TELEKONTRAN, VOL. 9, NO. 2, OKTOBER 2021
DOI : 10.34010/telekontran.v9i2.5579
p-ISSN : 2303 – 2901
e-ISSN : 2654 – 7384

Metode *Improved Gaussian Sampling* pada Algoritma *Rapidly Exploring Random Tree**

Improved Gaussian Sampling for Rapidly Exploring Random Tree Algorithm*

Muhammda Aria Rajasa Pohan[*], Habiba Octaviani Pakaya
Program Studi Teknik Elektro, Fakultas Teknik dan Ilmu Komputer
Universitas Komputer Indonesia Jl. Dipati ukur No 112, Bandung
[*]Email : muhammad.aria@email.unikom.co.id

Abstrak - Robot otonom adalah sebuah perangkat mekanis yang mampu bergerak bebas di lingkungan yang terdapat rintangan, dapat menjalankan berbagai jenis fungsi, mampu menyimpan program yang telah dibuat dan memperoleh informasi lingkungan melalui sensor yang dimiliki. Pada proses pengendalian *autonomous robot* dibutuhkan perencanaan jalur agar robot mampu mencapai tujuan yang dituju dengan jalur yang dilewati bebas tabrakan. Penelitian ini bertujuan merancang metode *improved gaussian sampling* dan mengimplementasikan pada algoritma Rapidly Exploring *Random* Tree* (RRT*). Perancangan yang dilakukan menggunakan bahasa pemograman *Laboratory Virtual Instrumentation Engineering Workbench* (LabVIEW) dengan metode *improved gaussian sampling, gaussian sampling* dan *random sampling*. *Improved gaussian sampling* merupakan pengembangan dari metode *gaussian sampling* dengan menambahkan jumlah sampling. Jumlah *sampling* yang digunakan pada metode ini yaitu sejumlah 10 *sampling*. Untuk mengukur performansi dari metode *sampling* yang diusulkan, maka kami melakukan perbandingan performansi metode *improved gaussian sampling*, *gaussian sampling* dan *random sampling*. Disimpulkan bahwa pada algoritma RRT* dengan menggunakan metode *improved gaussian sampling* menghasilkan performansi yang lebih baik dibandingkan algoritma RRT* yang menggunakan metode *sampling* lainnya. Hasil perbandingan pengukuran berdasarkan nilai sampling didapatkan rata-rata nilai *path cost* rata-rata 10,12 dengan jumlah sampling hanya 1 dan nilai *path cost* terpendek 8,9 dengan jumlah sampling 10. Dari hasil perancangan algoritma pada bahasa pemograman LabVIEW dapat disimulasikan algoritma RRT* dengan berbagai metode sampling yang diujikan.

Kata kunci *:* *Gaussian sampling, improved gaussian sampling, path cost,* RRT*, *sampling,* waktu komputasi

Abstract *- An autonomous robot is a mechanical device that is able to move freely in an environment where there are obstacles, can perform various types of functions, is able to store programs that have been created and obtain environmental information through its sensors. In the process of controlling an autonomous robot, it is necessary to plan a path so that the robot is able to reach its intended destination with a collision-free path. In this paper a method called improved gaussian sampling design and implementation using the Rapidly Exploring Random Tree* (RRT*) algorithm. The design of algotihm using Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW)) with improved gaussian sampling method, gaussian sampling and random sampling. Improved gaussian sampling is the development of the gaussian sampling method by adding the number of sampling. The number of sampling used in this method using 10 sampels. To measure the performance of the proposed sampling method, we compare the performance of the improved gaussian sampling, gaussian sampling and random sampling methods. The results of the tests carried out showed that the RRT* algorithm using the improved Gaussian sampling method was more optimal than using other methods. The results of the comparison of measurements based on the sampling value obtained an average path cost value of 10.12 with the number of sampling only 1 and the shortest path cost value of 8.9 with a sampling number of 10. From the results of the algorithm design in the LabVIEW programming language, the RRT* algorithm can be simulated with various sampling methods tested.*

Keywords *: Computational time, gaussian sampling, improved gaussian sampling, path cost, RRT*, sampling*

I. PENDAHULUAN

A. Latar Belakang

Perencanaan jalur pada robot dapat bermanfaat pada berbagai bidang diantaranya bidang kimia protein *folding* [1] virtual prototype pada manufaktur [2] dan sistem transportasi [3]. Perencanaan jalur dibagi menjadi dua kelompok yaitu perencanaan jalur berbasis grafik dan berbasis *sampling*. Salah satu contoh perencanaan jalur berbasis grafik yang sering digunakan yaitu algoritma Dijkstra [4] dan A* [5]. Algoritma pencarian A* memiliki kelebihan dalam mencari jalur terpendek dengan perhitungan terkecil pada jalur pencarian [4]. Perencanaan jalur berbasis *sampling* biasa disebut dengan *Sampling Based Planning* (SBP). Algoritma SBP memiliki keunggulan dalam perencanaan jalur yang lebih singkat pada jalur yang sulit. Algoritma SBP yang paling sering digunakan adalah algoritma PRM (*Probabilistic Roadmap Planners*) [6] dan algoritma *Rapidly Exploring Random Tree* (RRT) [7]. Tahun 1998 algoritma RRT diperkenalkan oleh LaValle dan Kuffner [7] yang merupakan salah satu algoritma popular dan sering digunakan hingga saat ini. Tahun 2011 RRT dikembangkan menjadi versi lebih optimal oleh Karaman dan Frazolli yang disebut dengan *Rapidly Exploring Random Tree Star (*RRT*) yang bertujuan untuk mencapai jalur terpendek dan waktu komputasi mencapai *goal* tercepat [8]. RRT* ialah algoritma hasil pengembangan yang lebih optimal dari algoritma RRT. Berbeda dengan algoritma RRT, algoritma RRT* lebih mempertimbangkan semua *node* terdekat dari q_{new} dan selalu mengecek *cost* terkecil dari masing-masing q_{near} tersebut.

Pada algoritma SBP terdapat strategi perencanaan pengambilan sampel untuk mempersempit penjelajahan ruang pencarian jalur. Terdapat beberapa strategi pengambilan sampel yang populer diantaranya *boundary sampling* [9], *goal biassing sampling* [10] dan *gaussian sampling* [11]. *Gaussian sampling* atau bisa disebut juga distribusi normal merupakan jenis *sampling* yang mengambil sample secara acak didaerah sekitar sisi *obstacle*. Distribusi *gaussian* atau bisa dikenal dengan distribusi normal merupakan fungsi probabilitas yang menunjukan penyebaran suatu variable acak yang diindeks berdasarkan nilai waktu dan ruang konfigurasi. Distribusi *gaussian* diambil dari nama penemunya yaitu Carl Friederich Gauss seorang matematikan asal Jerman yang mengembangkan teori distribusi pada periode 1794 – 1809 [11]. Pada metode *gaussian sampling* jumlah *sampling* yang digunakan setiap iterasi hanya sebanyak satu buah *sampling*. Sehingga membutuhkan banyak jumlah iterasi yang digunakan untuk mencapai titik goal dan waktu komputasi yang dibutuhkan semakin lama. Maka pada penelitian ini, penulis merancang sebuah metode peningkatan dari metode *gaussian sampling* yaitu metode *improved gaussian sampling*. Metode *improved gaussian sampling* memiliki kelebihan dalam pengambilan jumlah *sampling* yang digunakan yaitu sebanyak 10 *sampling* nilai titik sampling ini diambil dengan menggunakan metode *box muller*. Metode transformasi *box muller* dikemukakan oleh George E. P dan Mervin E. Muller pada tahun 1958 [12]. Dengan adanya peningkatan metode ini maka diharapkan proses pencarian jalur akan lebih optimal dan waktu komputasi akan lebih cepat pada algoritma RRT*.

