TELEKONTRAN, VOL. 9, NO. 2, OKTOBER 2021
DOI : 10.34010/telekontran.v9i2.5675
p-ISSN : 2303 – 2901
e-ISSN : 2654 – 7384

Algoritma *Rapidly Exploring Random Tree** dengan Integrasi Metode *Sampling Goal Biassing, Gaussian*, dan *Boundary*

Rapidly Exploring Random Tree Algorithm with Integration of Goal Biassing, Gaussian, and Boundary Sampling Methods*

Aldi Sopa[*], Tri Rahajoeningroem, Rodi Hartono
Program Studi Teknik Elektro, Fakultas Teknik dan Ilmu Komputer
Universitas Komputer Indonesia Jl. Dipati ukur No 112, Bandung
[*]Email : aldisopa@mahasiswa.unikom.ac.id

Abstrak - Algoritma perencanaan jalur adalah untuk menemukan lintasan yang membawa robot dari keadaan awal (*start*) ke keadaan tujuan (*goal*) sambil menghindari tabrakan dengan rintangan. Dalam algoritma perencanaan jalur, berbagai aplikasi telah digunakan seperti animasi, kedokteran, pesawat, dll. Tujuan penelitian ini adalah merancang metode *sampling* baru dengan cara melakukan integrasi metode *sampling* berbasis goal biassing, *Gaussian* dan *Boundary* lalu mengimplementasikannya pada masalah perencanaan jalur menggunakan algoritma *Rapidly Exploring Random Tree** (RRT*). Metode *sampling* tersebut kami namakan metode *sampling* integrasi. Algoritma perencanaan jalur menggunakan metode *sampling* integrasi ini diimplementasikan pada bahasa pemograman Labview. Pengujian dilakukan pada lingkungan *obstacle clutter, SquareField BW,* dan *trap,* dimana pengujian dilakukan 20 kali percobaan pada masing-masing *obstacle.* Pengujian dilakukan untuk membandingan jarak jalur serta waktu komputasi dari algoritma RRT* yang menggunakan metode *sampling* integrasi, terhadap algoritma RRT* yang menggunakan metode *sampling Gaussian,* dan *Boundary*. Berdasarkan hasil pengujian, diperoleh bahwa algoritma RRT* yang menggunakan metode *sampling* integrasi dapat menghasilkan jalur yang lebih pendek dibandingkan dengan algoritma RRT* yang menggunakan metode *Gaussian* maupun algoritma RRT* yang menggunakan *sampling Boundary.* Perbandingan waktu komputasi yang dihasilkan lebih cepat metode integrasi dibandingkan dengan *Gaussian.* Akan tetapi, pada perbandingan dengan *Boundary* menunjukkan bahwa *Boundary* memerlukan lebih sedikit waktu dibandingkan dengan integrasi. Maka dari itu dapat disimpulkan bahwa algortima *Rapidly Exploring Random Tree** metode integrasi lebih unggul dibandingkan dengan metode *Gaussian* maupun metode *Boundary.* Jalur yang lebih pendek serta waktu komputasi yang lebih cepat menjadikan metode integrasi lebih unggul dibandingkan metode *Gaussian, Boundary,* dan *goal biassing.*

Kata kunci *:* *Rapidly Exploring Random Tree**, integrasi *sampling, Gaussian sampling, Boundary sampling, goal biassing sampling*

Abstract *- The path planning algorithm is to find a path that takes the robot from the start state to the goal state while avoiding collisions with obstacles. In path planning, various applications have been used such as animation, medicine, aircraft, etc. The purpose of this study is to design a new sampling method by integrating sampling methods based on goal biasing, Gaussian and Boundary and then implementing it in path planning problems using the Rapidly Exploring Random Tree* (RRT*) algorithm. We call this sampling method the integration sampling method. The path planning algorithm using this integration sampling method is implemented in the Labview programming language. The algorithm parameters in Labview can be modified to observe the output performance of the RRT* algorithm. The test was carried out in an environment of obstacle clutter, SquareField BW, and traps, where the test was carried out 20 times for each obstacle. The test was conducted to compare the path distance and computation time of the RRT* algorithm using the integration sampling method, against the RRT* algorithm using the Gaussian, and Boundary sampling method. Based on the test results, it is found that the RRT* algorithm using the integration sampling method can produce a shorter path than the RRT* algorithm using the Gaussian method and the RRT* algorithm using Boundary sampling. Comparison of the resulting computational time is faster than the Gaussian integration method. However, a comparison with Boundary shows that Boundary requires less time than integration. Therefore, it can be concluded that the Rapidly Exploring Random Tree* algorithm integration method is superior to the Gaussian method and the Boundary method.*

