametrik dari Bayes filter. Particle Filter sendiri memperkirakan probability posterior dengan sejumlah parameter finite. Namun parameter tersebut berbeda dalam dihasilkannya, dan juga dalam mempopulasikan state space nya. Ide utama dari particle filter alah dengan merepresentasikan posterior believe atau dinotasikan $bel(x_t)$ oleh sekumpulan sampel acak dari state yang diambil dari posterior ini. Ketimbang merepresentasikan distribusi dengan bentuk parametrik (fungsi eksponensial yang mendefinisikan densitas dari distribusi normal), particle filter merepresentasikan distribusi dari sejumlah kumpulan sampel yang diambil distribusi tersebut. Representasi demikian merupakan suatu bentuk aproksimasi, tetapi bersifat nonparametrik, dan karena itu dapat mewakili ruang atau space state yang lebih luas daripada distribusi lainya seperti distribusi Gaussian.

Pada particle filter, sample dari distribusi posterior disebut dengan partikel dan dinotasikan dengan notasi berikut.

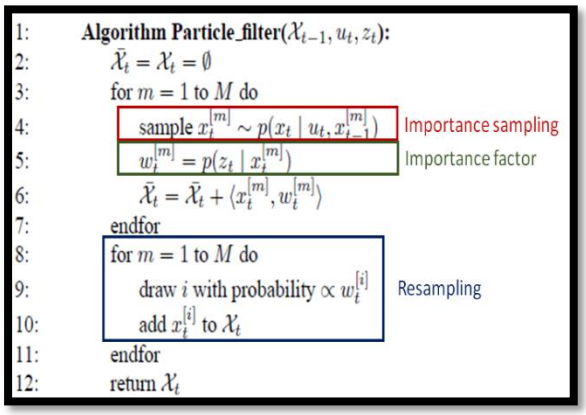
$$\chi_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]} \qquad (2.15)$$

Setiap partikel $x_t^{[m]}$ (dengan $1 \leq m \leq M$) adalah instansiasi konkret dari state pada waktu t. Di sini M menunjukkan jumlah partikel dalam kumpulan partikel X_t. Dalam prakteknya, jumlah partikel M

sering berjumlah besar, misalnya, M = 1000. Pada implementasi lainnya M adalah fungsi dari t atau jumlah lain yang berkaitan dengan bel (believe) atau tingkat keyakinan (x_t). Intuisi di balik particle untuk mendekati keyakinan atau belive yang dinotasikan bel(x_t) oleh himpunan partikel X_t. Secara ideal, kemungkinan peluang untuk hipotesis state x_t untuk disertakan dalam himpunan partikel Xt harus proporsional dengan posteriror dari Bayes Filter nya yaitu bel(x_t):

$$x_t^{[m]} \sim p(x_t \mid z_{1:t}, u_{1:t}) \quad (2.16)$$

Algoritma dari partikel filter sendiri membangun belief bel(x_t) secara rekursif dari belief sebelumnya bel(x_{t-1}). Karena belief direpresentasikan oleh sekumpulan partikel, maka berarti particle filter mengkonstruksi kumpulan partikel X_t secara rekursif dari kumpulan X_{t-1}. Untuk implementasi dari particle filter sendiri dapat dilihat pada gambar berikut.



Gambar 2.2: Algoritma Particle Filter

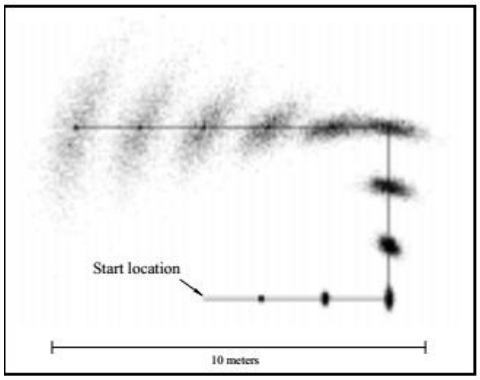
2.4 Monte Carlo Localization (Particle Filter-based Localization)

Monte Carlo Localization atau MCL adalah versi lain dari metode sampling/re-sampling dengan tingkat kepentingan. Menurut penelitian Fox, et al 1999, ide utama dari MCL adalah merepresentasikan posterior belief Bel(l) dengan sekumpulan N yang diberi bobot, sampel random dari partikel S = {s_i | i = 1::N}. Sebuah kumpulan sampel membentuk aproksimasi diskri dari distribusi peluang atau probabilitas. Sampel-sampe pada MCL dapat dinotasikan <<*x*, *y*, θ>, *p*> dimana <*x*, *y*, θ> menyatakan posisi robot, dan p p >= 0 adalah faktor numeris pembobotannya, yang analogi dengan probabilitas diskritnya. Dan dengan sifat konsistensi penjumlahan total nya $\sum_{n=1}^{N} pn = 1$. MCL sendiri diproses dalam dua tahapan sebagai berikut.

Robot motion. Ketika robot bergerak MCL menghasilkan sampel baru N yang mengaproksimasikan posisi robot setelah perintah bergerak (motion command). Setiap sampel dihasilkan dengan secara random mengambil sebuah sampel dari sekumpulan sampel yang telah dikalkulasi sebelumnya nilai peluang *p* nya. Misalkan *l'* menyatakan posisi dari sampel ini. Maka sampel baru dari *l* akan dihasilkan dengan membuat sebuah random sampel tunggal dari *P*(*l'* | *l*, □). Menggunakan aksi □ yang telah diobservasi. Nilai p dari sampel yang baru adalah N^{-1}.