B. Tinjauan *State of Art*

Pengambilan sampel adalah inti proses penelitian ini, awalnya proses *sampling* diusulkan pada algoritma PRM dan PRC dengan menggunakn *uniform sampling* [13]. Namun, *uniform sampling* memiliki kelemahan yaitu nilai probabilitas yang rendah dalam mengambil *sample* dilingkungan konfigurasi yang luas tetapi titik sampling yang digunakan akan secara acak dan cenderung menjauhi titik *goal*. Hal ini terjadi karena *uniform sampling* memiliki probabilitas seragam pada seluruh ruang konfigurasi untuk dijadikan *sample* [14]. Kelemahan dari proses pengambilan *sampling* ini dikembangkan menjadi beberapa metode *sampling* salah satunya adalah *gaussian sampling.*

Valérie telah mempublikasikan metode strategi *gaussian sampling* pada algoritma PRM tetapi dibuat dengan menggunakan program MATLAB [11]. Lars juga telah mempublikasikan perencanaan jalur pada algoritma *probabilistic path planning* dengan menggunakan distribusi normal pada bahasa C+ [12]. Tetapi belum adanya perencanaan jalur RRT* yang menggunakan metode *improved gaussian sampling* pada bahasa pemograman LabVIEW. Maka pada peneliti ini akan mengimplementasikan perencanaan jalur RRT* dengan metode pengambilan *sampling* secara *improved gaussian sampling* pada bahasa pemograman LabVIEW.

C. Tujuan

Berdasarkan pada latar belakang yang telah dikemukakan, maka peneliti memiliki tujuan dalam penelitian yaitu merancang metode

improved gaussian sampling yang dapat menghasilkan *path cost* dan waktu komputasi lebih baik dari metode *gaussian sampling* pada algoritma RRT* pada bahasa pemrograman LabVIEW, membandingkan performansi algoritma RRT* yang menggunakan metode *improved gaussian sampling* terhadap algoritma RRT* yang menggunakan metode *gaussian sampling* dan *random sampling* dan membandingkan pengukuran performansi sistem algoritma RRT* metode *improved gaussian sampling* terhadap jumlah *sampling*.

D. Sistematika Pembahasan

Makalah ini menyajikan beberapa bagian. Bagian 2 akan menjelaskan proses perancangan algoritma RRT* metode improved gaussian sampling. Bagian 3 menjelaskan hasil pengujian dan analisa dari seluruh pengujian. Bagian terakhir menjelaskan seluruh kesimpulan dan saran dari makalah ini.

II. METODOLOGI

Algoritma RRT adalah struktur data yang dirancang secara efisien untuk mencari ruang bebas tabrakan. RRT seperti cabang pada pohon, dimana algoritma ini menumbuhkan setiap cabang yang akan berakar pada konfigurasi awal dengan menggunakan sampel acak pada ruang pencarian. Jika koneksi antara cabang dengan sampel acak tersebut melewati ruang bebas dan bebas hambatan, maka cabang ini dapat ditambahkan ke cabang pohon baru. Cabang pohon algoritma RRT akan terus bertumbuh ke segala arah yang tidak terhalang oleh collsion hingga menemukan titik tujuan. Jalur yang dilewati cabang pohon pertama merupakan jalur terpendek untuk mencapai titik tujuan tersebut. Algortima RRT dimulai dengan inisialisasi *node start* (q_{start}) pada daerah ruang konfigurasi bebas (C_{free}). Setiap iterasi terdapat *node random* (q_{rand}) yang akan berpindah-pindah pada seluruh daerah konfigurasi C_{free}. Setiap perpindahan q_{rand} akan dilakukan pengecekan *collison* disekitar q_{rand}. Jika q_{rand} terdapat pada titik bebas *obstacle* maka akan dipilih *node* terdekat dari pohon pencarian yang dinamakan $q_{nearest}$. Selanjutnya dibuat cabang baru dari q_{rand} menuju $q_{nearest}$ sejauh *expand distance* (Δq). Cabang baru yang telah dihubungkan tersebut dinamakan *new node* (q_{new}). Lalu algoritma RRT akan terus berlanjut ke iterasi selanjutnya hingga menemukan titik *goal node*. Pohon pencarian pada algoritma RRT lebih cenderung mengeksplore wilayah konfigurasi yang belum pernah dijelajahi. Waktu komputasi yang dibutuhkan pohon pencarian RRT juga lebih singkat dikarenakan algoritma RRT tidak mempertimbangkan jalur terpendek menuju ke *goal node*.

Algoritma RRT* memiliki penambahan dua subprogram yang berbeda dengan algoritma RRT diantaranya subprogram *choose parent* dan subprogram *rewire*. Seperti yang dijelaskan sebelumnya algoritma RRT* sangat mempertimbangkan semua *node* terdekat, proses pemilihan *node* yang lebih optimal ini berada pada program *choose parent*. Algoritma inti RRT* dengan menggunakan metode sampling secara acak pada ruang konfigurasi ditunjukkan pada **Algorithm.** Algoritma RRT* dengan menggunakan metode *improved gaussian sampling* dapat dilihat pada **Algorithm 1.**

Algorithm : $T = (V, E) \leftarrow RRT^*(q_{init})$
$T \leftarrow InitializeTree()$
$T \leftarrow InsertNode(\emptyset, q_{init}, T)$
for $k \leftarrow 1$ **to** N **do**
$q_{rand} \leftarrow RandomSample(k)$
$q_{nearest} \leftarrow NearestNeighbor(q_{rand}, Q_{near}, T)$
$q_{new} \leftarrow Steer\ (q_{nearest}, q_{rand}, \Delta q)$
if $Obstaclefree(q_{new}, q_{nearest})$ **then**
$Q_{near} \leftarrow Near(T, z_{new})$
$q_{parent} \leftarrow ChooseParent(q_{new}, Q_{near}, q_{nearest})$
$T \leftarrow InsertNode(q_{parent}, q_{new}, T)$
$T \leftarrow Rewire(T, Q_{near}, q_{parent}, q_{new})$
end

Algorithm 1 : $T = (V, E) \leftarrow RRT^*(q_{init})$
Input :
q_{start} : the configuration where the tree is rooted
n : maximum iteration
Output : A roadmap $T = (V,E)$

1. $T \leftarrow InitializeTree\ ()$
2. $T \leftarrow InsertNode\ (\emptyset, q_{init}, T)$
3. *For* $k \leftarrow 1$ *to* N *do*
4. $q_{gauss} \leftarrow ImprovedGaussianSample\ (C1, C2)$
5. $q_{nearest} \leftarrow NearestNeighbor\ (q_{gauss}, Q_{near}, T)$
6. $q_{new} \leftarrow Steer\ (q_{nearest}, q_{gauss}, \Delta q)$
7. **If** $ObstacleFree\ (q_{new}, q_{nearest})$ **then**
8. $Q_{near} \leftarrow Near\ (T, z_{new})$
9. $Q_{min} \leftarrow ChooseParent\ (q_{new}, Q_{near}, q_{nearest})$
10. $T \leftarrow InsertNode\ (q_{min}, q_{new}, T)$
11. $T \leftarrow Rewire\ (T, q_{near}, q_{min}, q_{new})$
12. **end**

Dapat terlihat pada **Algorithm 1** di bawah terdapat penambahan beberapa sub program. Penambahan subprogram *improved gaussian sample* digunakan untuk mendapatkan nilai *sampling* yang selanjutnya akan dijadikan *random*

node. Pada proses *choose parent* akan mencari *node-node* terdekat dari q_{new}, lalu *node* terdekat yang ini dinamakan *node* q_{near}. Jalur yang memiliki *cost* terkecil antara Q_{near} dengan q_{new} memiliki kemungkinan untuk dijadikannya *node parent* dari q_{new}. Algoritma subprogram *choose parent* pada RRT* ditunjukkan pada **Algorithm 2.** Jika pemilihan *parent* dilakukan, maka akan dicari *node-node* terdekat dengan q_{new} lalu dihubungkan antara *node* awal hingga q_{new} akan diperoleh jalur yang lebih dekat. Program perubahan *path planning* ini terjadi pada proses algoritma *rewire* (algoritma 5) yang dapat dilihat pada **Algorithm 3.**

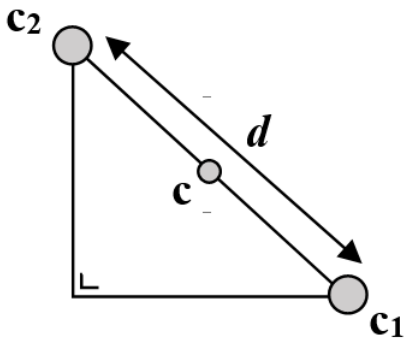
Pada **Algorithm 3** subprogram *rewire* pada perancangan algoritma RRT* terjadi proses pencarian *node-node* terdekat dengan *node* q_{new}, jika *node* antara q_{new} melalui *node* awal maka akan dihasilkan jalur yang lebih singkat. Sebelum proses *rewire* jalur yang dihasilkan akan lebih berliku-liku, namun setelah dilakukan proses *rewire* jalur yang dihasilkan akan lebih optimal. Titik *sampling* yang diambil berada diantara titik c_1 dengan c_2. Ilustrasi pengambilan titik *sampling* dapat dilihat pada **Gambar 1.**

Algorithm 2: $q_{min} \leftarrow ChooseParent\ (q_{gauss}, Q_{near}, q_{nearest}, \Delta q)$

13. $q_{min} \leftarrow q_{nearest}$
14. $c_{min} \leftarrow Cost\ (q_{nearest}) + c\ (q_{gauss})$
15. **for** $q_{near} \in Q_{near}$ **do**
16. $q_{path} \leftarrow Streer\ (q_{near}, q_{gauss}, \Delta q)$
17. **if** *ObstacleFree* (q_{path}) *then*
18. $c_{new} \leftarrow Cost\ (q_{near}) + c\ (q_{gauss})$
19. **If** $c_{min} < c_{new}$ **then**
20. $c_{min} \leftarrow c_{new}$
21. $q_{min} \leftarrow q_{new}$
22. **end**
23. **end**
24. **end**
25. **Return** q_{min}

Algorithm 3: $T \leftarrow Rewire\ (T, Q_{near}, q_{min}, q_{gauss})$

26. **for** $q_{near} \in Q_{near}$ **do**
27. $q_{path} \leftarrow Steer\ (q_{near}, q_{gauss}, \Delta q)$
28. **if** *ObstacleFree* (q_{path}) *then*
28. $Cost\ (q_{gauss}) + c\ (q_{path}) < Cost\ (q_{near})$ **then**
29. $T \leftarrow ReConnect\ (q_{gauss}, q_{near}, T)$
30. **end**
31. **return** T



Gambar 1. Ilustrasi pengambilan titik *sampling*

Pada **Gambar 1** di atas dapat terlihat ilustrasi proses pengambilan *sampling* dengan metode *gaussian*. Mula-mula pada subprogram ini diinisialisakan sebuah *random node* yang diberi nama c_1 pada C_{free}. Selanjutnya peneliti menentukan nilai jarak yang akan digunakan pada distribusi normal dengan nama d. Dengan rumus phytagoras ditariklah garis dari titik c_1 sejauh d yang akan menghasilkan titik baru yang dinamakan titik c_2. Metode *box muller* ini menentukan nilai rata-rata sample acak dari distribusi *gaussian* standar. Persamaan *box muller* secara umum terdapat dua parameter dapat dilihat pada persamaan berikut ini.

$$x_1 = \sqrt{-2ln(R_1)cos(2\pi R_2)} \quad (1)$$

dan

$$x_2 = \sqrt{-2ln(R_1)sin(2\pi R_2)} \quad (2)$$

Sebuah *sampling* dapat disebut metode *gaussian* apabila posisi c_1 dan c_2 yang terhubung sejauh d, salat satu titik nya berada didalam *obstacle* dan titik lainnya berada diruang bebas *obstacle* (baris 18 algoritma 2). Ketika titik c_1 yang berada di luar bebas *obstacle* maka c_1 dapat dimasukan ke graph. Begitupun sebaliknya jika c_2 yang berada di ruang bebas *obstacle* maka titik c_2 yang akan ditambahkan ke graph (baris 19 algoritma 2). Pada metode *gaussian* titik *sampling* yang dapat diambil sebagai *node random* terdapat pada titik-titik antara jarak c_1 dengan c_2. Nilai titik *random* diantara garis c_1 dan c_2 dinotasikan sebagai c. Algoritma *improved gaussian sampling* ditunjukkan pada **Algorithm 4.**

Penentuan nilai *gaussian random sampling* pada bahasa pemograman LabVIEW peneliti menggunakan metode transformasi *box muller*. Algoritma *box muller* dapat dilihat pada **Algorithm 5**. Selanjutnya algoritma RRT* *box muller* pada **Algorithm 5** akan dimasukkan kedalam perancangan program inti RRT* untuk membangkitkan nilai *gaussian random number*

sebanyak 10 titik yang berada diantara titik c_1 dan c_2.

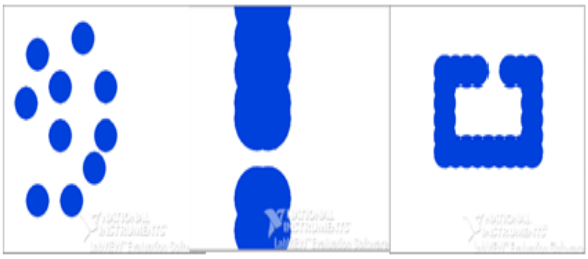
Algorithm 4: $q_{gauss} \leftarrow ImprovedGaussianSample\ (c_1, c_2, c)$

```
loop
    c1 ← a random configuration
    d ← a distance choosen according to a normal distribution
    c ← range where the point c is the centre d
    c2 ← a random conf. at distance d from c1
    c = 10 a number of sample distribussion
    if c1 ∈ Cfree and c2 ∉ Cfree then
        add c1 to the graph
        c ← BoxMuller (g1, g2)
        add c sample to the graph
    else if c2 ∈ Cfree and c1 ∉ Cfree then
            add c2 to the graph
            c ← BoxMuller (g1, g2)
            add c sample to the graph
        else
                discard both
```

Algorithm 5: $c \leftarrow$ BoxMuller (g_1, g_2)

```
c = Randomly 10 sample a bit according
    to the density of obstacle in the map space
Take d = 2
x ← g1, y ← g2
until 0 < d < 2
    a ←√(−2lnU) and b ← 2πnU
return g1 ← a sin b, g2 ← a cos b
```

Pengujian algoritma dilakukan pada 3 lingkungan yang berbeda yaitu pada lingkungan *clutter* yang memiliki jenis rintangan terdapat berbagai rintangan pada ruang konfigurasi, lingkungan *trap* yaitu rintangan berupa celah sempit diantara 2 *obstacle*. Lingkungan yang diujikan terakhir yaitu lingkungan trap berupa rintangan. Fungsi dari setiap sub-program algoritma inti RRT* *improved gaussian sampling* terdiri dari 8 sub-program inti dapat dilihat pada **Tabel I.**



(a) (b) (c)

Gambar 2. Pengujian Algoritma pada linkungan (a) Cluter, (b) Narrow, dan (c) Trap