Gambar 2.3 menunjukkan efek dari teknik

sampling ini, dengan memulai posisi inisial diketahui lebih dulu (posisi dibawah) dan kemudian mengeksekusi perintah seperti yang digambarkan dengan garis lurus.



Gambar 2.3: Aproksimasi berdasarkan sampling dari tingkat belief posisi sebuah non-sensing robot

Seperti yang dapat dilihat pada gambar 2.3, kumpulan sampel mengaproksimasikan distribusi dengan meningkatkan ketidakpastian, dan merepresentasikan penurunan kemungkinan kehilangan informasi dikarenakan robot slip atau bergeser.

Sensor reading adalah tahapan yang menggabungkan pembobotan ulang dari sekumpulan sampel, dengan mengimplementasikan Bayes rule di dalam Markov Localization (Thurn, 2001). Secara lebih spesifik misalkan terdapat sampel <*l*. *p*>. Kemudian p ← □ *P* (*s* | *l*), dimana *s* adalah nilai pengukuran sensor, dan □ adalah nilai konstanta normalisasi yang membuat nilai $\sum_{n=1}^{N} \boldsymbol{pn} = \mathbf{1}$. Penggabungan dari pembacaan sensor secara tipikal dilakukan dalam dua fase, pertama dengan dengan memultiplikasikan *p* dengan *P*(*s* | *l*), dan kedua beragam nilai-*p* nya di normalisasikan.

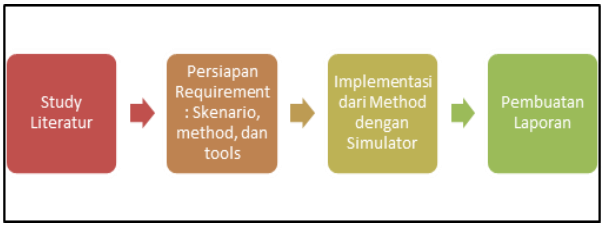
Pada prakteknya oleh Thurn et al, 2006, algoritma MCL merepresentasikan belief dengan notasi sebagai persebaran M partikel $X_t = \{x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}\}$ pada peta. Partikel yang dimaksud disini adalah representasi pose robot pada suatu titik tertentu, dan setiap partikel pada peta memiliki motion model serta measurement model masing. Initial belief dari robot tersebut dinotasikan dengan *bel*(x_0) dan partikel akan disebar secara random pada peta. Untuk pseudo code dari algoritma MCL sendiri adalah sebagai berikut.

```
1:   Algorithm MCL(X_{t-1}, u_t, z_t, m):
2:       X̄_t = X_t = ∅
3:       for m = 1 to M do
4:           x_t^[m] = sample_motion_model(u_t, x_{t-1}^[m])
5:           w_t^[m] = measurement_model(z_t, x_t^[m], m)
6:           X̄_t = X̄_t + ⟨x_t^[m], w_t^[m]⟩
7:       endfor
8:       for m = 1 to M do
9:           draw i with probability ∝ w_t^[i]
10:          add x_t^[i] to X_t
11:      endfor
12:      return X_t
```

Gambar 2.4 :
Algoritma Monte Carlo Localization (MCL)

3. Metodologi

Penelitian ini bertujuan untuk menerapkan algoritma MCL melalui simulator V-REP. Adapapun tahapan-tahapan dari metodologi yang dilakukan dari penelitian ini adalah sebagai berikut.



Gambar 3.1: Metodologi Penelitian

1. Study Literatur : merupakan tahapan survey terhadap literatur-literatur yang bersumber dari buku dan paper jurnal maupun prosiding terkait dengan probabilistic robotic dan lebih spesifik lagi terkait dengan teknik Monte Carlo Localization pada robot.

2. Persiapan Requirement : Skenario, Method, dan Tools.
   - Tools yang digunakan adalah sebagai berikut :
     - Simulator V-REP Pro EDU versi 3.2.3
     - Python (x,y) versi 2.7
     - Library numpy
   - Skenario sebagai berikut :

TABEL 3.1:

No.	Skenario
1.	Pembuatan Peta pada Simulator V-REP dengan spesifikasi sebagai berikut : • Peta dapat diasumsikan sebgai diagram Cartesian dengan ukuran 5 satuan (diasumsikan sebagai 5 meter) dengan kordinat (xMin, yMin) = (-2.5, -2.5) dan (xMax, yMax) = (2.5, 2.5) • Tembok tersusun atas blok-blok tembok kecil yang dapat diketahui titik tengahnya.
2.	Penggunaan model robot pioneer pada simolator VREP, dimana robot bergerak dengan prinsip *dofferential drive*.
3.	Partikel dapat diasumsikan sebagai sebuah robot tambahan bernama ‘particlebot’. Particlebot dapat dipindahkan sesuai pose yang diinginkan dan dapat membaca sensor layaknya robot asli sebagai asumsi ray casting pada sensor model.

Skenario dari Environment Simulator V-REP

- Method yang digunaakan adalah algoritma MCL yang terdapat pada gambar 2.4.

3. Implementasi dari Metod dengan sebagai berikut :

No	Implementasi Method
1.	Penerapan algoritma MCL dengan mengimplementasikan motion model robot melalui asumsi velocity-based dan diferential drive
2.	Menentukan fungsi untuk mengubah input awal berupa kecepatan roda kiri dan kanan menjadi kecepatan translasi (v) dan rotasi (w) dari robot
3.	Melakukan tuning parameter-parameter terkait untuk menemukan model yang paling mirip dengan kondisi asli robot
4.	Melakukan pengujian dan eksperimen terhadap pengaruh

	parameter, jarak maksimal, pembacaan sensor, dan hynkag sensor yang digunakan dalam penentuan bobot partikel.