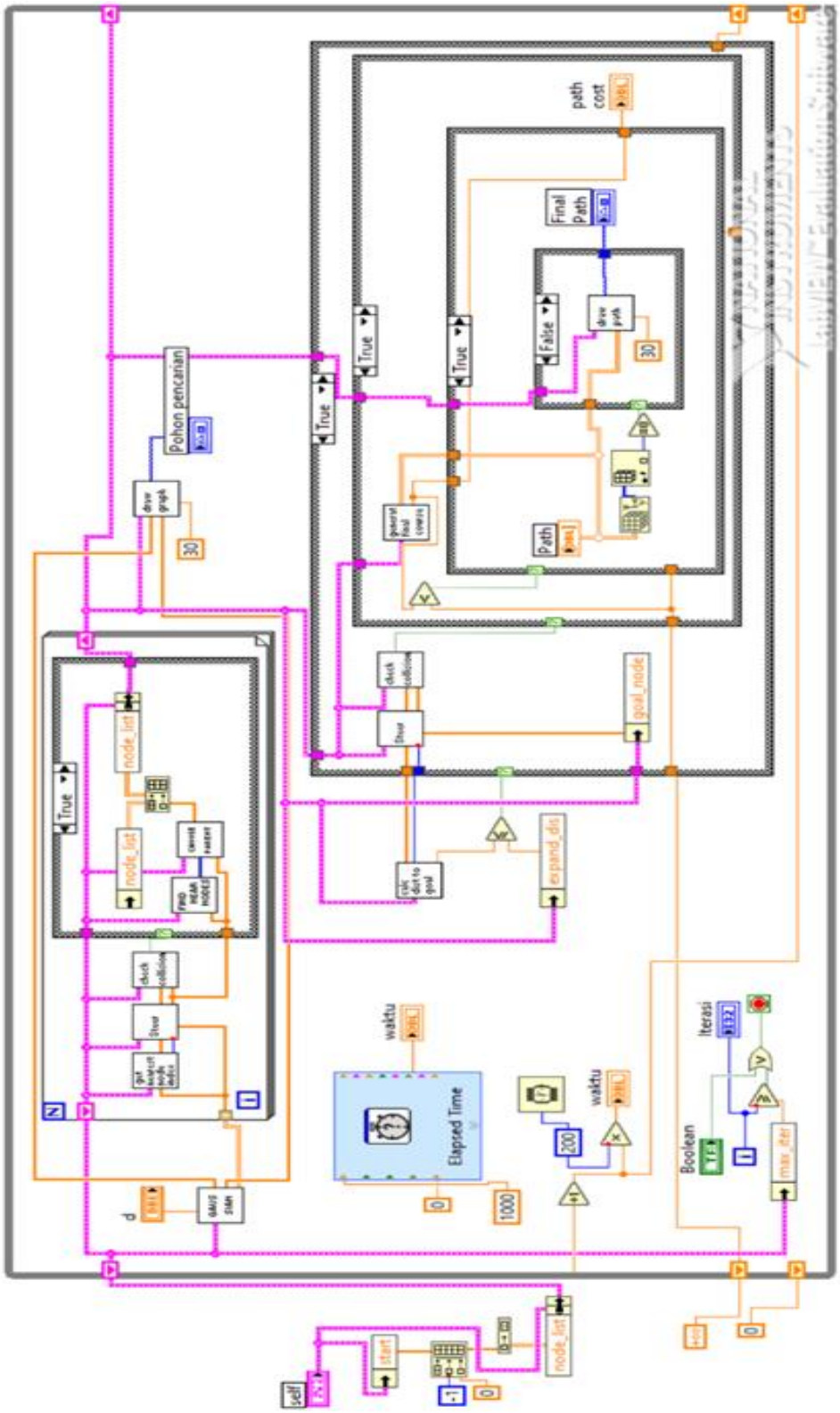
Tabel IX. Fungsi sub program pada RRT* *improved gaussian sampling*

No	SubProgram	Fungsi
1	*Gaussian sampling*.vi	Mencari nilai *gaussian random sampling* pada ruang konfigurasi.
2	*Gaussian White Noise.vi*	Proses *box muller* atau proses mendapatkan *gaussian random number*.
3	*Choose parent*.vi	Pemilihan *gaussian random sampling node* yang dapat dijadikan *node parent* (q_{parent}) yang selanjutnya akan ditumbuhkan cabang baru.
4	*Get Nearest Node.vi*	Melakukan pemilihan *node* yang dapat dijadikan q_{parent}.
5	*Check Collision.vi*	Pengecekan apakah pada pohon pencarian RRT* terdapat *collision*
6	*Check Obstacle.vi*	Pengecekan proses *gaussian sampling* apakah titik c_1 atau c_2 berada didalam *obstacle* atau tidak
7	*Draw Graph.vi*	Visualisasi lingkungan yang akan dilalui dan menampilkan proses RRT* setiap iterasi nya
8	*Draw Path.vi*	Menampilkan jalur yang dihasilkan pada algoritma RRT* yang divisualisasikan menggunakan garis hitam tebal

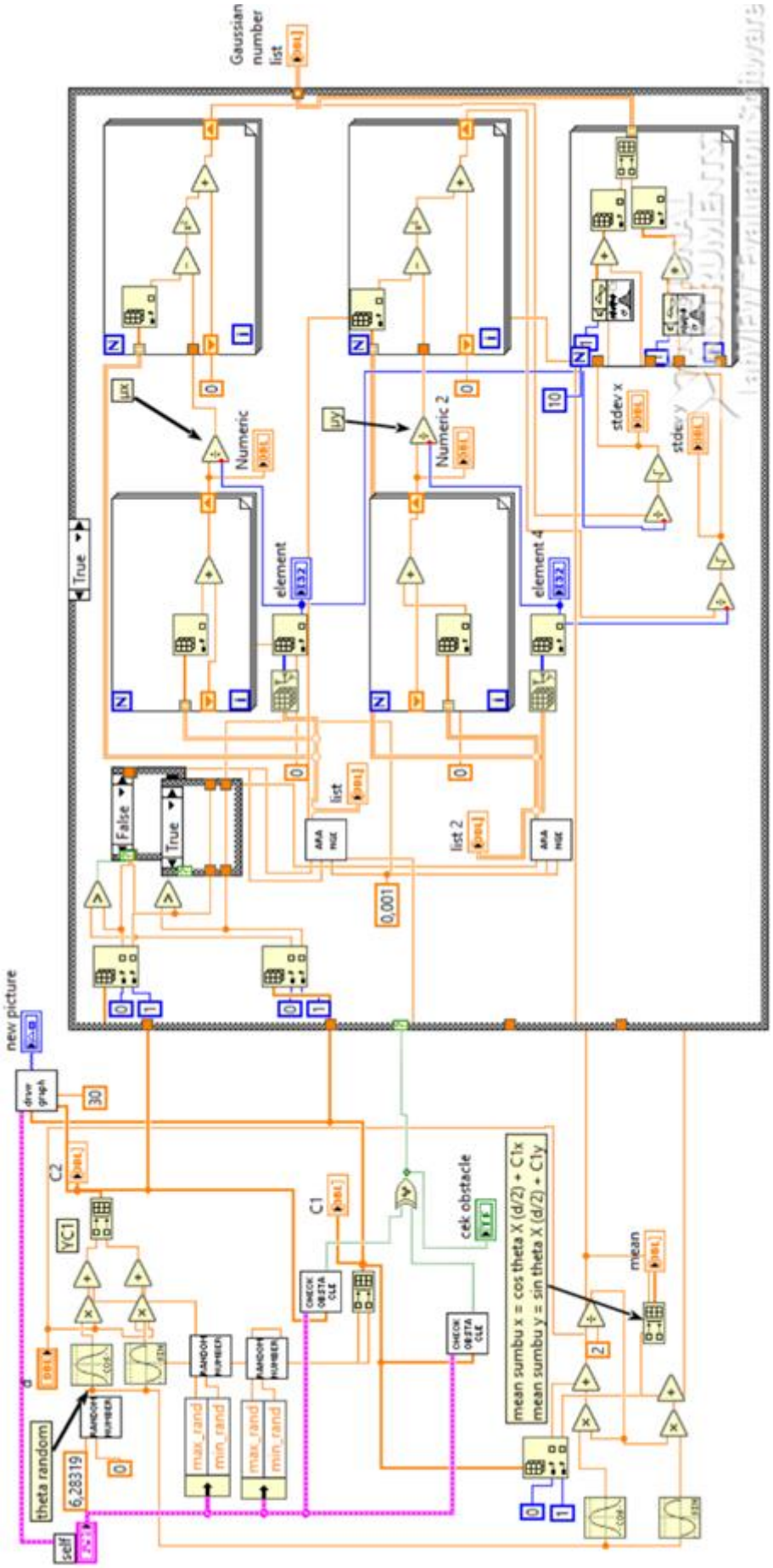
III. HASIL DAN PEMBAHASAN

Untuk proses awal titik *start* dibuat dengan membuat titik berwarna hijau sedangkan untuk titik tujuan dibuat berwarna. Selanjutnya penempatan titik secara *rand*om pada ruang konfigurasi jalur dibuat dengan menggunakan titik c_1. Peneliti dapat menempatkan *obstacle* pada seluruh permukaan konfigurasi C_{free} dengan bebas. Ruang konfigurasi yang terdapat *obstacle* diwarnai dengan warna biru. Daerah yang dapat dilewati oleh algoritma RRT* memiliki warna putih. Pengujian algoritma dilakukan dengan membandingkan performasi algoritma RRT* antara metode *improved gaussian sampling*, *gaussian sampling* dan *random sampling*. Metode

dengan *random sampling* merupakan hasil perancangan algoritma RRT* pada bahasa pemograman LabVIEW. Hasil perancangan block diagram RRT* metode *improved gaussian sampling* pada LabVIEW ditunjukkan pada **Gambar 3.** Block diagram berupa proses *sampling* pada RRT* dengan menggunakan metode *improved gaussian sampling* dapat dilihat pada **Gambar 4.**



Gambar 3. Block diagram RRT* *improved gaussian sampling*



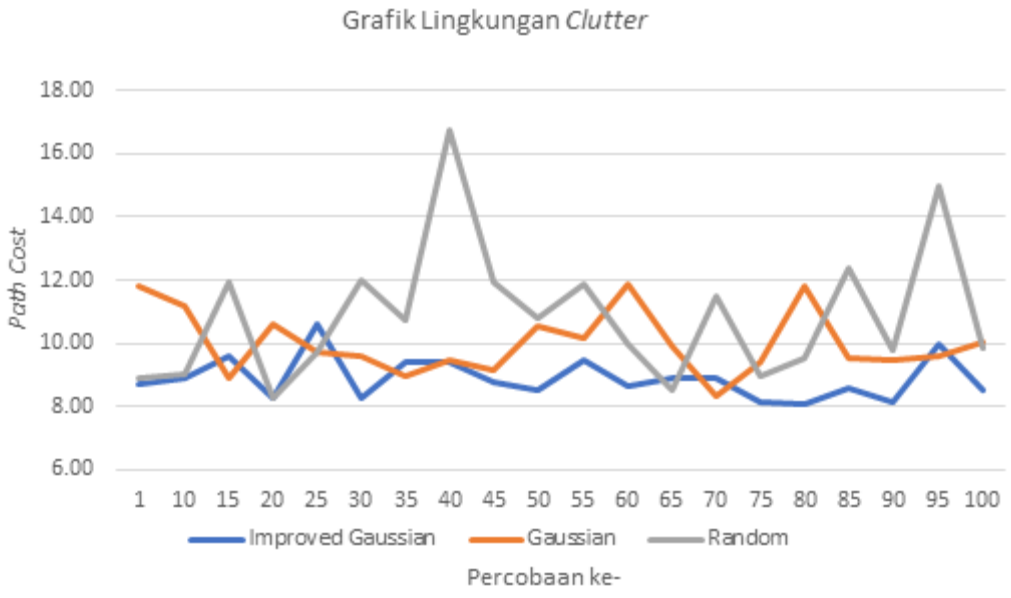
Gambar 4. Block diagram *Improved gaussian sampling*

A. Perbandingan *Improved Gaussian Sampling*, *Gaussian Sampling* dan *Random Sampling*

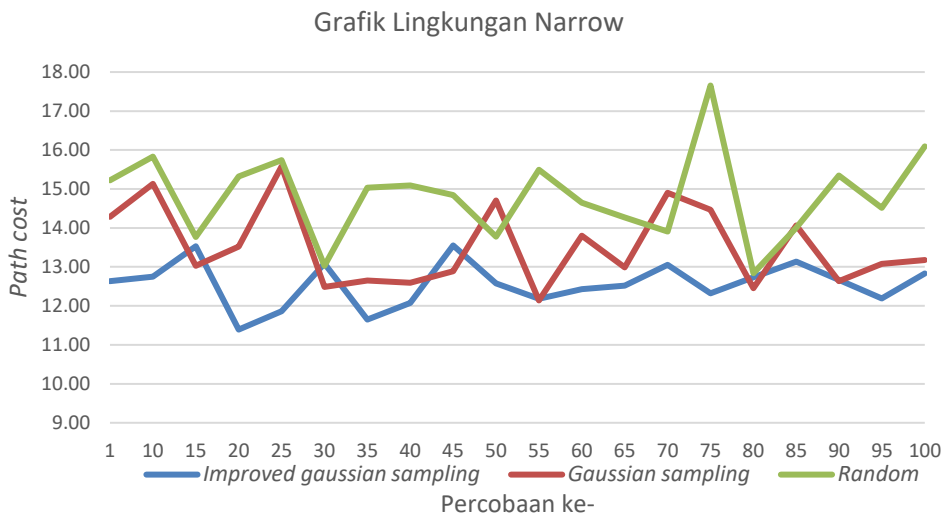
Perbandingan pengujian algoritma RRT* dilakukan pada berbagai lingkungan yaitu *clutter*, *narrow* dan *trap*. Perbandingan kinerja pengambilan data yang dilakukan yaitu jarak dari nilai *path cost* dan waktu komputasi yang dibutuhkan untuk mencapai titik goal dalam hitungan detik. Pengujian dilakukan sebanyak 100 kali pada setiap lingkungan yang diujikan. Pada pengujian ini peneliti mengambil sendiri seluruh data yang diambil pada metode *improved gausian sampling*, *gaussian sampling* dan *random sampling*. Lalu, diambil nilai terbaik, terburuk dan rata-rata (*mean*) dari seluruh pengujian. Hasil perbandingan ini akan dianalisis untuk melihat apakah telah sesuai dengan tujuan yang diinginkan. Hasil perbandingan algoritma RRT* dengan menggunakan metode *improved gaussian sampling*, *gaussian sampling* dan *random sampling* pada berbagai lingkungan dapat terlihat pada **Tabel II.** Grafik hasil pengukuran perbandingan *improved gaussian sampling*, *gaussian sampling* dan *random sampling* pada berbagai lingkungan dapat dilihat pada **Gambar 5** yaitu lingkungan *clutter*, **Gambar 6** yaitu lingkungan *narrow* dan **Gambar 7** yaitu lingkungan *trap*.

Tabel II. Perbandingan *Improved Gaussian Sampling*, *Gaussian Sampling* dan *Random Sampling pada berbagai lingkungan*

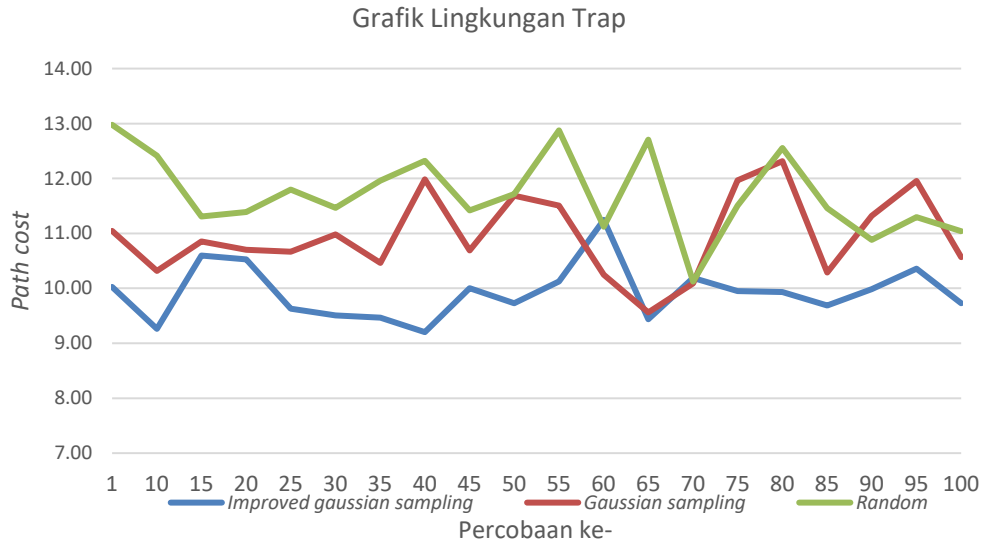
	Jenis Obstacle	*Improved Gaussian*		*Gaussian Sampling*		*Random*	
		Path cost	*Time* (s)	*Path cost*	*Time* (s)	*Path cost*	*Time* (s)
Terbaik	*Clutter*	8,09	40	8,92	40,3	8,28	0,01
Terburuk		10,59	40,18	11,82	40,2	16,73	0,08
Mean		8,89	40,05	10	40,09	10,86	0,03
STD		0,64	0,05	0,99	0,06	2,09	0,04
Terbaik	*Narrow*	11,39	40	12,45	40,01	13,02	0,03
Terburuk		13,53	40,04	15,57	40,04	17,65	0,84
Mean		12,58	40,03	13,53	40,03	14,82	0,25
STD		0,55	0,01	0,99	0,01	1,09	0,28
Terbaik	*Trap*	9,20	40	9,56	40,03	10,12	0,01
Terburuk		11,24	40,1	12,31	40,18	12,97	0,79
Mean		9,93	40,04	10,95	40,12	11,71	0,21
STD		0,48	0,03	0,72	0,05	0,71	0,25