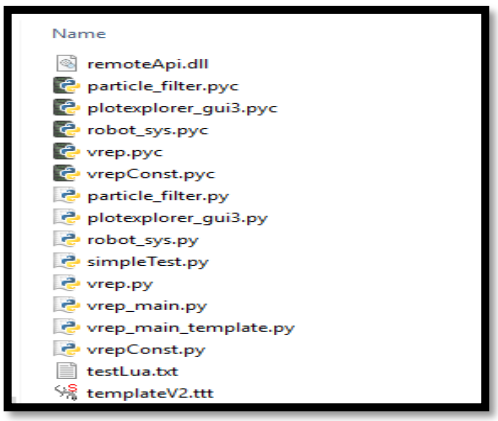
Implementasi dari Algoritma MCL

4. Pembuatan Laporan

Langkah terakhir adalah melakukan analisa terhadap hasil penelitian dan kemudian melakukan dokumentasi terhadap data-data yang dihasilkan serta membuat kesimpulan general dari eksperimen yang dilakukan.

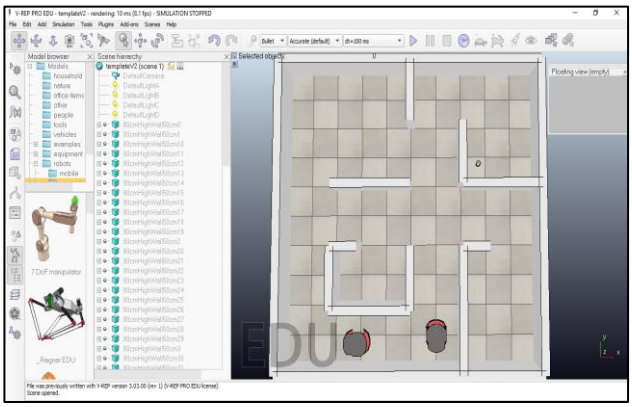
4. Implementasi

Hasil dari implementasi dari penelitian ini adalah scene dari simulator v-rep yang merepresentasikan skenario environment robot; dan ditambah dengan penerapan algoritma MCL melalui script ekstensi python sebagai logik daripada behaviour robot pioneer. Berikut struktur file-file hasil dari pemodelan yang dibuat dengan menggunakan simulator V-REP dan Python (x,y).



Gambar 2.5: Struktur Project File

Dimana scene dari robot dan environmentnya terdapat pada file templateV2.ttt; dan penerapan dari algoritma MCL terdapat pada file particle_filter.py. Untuk gambaran scene-nya sendiri terdapat pada gambar 2.6 sebagai berikut.



Gambar 2.6: Implementasi Scene dari Simulator V-REP

Sedangkan implementasi dari algoritma Monter Carlo Localization yang terdapat pada gambar 2.4, diterapkan dengan python (x, y) melalui script sebagai berikut.

5. Hasil dan Analisis

Untuk hasil dan kesimpulannya mengacu terhadap penerapan metodologi yang terkait dengan Implementasi Metode MCL. Hasilnya adalah Pertama, skenario dan hasil dari eksperimen motion model, diikuti dengan eksperimen sensor model dan Monte Carlo Localization. Hasil dari masing-masing skenario tersebut kemudian dianalisa dan diambil kesimpulan.

5.1 Motion Model

Ada dua skenario untuk motion model, yaitu mengubah parameter a1 s/d a6 dan kecepatan. Untuk skenario motion model, dilakukan penyesuaian terhadap parameter-parameter yang terkait dengan estimasi error motion model. Parameter yang akan dioptimalisasi adalah sebagai berikut: $\alpha 1$, $\alpha 2$, $\alpha 3$, $\alpha 4$, $\alpha 5$, $\alpha 6$, parameter-parameter tersebut adalah nilai yang akan mempengaruhi perhitungan eror spesifik yang akan digunakan (akurasi posisi robot real dengan partikelnya atau average error pose). Parameter alpha tersebut memodelkan akurasi posisi robot. Semakin tidak akurat robot, semakin besar nilai dari parameter-parameter ini. Tujuan dari optimalisasi parameter ini untuk mendapatkan akurasi terbaik posisi partikel terhadap posisi robot setelah bergerak pada selang waktu tertentu. Selang waktu yang digunakan pada eksperimen ini adalah 10 iterasi dengan masing-masing iterasi berkisar sekitar 1 detik. Dari setiap iterasi nilai eror yang ada di rata-ratakan pada setiap iterasi. (rata-rata dari sepuluh iterasi).

Terdapat 10 partikel yang digunakan pada skenario ini. Mula-mula, partikel diletakkan tepat di posisi robot berada Selanjutnya, Ada tiga skenario eksperimen yang dilakukan untuk melakukan parameter tuning tersebut, yaitu:

1. *Gerak arah lurus dengan kecepatan Vr = 2 dan Vl = 2*

Dikarenakan skenario pertama ini berhubungan dengan gerak lurus, parameter yang diuji adalah yang berhubungan dengan kecepatan linear saja, yaitu $\alpha 1$ dan $\alpha 2$. Akurasi dinilai dari jarak euclidian antara titik tengah partikel dan titik tengah robot. Semakin kecil jarak Euclidian tersebut, semakin kecil error dari posisi partikel. Artinya, semakin akurat posisi partikel tersebut. Terdapat tiga percobaan untuk parameter-parameter tersebut, yaitu:

a) $\alpha 1 = \alpha 2 = 0.01$
b) $\alpha 1 = \alpha 2 = 0.05$
c) $\alpha 1 = \alpha 2 = 0.1$

Masing-masing percobaan dilakukan sebanyak 5 kali eksperimen. Setiap percobaan
dihitung akurasinya berdasarkan rata-rata error posisi partikel pada 5 eksperimen tersebut.

Berikut hasil eksperimen pertama parameter tuning untuk motion model, yaitu robot bergerak lurus dengan kecepatan linear roda Vr = 2 dan Vl = 2.

Parameter	**Avg err pose**
a1=a2=0.01	**0.517580518**
a1=a2=0.05	0.525936277
a1=a2=0.1	0.549110223
a1=a2=0.5	0.550981224
a1=a2=1.0	0.703575364

Dari tabel tersebut dapat dilihat bahwa parameter pertama, yaitu $\alpha 1 = \alpha 2 = 0.01$, adalah parameter yang menhasilkan rata-rata error posisi terkecil, yaitu **0.517580518**. Dan secara umum semakin besar nilai a1 dan a2 maka semakin besar juga error rata-rata pose nya.

2. *Gerak memutar dengan kecepatan Vr = -2 dan Vl = 2*

Dikarenakan skenario kedua ini terkait dengan gerak memutar, parameter yang diuji adalah yang berhubungan dengan kecepatan angular dan sudut theta robot, yaitu $\alpha 3$, $\alpha 4$, $\alpha 5$, dan $\alpha 6$. Akurasi dinilai dari jarak euclidian antara titik tengah partikel dan titik tengah robot. Akurasi juga dinilai dari selisih sudut theta antara partikel dengan robot.

Terdapat empat percobaan untuk parameter parameter tersebut, yaitu:

a) $\alpha 3 = \alpha 4 = 0.01$ dan $\alpha 5 = \alpha 6 = 0.01$
b) $\alpha 3 = \alpha 4 = 0.01$ dan $\alpha 5 = \alpha 6 = 0.05$
c) $\alpha 3 = \alpha 4 = 0.05$ dan $\alpha 5 = \alpha 6 = 0.01$
d) $\alpha 3 = \alpha 4 = 0.05$ dan $\alpha 5 = \alpha 6 = 0.05$

Masing-masing percobaan dilakukan sebanyak 5 kali eksperimen. Setiap percobaan
dihitung akurasinya berdasarkan rata-rata error posisi dan error sudut theta partikel pada 5 eksperimen tersebut.

Untuk skenario kedua, dari sebelumnya parameter $\alpha 1 = \alpha 2 = 0.01$ digunakan. Hasil untuk skenario kedua, yaitu robot bergerak memutar dengan kecepatan Vr = -2 dan Vl = 2,
dapat dilihat pada tabel berikut.

Parameter	Avg err pose	Avg err theta
a3=a4=0.01 & a5=a6=0.01	0.085277334	3.66358684
a3=a4=0.01 & a5=a6=0.05	0.087503319	3.23982354
a3=a4=0.05 & a5=a6=0.05	0.080201783	**3.31687889**
a3=a4=0.1 & a5=a6=0.05	0.086867418	3.69736732
a3=a4=0.1 & a5=a6=0.1	0.087706794	3.69981779

Dari tabel tersebut, dapat kita lihat bahwa nilai 0.05 untuk $\alpha 3$, $\alpha 4$, $\alpha 5$, $\alpha 6$ menghasilkan rata-rata error terkecil untuk posisi dan sudut theta, yaitu sebesar **0.080201783** dan **3.31687889**. Hal ini tentu sangat kecil dibandingkan nilai parameter lainnya. Pengaruh dari parameter ini berkaitan dengan akurasi terhadap posisi robot berputar serta sudut orientasinya

3. *Gerak lurus dan berbelok secara bersamaan dengan kecepatan linear roda Vr = 2 dan Vl = 1.*

Dikarenakan skenario kedua ini berhubungan dengan gerak lurus dan memutar, parameter yang diuji adalah yang berhubungan dengan kecepatan angular dan sudut theta robot, yaitu $\alpha 3$, $\alpha 4$, $\alpha 5$, dan $\alpha 6$. Terdapat empat percobaan untuk parameter-

parameter tersebut, yaitu:

a) $\alpha 3 = \alpha 4 = 0.01$ dan $\alpha 5 = \alpha 6 = 0.01$
b) $\alpha 3 = \alpha 4 = 0.01$ dan $\alpha 5 = \alpha 6 = 0.05$
c) $\alpha 3 = \alpha 4 = 0.05$ dan $\alpha 5 = \alpha 6 = 0.01$
d) $\alpha 3 = \alpha 4 = 0.05$ dan $\alpha 5 = \alpha 6 = 0.05$

Masing-masing percobaan dilakukan sebanyak 5 kali eksperimen. Setiap percobaan
dihitung akurasinya berdasarkan rata-rata error posisi dan error sudut theta partikel pada 5 eksperimen tersebut.

Hasilnya eksperimen berikutnya adalah robot bergerak lurus sambil berbelok dengan kecepatan linear roda Vr = 2 dan Vl = 1. Data dari eksperimen tersebut dapat dilihat pada table berikut.