Gambar 5. Perbandingan panjang jalur yang dihasilkan dengan nilai percobaan pada algoritma RRT* *improved gaussian*, *gaussian* dan *random* pada lingkungan *clutter*

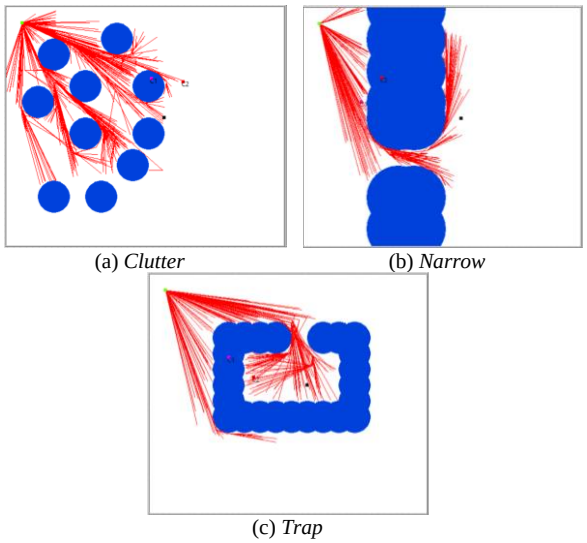


Gambar 6. Perbandingan panjang jalur yang dihasilkan dengan nilai percobaan pada algoritma RRT* *improved gaussian*, *gaussian* dan *random* pada lingkungan *narrow*



Gambar 7. Perbandingan panjang jalur yang dihasilkan dengan nilai percobaan pada algoritma RRT* *improved gaussian*, *gaussian* dan *random* pada lingkungan *trap*

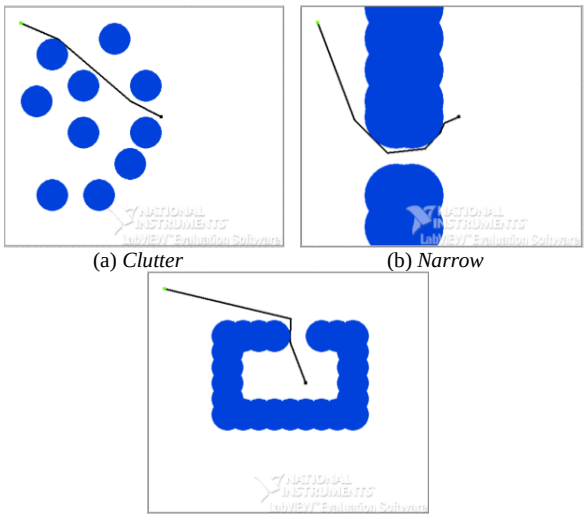
Pada **Tabel II** dapat terlihat bahwa nilai *path cost* pada algoritma RRT* metode *improved gaussian sampling* pada berbagai lingkungan memiliki nilai yang lebih kecil dibandingkan dengan metode *gaussian sampling* dan *random sampling*. Begitu pun dapat terlihat pada nilai waktu komputasi yang dibutuhkan untuk mencapai *goal node*, waktu komputasi dengan metode *improved gaussian sampling* lebih cepat dibandingkan dengan metode *gaussian sampling*. Namun pada bagian waktu komputasi didapatkan waktu komputasi *random sampling* lebih cepat dibandingkan dengan metode *improved gaussian sampling*. Berdasarkan jurnal Jauwaria Nasir dkk pada pengujian RRT* *random sampling* didapatkan bahwa RRT* memiliki nilai komputasi tercepat [16]. Hal ini dikarenakan algoritma RRT* metode *random sampling* melakukan proses *sampling* secara *random* diseluruh ruang konfigurasi pencarian. Hasil pohon pencarian RRT* metode *improved gaussian sampling* pada berbagai lingkungan (a) *clutter* (b) *narrow* (c) *trap* ditunjukkan pada **Gambar 8.** Visualisasi hasil *final path* RRT* metode *improved gaussian sampling* pada berbagai lingkungan (a) *clutter* (b) *narrow* (c) *trap* ditunjukkan pada **Gambar 9.** Hasil pohon pencarian RRT* metode *gaussian sampling* pada berbagai lingkungan (a) *clutter* (b) *narrow* (c) *trap* ditunjukkan pada **Gambar 10.** Visualisasi hasil *final path* RRT* metode *gaussian sampling* pada berbagai lingkungan (a) *clutter* (b) *narrow* (c) *trap* ditunjukkan pada **Gambar 11.**



(a) *Clutter* (b) *Narrow*

(c) *Trap*

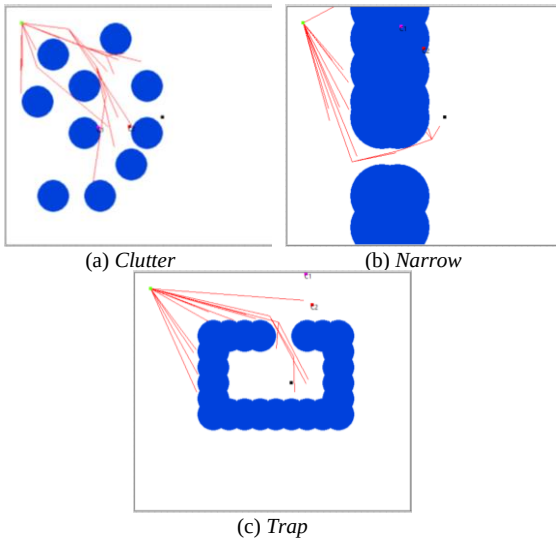
Gambar 8. Pohon pencarian RRT* *Improved gaussian sampling* pada berbagai lingkungan (a) *clutter* (b) *Narrow* (c) *Trap*



(a) *Clutter* (b) *Narrow*

(c) *Trap*

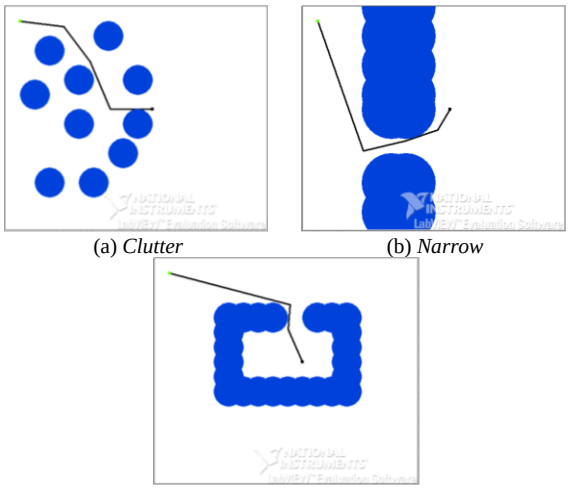
Gambar 9. Visualisasi final path RRT* *improved gaussian sampling* pada berbagai lingkungan (a) *clutter* (b) *Narrow* (c) *Trap*



(a) *Clutter* (b) *Narrow*

(c) *Trap*

Gambar 10. Pohon pencarian algoritma RRT* *gaussian sampling* pada berbagai lingkungan (a) *clutter* (b) *Narrow* (c) *Trap*