Parameter	Avg err pose	Avg err theta
a3=a4=0.01 & a5=a6=0.01	0.09055738	0.58386514
a3=a4=0.01 & a5=a6=0.05	0.08957552	0.58174881
a3=a4=0.05 & a5=a6=0.05	0.09265646	0.59184786
a3=a4=0.1 & a5=a6=0.05	0.08895663	0.58201771
a3=a4=0.1 & a5=a6=0.1	**0.07981847**	**0.55817047**

Nilai parameter terbaik untuk error sudut theta didapatkan oleh $\alpha 3 = \alpha 4 = 0.1$ *dan* $\alpha 5 = \alpha 6 = 0.1$, dengan error posisi **0.07981847** dan sudut theta sebesar **0.55817047**. Error tersebut sangat merupakan yang terkecil dibandingkan dengan error nilai parameter lainnya. Juga dibandingkan dengan nilai error pada gerak melingkar (percobaan sebelumnya ketika a3=a4=0.05 dan a5=a6=0.05 niali eror thetanya jauh lebih besar **3.31687889**). Jadi nilai yang dipilih untuk a3=a4=a5=a6=0.1.

4. Robot bergerak dengan variasi kecepatan Vr dan Vl

Skenario selanjutnya adalah pengujian terhadap pengaruh kecepatan linear kedua roda robot terhadap akurasi posisi partikel setelah partikel dan robot bergerak. Pada skenario ini, diggunakan hasil parameter tuning di skenario pertama. Selain itu, gerakan yang digunakan adalah gerak lurus. Terdapat tiga skenario kecepatan, yaitu:

a) Vr = Vl = 2
b) Vr = Vl = 3
c) Vr = Vl = 4

Selain parameter tuning, juga dilakukan eksperimen pengaruh kecepatan robot terhadap tingkat keakuratan posisi partikel dibandingkan posisi robot. Hasil dari eksperimen ini dapat dilihat pada tabel berikut.

Parameter	Avg err pose	Avg err theta
vl = vr = 2	**0.496210883**	**0.0588411**
vl = vr = 3	0.755616146	0.08598396
vl = vr = 4	0.640713548	0.18546879

Dapat dilihat bahwa hasil eror terbaik didapatkan dengan parameter vl = 2 dan vr = 2.

5.2 Sensor Model

Pada eksperimen sensor model, dilakukan tuning pada parameter zhit, zshort, zmax, zrand, deltahit, dan lambdashort untuk mendapatkan penentuan bobot partikel yang paling optimal. Berikut empat percobaan menggunakan sensor ke 1,2,4,5,7 pada Pioneer 3dx:

1) zhit = 0.25 , zshort = 0.25 , zmax = 0.25 , zrand = 0.25 , deltahit = 1 , lambdashort = 1
2) zhit = 0.2 , zshort = 0.3 , zmax = 0.25 , zrand = 0.25, deltahit = 1 , lambdashort = 1
3) zhit = 0.2 , zshort = 0.3 , zmax = 0.3 , zrand = 0.2, deltahit = 1 , lambdashort = 1
4) zhit = 0.2 , zshort = 0.3 , zmax = 0.25 , zrand = 0.25, deltahit = 0.5 , lambdashort = 0.5

Pada penelitian kali ini, dilakukan eksperimen terhadap empat konfigurasi parameter dengan sensor ke 1,2,4,5,7 dan max_range = 2 dalam pencarian hasil yang paling optimal. Jumlah iterasi yang digunakan pada percobaan hanya 1 kali.

Parameter	Probability q
zhit = 0.25 zshort = 0.25 zmax = 0.25 zrand = 0.25 deltahit = 1 lambdashort = 1	0.014013005
zhit = 0.2 **zshort = 0.3** **zmax = 0.25** **zrand = 0.25** **deltahit = 1** **lambdashort = 1**	**0.025534417**
zhit = 0.2 zshort = 0.3 zmax = 0.3 zrand = 0.2 deltahit = 1 lambdashort = 1	0.005926434
zhit = 0.2 zshort = 0.3 zmax = 0.25 zrand = 0.25 deltahit = 0.5 lambdashort= 0.5	0.010466593

Digunakan nilai prob_q sebagai tolak ukur dalam penentuan konfigurasi yang paling

optimal. Pemilihan prob_q didasari dikarenakan prob_q dapat merepresentasikan nilai

kemiripan antara partikel dengan kondisi robot aslinya. Dari hasil eksperimen diatas didapat bahwa konfigurasi kedua adalah yang paling baik.

Selain parameter tuning, juga dilakukan analisa pengaruh jarak maksimal

pembacaan sensor terhadap penentuan bobot partikel. Jarak yang digunakan adalah 2, 3 dan 4 satuan. Selanjutnya diggunakan hasil tuning paratemer terbaik diatas untuk mencari konfigurasi jarak maksimum (max_range) optimal. Berikut adalah data hasil ekperimen konfigurasi max_range.

Parameter	Probability q
Max_range = 2	**0.009782**
Max_range = 3	0.001906
Max_range = 4	0.013371

Eksperimen sensor model yang terakhir adalah analisis pengaruh jumlah sensor yang digunakan terhadap penentuan bobot partikel. Jumlah sensor yang digunakan adalah 4, 6, 8, dan 10 sensor. Pada hasil eksperimen sebelumnya digunakan nilai max_range = 2 karena nilai dengan eror yang paling kecil. Kemudian hasil ini dipergunakan untuk mencari penggunaan sensor yang paling tepat.

Sensor	Probability_q
1, 2, 7, 8	**0.010748**
1, 2, 4, 5, 7, 8	0.00582
1, 2, 4, 5, 7, 8, 12 13	0.005444
1, 2, 4, 5, 7, 8, 10, 12, 13, 15	0.005461

Dari data diatas, ternyata penentuan sensor tidak berpengaruh banyak karena setiap hasil prob_q akan di*normalize* saat resampling, sehingga semua nya dapat memiliki hasil yang sama optimal dan yang dipilih adalah penggunaan empat sensor dalam penelitian ini karena lebih sedikit memakan waktu pemrosesan.

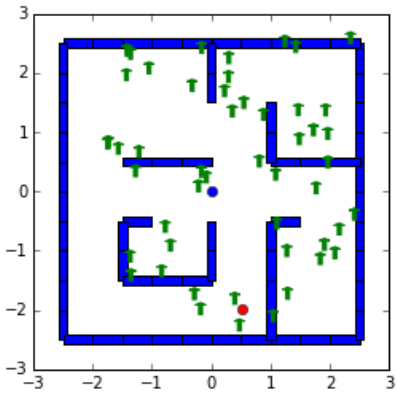
Untuk selanjutnya. Jadi, nilai parameter yang optimal akan digunakan adalah sebagai berikut :

- *a1=a2=0.01*
- *a3=a4=0.1 & a5=a6=0.1*
- *zhit = 0.2*
- *zshort = 0.3*
- *zmax = 0.25*
- *zrand = 0.25*
- *deltahit = 1*
- *lambdashort = 1*
- *Jumlah sensor = 4 (nomor 1,2,7,8)*

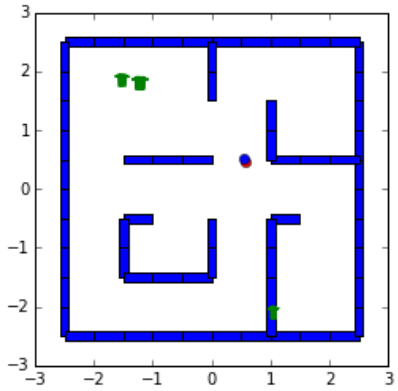
5.3 MCL

Pada eksperimen MCL, dilakukan eksperimen mengenai pengaruh gerakkan robot, jumlah partikel, dan interval antar iterasi terhadap kecepatan dan akurasi konvergensi partikel menuju titik posisi robot sebenarnya. Parameter-parameter yang digunakan adalah parameter terbaik yang didapat pada percobaan motion dan sensor model sebelumnya untuk menjalankan eksperimen MCL. Sedangkan untuk jumlah partikel, digunakan sejumlah 50, 100, dan 150 partikel. Akan dilihat sejauh mana pengaruh dari tingkat kecepatan konvergensi gerak lurus dari robot jika jumlah partikelnya ditambahkan. Pada pengujian disini dilakukan gerakan lurus dengan dibatasi waktu sejumlah 10 kali iterasi (loop).

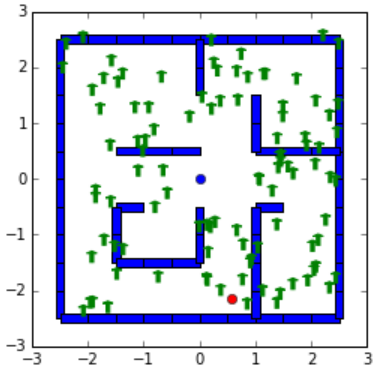
Jumlah Partikel = 50, awal inisialisasinya adalah sebagai berikut.



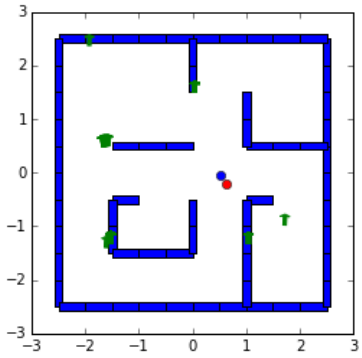
Dengan jumlah pertikel = 50, akan konvergen sesudah sampai ke-iterasi 9.



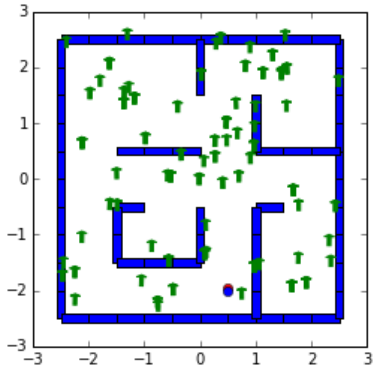
Kemudian untuk Partikel = 100, awal inisialisasinya adalah sebagai berikut :



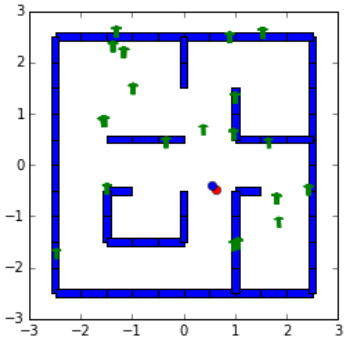
Dan setelah iterasi ke-7, didapatkan konvergensinya. Seperti yang dapat dilihat pada gambar berikut.



Terakhir, dengan jumlah partikel = 150. Berikut adalah inisialisasi awalnya.