(a) *Clutter* (b) *Narrow*

(c) *Trap*

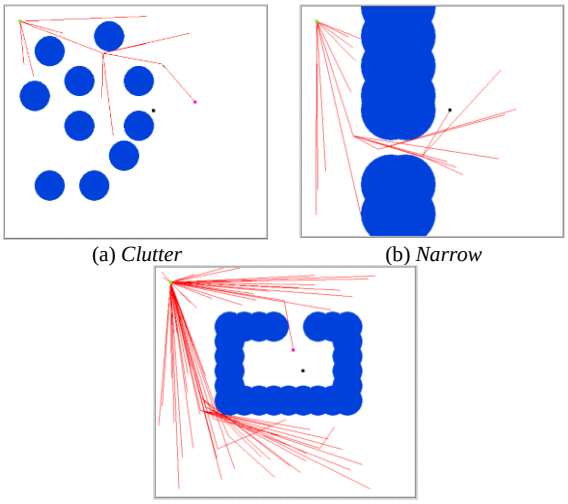
Gambar 11. Visualisasi final path RRT* *gaussian sampling* pada berbagai lingkungan (a) *clutter* (b) *Narrow* (c) *Trap*

Algoritma RRT* memiliki sifat yang *asymtotic* optimal dimana algoritma RRT* akan mengubah jalur yang lebih optimal ketika ditemukannya jalur baru [16]. Jalur yang dipilih *gaussian sampling* pada lingkungan *clutter* selalu berlika-liku sehingga nilai *path cost* yang dihasilkan akan lebih besar dan didapatkan rata-rata nilai *path cost* sepanjang 10 dengan waktu 40,09; *narrow* sepanjang 13,53 dengan waktu 40,03 dan *trap* sepanjang 10,95 dengan waktu 40,12. Sebaliknya, metode *improved gaussian sampling* memiliki banyak titik yang dapat dijadikan pilihan untuk memilih jalur yang lebih optimal. Sehingga jalur yang dihasilkan akan cenderung lebih lurus dan waktu komputasi yang dibutuhkan untuk mencari titik *goal* lebih cepat. Hasil pengujian didapatkan rata-rata nilai *path cost* dan waktu komputasi senilai: *clutter* sepanjang 8,89 dengan waktu 40,05; *narrow* sepanjang 12,58 dengan waktu 40,03 dan *trap* sepanjang 9,93 dengan waktu 40,04. Hasil pengujian *random sampling* didapatkan: *clutter* sepanjang 10,86 dengan waktu 0,03; *narrow* sepanjang 14,82 dengan waktu 0,25 dan *trap* sepanjang 11,71 dengan waktu 0,21. Hasil pohon pencarian RRT* metode *random sampling* pada berbagai lingkungan (a) *clutter* (b) *narrow* (c) *trap* ditunjukkan pada **Gambar 12.** Visualisasi hasil *final path* RRT* metode *random sampling* pada berbagai lingkungan (a) *clutter* (b) *narrow* (c) *trap* ditunjukkan pada **Gambar 13.**

B. Pengukuran Performansi Terhadap Jumlah *Sampling*

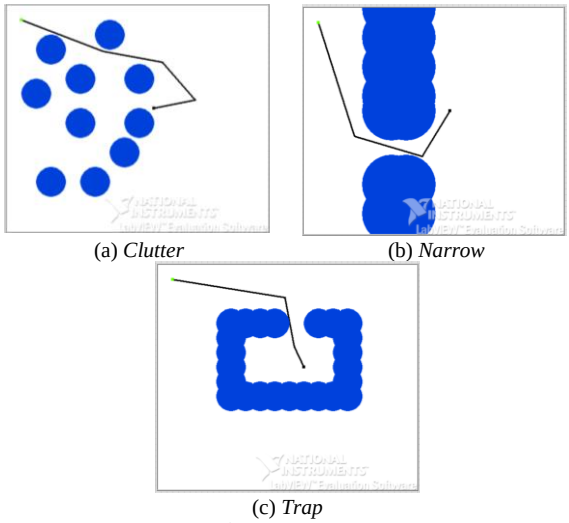
Pengujian pada bagian ini dilakukan dengan pengambilan performansi sistem terhadap jumlah *sampling* pada lingkungan *clutter*. Dalam algoritma *improved gaussian sampling* ini saat

ditemukannya nilai c (*gaussian random number*) maka terjadi pengambilan *sampling* sebanyak 1-10 kali *sampling* yang berada disekitar sisi *obstacle*. Pada publikasi Valérie dkk diujikan *gaussian sampling* pada PRM hanya dengan 1 buah *sampling* pada setiap sisi *obstacle* setiap iterasi [11]. Pengujian dilakukan sebanyak 100 kali percobaan dengan jumlah iterasi sebanyak 200 kali iterasi pada setiap nilai *sampling* yang berbeda lalu didapatkan nilai rata-rata pengujian tersebut yang kemudian akan dibandingkan antara *sampling* satu dengan *sampling* lainnya. Lalu, diambil nilai terbaik, terburuk, rata-rata (*mean*) dan nilai simpangan baku (standar deviasi/STD) dari seluruh pengujian. Hasil pengujian pengukuran performansi sistem terhadap jumlah *sampling* dapat dilihat pada **Tabel III** berikut ini.



(a) *Clutter* (b) *Narrow* (c) *Trap*

Gambar 12. Pohon pencarian algoritma RRT* *random sampling* pada berbagai lingkungan (a) *clutter* (b) *Narrow* (c) *Trap*



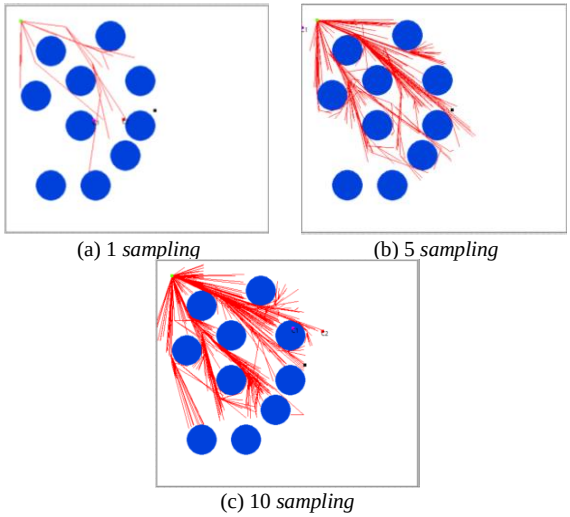
(a) *Clutter* (b) *Narrow* (c) *Trap*

Gambar 13. Visualisasi final path RRT* *random sampling* pada berbagai lingkungan (a) *clutter* (b) *Narrow* (c) *Trap*

Tabel III. Pengukuran Performansi Terhadap Jumlah *Sampling*

Jumlah *sampling*	Terbaik	Terburuk	*Mean*	STD
1	9,07	10,72	10,12	0,60
2	8,79	10,50	9,69	0,66
3	9,22	10,46	9,63	0,5
4	8,46	10,32	9,44	0,75
5	7,77	10,23	9,39	0,81
6	7,84	10	9,2	0,71
7	8,29	10,5	9,11	0,66
8	8,11	10,96	9,07	1,08
9	8,47	10,1	9	0,57
10	7,95	10,12	8,9	0,68

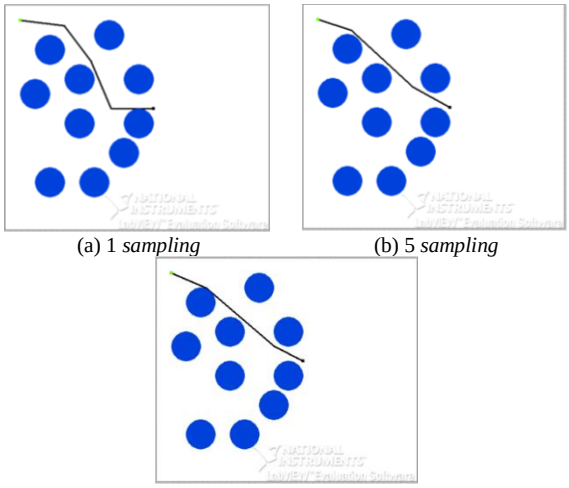
Pada **Tabel III** hasil pengujian performansi terhadap jumlah *sampling* dapat terlihat bahwa nilai *path cost* pada algoritma RRT* metode *improved gaussian sampling* semakin mengecil. Begitu pun dapat terlihat pada nilai waktu komputasi yang dibutuhkan untuk mencapai *goal node*, waktu komputasi semakin cepat. Dapat dianalisis bahwa jumlah *sampling* yang digunakan pada metode *improved gaussian* sangat berpengaruh terhadap nilai *path cost* dan waktu komputasi. Semakin banyak jumlah *sampling* yang digunakan maka nilai *path cost* akan semakin kecil dan waktu yang dibuthkan untuk mencapai titik *goal* semakin cepat. Visualisai hasil pohon pencarian algoritma RRT* metode *improved gaussian sampling* terhadap jumlah *sampling* ditunjukkan pada **Gambar 14** dengan jumlah *sampling* 1 *sampling*, 5 *sampling* dan 10 *sampling*.