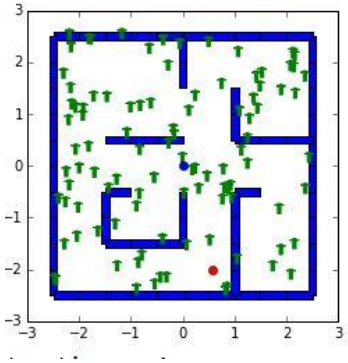
Dan pada iterasi ke-7 sudah didapatkan konvergensinya. Seperti yang dapat dilihat pada gambar berikut.



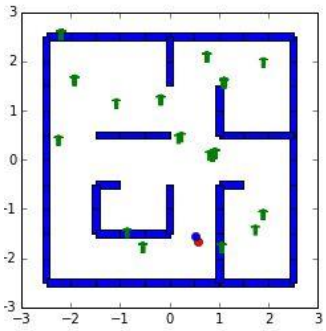
Jadi dapat disimpulkan untuk pergerakan dengan parameter optimal dari sensor dan motion model, pengaruh jumlah partikel yang semakin banyak mempercepat waktu konvergensinya jika dilihat dari jumlah iterasi waktu.

Kemudian dengan variasi iterasi, dengan jumlah partikel = 100 dilakukan variasi

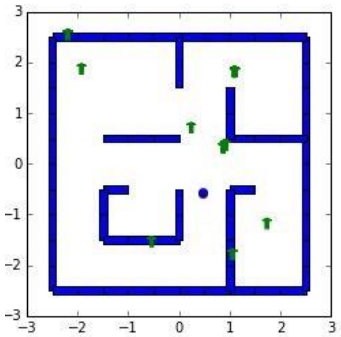
iterasi=1, 5, 10 kali. Berikut hasil dari variasi iterasi=1 kali.



Sedangkan berikut ini adalah hasil dari variasi iterasi = 5 kali:



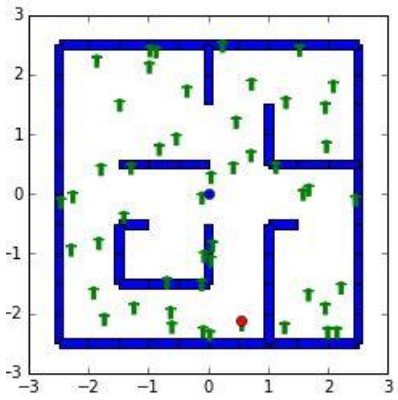
Berikut adalah variasi iterasi = 10 kali :



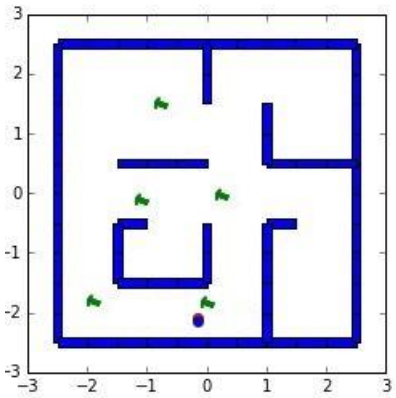
Tentu saja terlihat dengan semakin banyak jumlah iterasi, tingkat kecepatan konvergennya semakin tinggi. Sehingga iterasi yang dipilih adalah sejumlah 10 kali.

Selain itu dilakukan percobaan jika dilakukan gerakan belok ke kiri. Dan coba dibandingkan dengan gerakan lurus ke depan. Berikut adalah perbandingannya. (dengan menggunakan jumlah partikel yg sama yaitu 100 dan iterasi yang sama yaitun10).

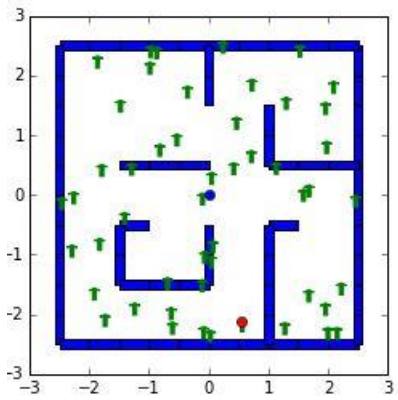
- Gerak belok sewaktu iterasi awal



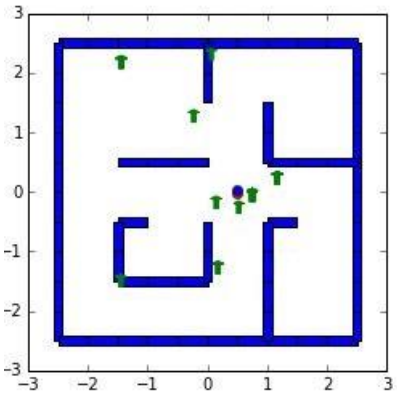
- Gerak belok sewaktu iterasi akhir



- Gerak lurus sewaktu iterasi awal

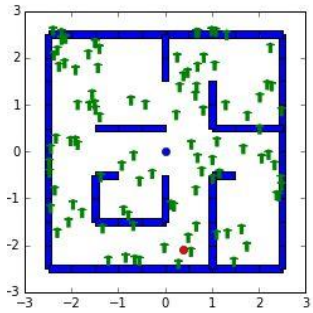


- Gerak lurus sewaktu iterasi akhir

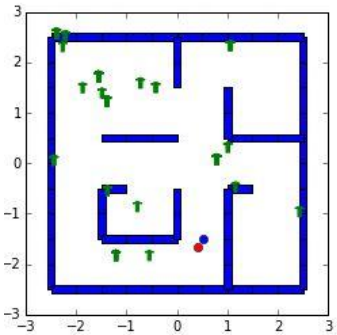


Dapat dilihat bahwa tingkat konvergensi gerak berbelok lebih baik dari gerakan lurus. Kemudian dilakukan lagi variasi terhadap penambahan objek resizeable concrete block di dalam maze. Maka hasilnya adalah sebagai berikut.

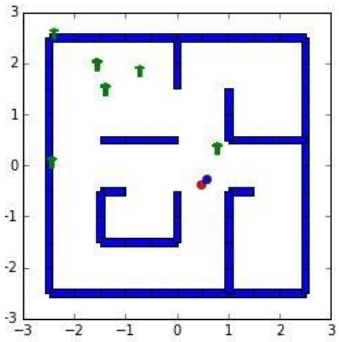
- Dengan objek iterasi pertama



- Dengan objek iterasi kelima



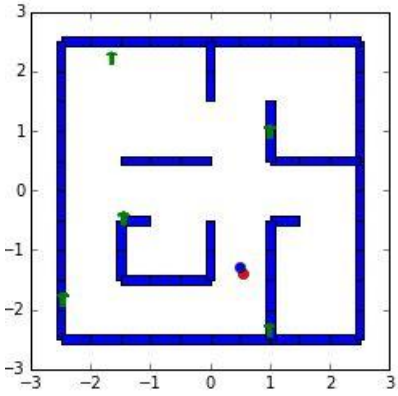
- Dengan objek iterasi kesepuluh



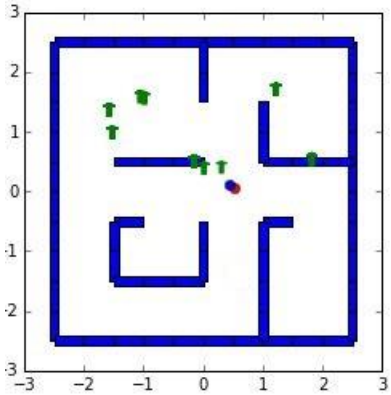
Dapat dilihat bahwa hasil penambahan objek resizeable concrete block membuat proses konvergen pada area sekitar sofa menjadi berkurang tetapi kecepatan konvergensi nya relatif tetap sama.

Dan terakhir variasi waktu timer yang digunakan untuk t=0.5 dan t=1.0 adalah sebagai berikut.

- t=0.5. Capture ketika iterasi ke-10.



- t=1.0. Capture ketika iterasi ke-10.



Dapat dilihat bahwa t yang lebih lama menghasilkan hasil noise yang lebih sedikit karena posisi-posisi dari partikelnya cenderung dekat dengan robot. Sedangkan ketika t=0,5 noisenya relatif lebih banyak.

6. Kesimpulan

Problem dari aspek Localization pada robotika sudah menjadi fokus penelitian dalam dekade terakhir ini, dan solusi dari permasalahan tersebut yang menjanjikan di dalam Localization adalah melalui Probabilistic Robotics melalui salah satu tekniknya yaitu Monte Carlo Localization yang merupakan salah satu turunan dari metode Particle Filer. Untuk menghasilkan model MCL yang optimal dengan tingkat error yang relatif rendah serta kecepatan konvergensi. Dilakukan percobaan-percobaan dan didapatkan kesimpulan data sebagai berikut untuk implementasi MCL :

- *a1=a2=0.01*
- *a3=a4=0.1 & a5=a6=0.1*
- *zhit = 0.2*
- *zshort = 0.3*
- *zmax = 0.25*
- *zrand = 0.25*
- *deltahit = 1*
- *lambdashort = 1*
- *Jumlah sensor = 4 (nomor 1,2,7,8)*
- *Jumlah partikel agar konvergen :* 100

Reference

[1] A. Doucet & A. M. Johanse, “A Tutorial on Particle Filtering and Smoothing: Fifteen years later”, Handbook of Nonlinear Filtering, University of Padova, 2009.

[2] A. Doucet, N. d. Freitas, & N. Gordon, “An introduction to Sequential Monte Carlo Methods”, Springer Publisher New York, 2001.

[3] A. B. Owen, “Monte Carlo: a tutorial”, Stanford University, MCQMC’12 Sydney Australia, 2012.

[4] D. Fox, W. Burgard, F. Dellaert, & S. Thrun, “Monte Carlo Localization: Efficient Position Estimation for Mobile Robots”, Proceedings of AAAI, 1999.

[5] F. Dellaert, D. Fox, W. Burgard, & S. Thurn, “Monte Carlo Localization for Mobile Robots”, Proceedings of Robotics and Automation, Detroit, 1999.

[6] S. Thurn, W. Burgard, & D. Fox, “Probabilistic Robotic”, The MIT Press , Cambridge, Masachusetts, 2006.

[7] S. Thurn, D. Fox, W. Burgard, & F.Dellaert, “Robust Monte Carlo Localization for Mobile Robots”, Artificial Intelligence – Elsevier, 2001.

[8] G. N. DeSouza & A. C. Kak, “Vision for Mobile Robot Navigation: A Survey”, IEEE Transactions on Pattern Analysis and Machine Intelliegence, Vol. 24, 2002.

[9] J. F. Villaverde & J. F. R. Ramfrez, “Sequential Monte Carlo Filterint: an Example”, University of Pennsylvania, 2004

[10] R. Siegwart & I. R. Nourbakhsh. “Introduction to Autonomus Mobile Robots”, The MIT Press, 2004.