(a) 1 *sampling* (b) 5 *sampling* (c) 10 *sampling*

Gambar 14. Hasil pohon pencarian terhadap jumlah *sampling* (a) 1 *sampling* (b) 5 *sampling* (c) 10 *sampling*

Setiap pengujian dilakukan sebanyak 100 kali iterasi dengan jumlah *sampling* yang berbeda-beda. Dapat terlihat pada **Gambar 14 (a)** dengan jumlah *sampling* 1 *sampling* hasil pohon pencarian, cabang yang dijelajahi tidak meluas dikarenakan setiap iterasi hanya diambil 1 jumlah *sampling* diantara titik c_1 dan c_2. Jumlah *sampling* yang digunakan sangat berpengaruh terhadap hasil pohon pencarian dapat terlihat pada **Gambar 14 (a)** dengan jumlah *sampling* 1 *sampling* dan **Gambar 14 (c)** dengan jumlah *sampling* 10 *sampling*. Semakin banyak jumlah *sampling* yang digunakan maka jumlah pohon pencarian dan penjelajahan ruang konfigurasi akan semakin luas. Visualisai *final path* algoritma RRT* metode *improved gaussian sampling* terhadap jumlah *sampling* ditunjukkan pada **Gambar 15** dengan jumlah *sampling* 1 *sampling*, 5 *sampling* dan 10 *sampling*.



(a) 1 *sampling* (b) 5 *sampling*

(c) 10 *sampling*

Gambar 15. Hasil *final path* terhadap jumlah *sampling* (a) 1 *sampling* (b) 5 *sampling* (c) 10 *sampling*

Dapat terlihat pada **Gambar 15 (a)** dengan **Gambar 15 (b)** hasil jalur yang hasilkan pada **Gambar 15 (a)** lebih berlika-liku dan cenderung mengambil jalur yang jauh dengan sisi obstacle. Sedangkan pada **Gambar 15 (b)** jalur yang dilalui tidak berlika liku tetapi langsung menuju titik *goal* dan nilai *path cost* yang dihasilkan lebih rendah dibandingkan dengan **Gambar 15 (a)** yang menggunkan jumlah *sampling* 1 *sampling*. Peneliti membandingkan lagi dengan pengujian pada 10 *sampling*. Hasil pencarian jalur yang didaptkan dapat terlihat pada **Gambar 15 (c)** jalur yang dihasilkan lebih optimal dibandingkan dengan **Gambar 15 (a).** Nilai *path cost* yang dihasilkan lebih kecil dibandingkan dengan hasil pencarian jalur *sampling* sebelumnya. Dapat disimpulkan bahwa jumlah *sampling* sangat berpengaruh terhadap jalur yang dihasilkan, semakin banyak *sampling* yang digunakan maka jalur yang dihasilkan akan semakin optimal.

IV. KESIMPULAN

Berdasarkan Hasil pengujian perbandingan performansi algoritma metode *improved gaussian sampling*, *gaussian sampling* dan *random sampling* pada berbagai lingkungan didapatkan bahwa metode *improved gaussian sampling* lebih optimal dalam proses pencarian jalur tercepat. Disimpulkan bahwa pada algoritma RRT* dengan menggunakan metode *improved gaussian sampling* menghasilkan performansi yang lebih baik dibandingkan algoritma RRT* yang menggunakan metode *sampling* lainnya. Berdasarkan hasil pengujian didapatkan semakin banyak jumlah sampling yang diberikan maka nilai *path cost* yang dihasilkan optimal. Saran pengembangan penelitian ini pada proses pengukuran pengambilan sampling lebih banyak lagi agar didapatkan nilai yang lebih optimal.

DAFTAR PUSTAKA

[1] N. M. Amato dan G. Song, “Using motion planning to study protein folding pathways,” *J. Comput. Biol.*, vol. 9, no. 2, hal. 149–168, 2001.

[2] O. Burçhan Bayazit, G. Song, dan N. M. Amato, “Enhancing *random*ized motion planners: Exploring with haptic hints,” *Auton. Robots*, vol. 10, no. 2, hal. 163–174, 2001.

[3] M. Wang, H. Shan, R. Lu, R. Zhang, X. Shen, dan F. Bai, “Real-Time path planning based on hybrid-VANET-enhanced transportation system,” *IEEE Trans. Veh. Technol.*, vol. 64, no. 5, hal. 1664–1678, 2015.

[4] Parada, V., Ferland, J. A., Arias, M., Schwarzenberg, P., & Vargas, L. S. (2009). Heuristic determination of distribution trees. IEEE Transactions on Power Delivery, vol. 25, no. 2, hal. 861-869, 2009.

[5] Y. Deng, Y. Chen, Y. Zhang, dan S. Mahadevan, “Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment,” *Ahall. Soft Comput. J.*, vol. 12, no. 3, hal. 1231–1237, 2012.

[6] R. Bohlin dan L. E. Kavraki, “Path Planning Using Lazy PRM,” *Interpret.* In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*., vol. 1, hal. 521-528, 2000.

[7] S. M. LaValle, “Rapidly-exploring *random* trees: A new tool for path planning,” Tech. Rep., Computer Science Dept., Iowa State University, hal. 1-4, 1998.

[8] S. Karaman dan E. Frazzoli, “Optimal kinodynamic motion planning using incremental *sampling*-based methods,” In 49th IEEE conference on decision and control (CDC), hal. 7681-7687, 2010.

[9] E. D’Hoe, L. Van Hoorebeke, E. Pauwels, dan M. Dierick, “Optimisation of organic tissue visualisation in high resolution X-ray computed tomography,” *Lib.Ugent.Be* , hal. 113–120, 1996.

[10] J. J. Kuffner dan S. M. La Valle, “RRT-connect: an efficient ahalroach to single-query path planning,” *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2, no. Icra, hal. 995–1001, 2000.

[11] V. Boor, M. H. Overmars, dan A. F. van der Stahalen, "The *Gaussian sampling* strategy for probabilistic roadmap planners," in Robotics and Automation, 1999. Proceedings. 1999 IEEE

International Conference on, vol. 2, hal. 1018-1023, 1999.
[12] L. Blackmore, H. Li, dan B. Williams, "A probabilistic ahalroach to optimal robust path planning with obstacles," *Proc. Am. Control Conf*, vol. 2006, hal. 2831–2837, 2006.
[13] Kavraki, Lydia E., et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, hal. 566-580, 1996.
[14] M. Elbanhawi dan M. Simic, "*Sampling*-based robot motion planning: A review," *IEEE Access*, vol. 2, hal. 56–77, 2014.
[15] M. Aria, "Algoritma Perencanaan Jalur Kendaraan Otonom di Lingkungan Perkotaan dari Sudut Pandang Filosofi Kuhn dan Filosofi Pohaler," *Telekontran J. Ilm. Telekomun. Kendali dan Elektron. Terap.*, vol. 7, no. 2, hal. 145-156, 2019.
[16] Nasir, F. Islam, dan Y. Ayaz, "Adaptive Rapidly-Exploring-*Random*-Tree-Star (RRT*) -Smart: Algorithm Characteristics and Behavior Analysis in Complex Environments," *Asia-Pacific J. Inf. Technol. Multimed.*, vol. 02, no. 02, hal. 39–51, 2013.