***Keywords** : Rapidly Exploring Random Tree*, integration sampling, Gaussian sampling, Boundary sampling, Goal Biassing sampling*

I. PENDAHULUAN

A. Latar Belakang

Masalah perencanaan jalur adalah bidang penelitian mendasar dalam dunia robotika [1]. Masalah perencanaan jalur adalah untuk menemukan lintasan yang membawa robot dari keadaan awal (start) ke keadaan tujuan (goal) sambil menghindari tabrakan dengan rintangan [2]. Terdapat berbagai aplikasi dalam perencanaan jalur, B Paden [3] menggunakan perencanaan jalur pada self driving vehicle, perilaku mengemudi yang berupa change lane, cruise in lane, atau turn right harus diterjemahkan ke dalam jalur atau lintasan. Jalur lintasan yang dihasilkan layak secara dinamis, dan menghindari tabrakan dengan rintangan yang terdeteksi oleh sensor on-board.

D. Gonzales [4] dalam penelitiannya pada automated vehicle menggunakan teknik perencanaan yang diklasifikasikan dalam 4 kelompok, menurut penerapannya dalam penggerak otomatis: pencarian grafik, pengambilan sampel, interpolasi, dan optimisasi numerik. Y. Liu [5] melakukan penelitian algoritma perencanaan jangkauan real-time berbasis heuristik untuk sosok manusia virtual. Posisi awal dan tujuan dibuat dalam ruang kerja 3D, Menghitung jalur bebas tabrakan yang menentukan semua konfigurasi lengan manusia untuk bergerak dari awal ke tujuan.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 *Rapidly Exploring Random Tree* diperkenalkan oleh Steven M. LaValle dan James J. Kuffner Jr yang merupakan salah satu algoritma popular dan sering digunakan hingga saat ini [7]. Tahun 2011 RRT dikembangkan menjadi versi lebih optimal oleh Karaman dan Frazolli yang disebut dengan *Rapidly Exploring Random Tree Star (*RRT*) [8]. RRT* merupakan algoritma RRT dalam versi lebih optimal.

B. Tinjauan *State of Art*

Berbagai metodologi telah dilakukan untuk menghasilkan jalur yang optimal [6]. Perencana jalur berbasis pohon dikembangkan oleh LaValle dan Kuffner [7] yang berguna untuk menjelajahi ruang kerja atau C-space. Teknik perencanaan jalur ini dikenal dengan *Rapidly Exploring Random Tree* (RRT). RRT diperkenalkan pada penelitian [8] sebagai algoritma perencanaan untuk dengan cepat mencari ruang berdimensi tinggi yang memiliki kendala aljabar (yang muncul dari hambatan) dan kendala diferensial (yang muncul dari nonholonomi dan dinamis). Pada penelitian [9] diterangkan bahwa RRT memiliki keunggulan dalam memberikan jalur pertama tercepat dan juga kelengkapan probabilistik tetapi tidak menjamin optimalitas asimtotik. Pada tahun 2010, Karaman dan Frazolli pertama kali mengusulkan algoritma *Rapidly Exploring Random Tree** (RRT*). Algoritma ini bertujuan untuk mengatasi kekurangan dari algoritma RRT yang konvergen ke nilai yang tidak optimal [10].

Salah satu bagian dari algoritma perencanaan jalur adalah metode *sampling*, terdapat beberapa metode *sampling* yang telah diusulkan untuk mengurangi waktu konvergensi yang telah diterapkan pada navigasi perencanaan jalur [11]. Penelitian Jing Dong menunjukkan bagaimana interpolasi *Gaussian Process* dapat lebih meningkatkan kecepatan algoritma, dan menggunakan grafik faktor serta optimasi numerik untuk melakukan interfensi dengan cepat [12]. Penelitian J. Wang [13] menggunakan metode *Boundary Sampling* yang mengklasifikasikan hambatan pada ruang kerja menjadi hambatan yang berharga dan yang tidak berharga, kemudian menggunakan informasi batasnya untuk membuat titik sampel perencana jalur di sekitarnya, selanjutnya pohon acak akan tumbuh ke posisi tujuan. Terdapat beberapa *library path planning* pada bahasa pemograman, salah satu contohnya penelitian [14] melakukan implementasi algoritma pada Borland C++, kemudian diuji menggunakan bahasa pemograman *visual basic* dan *delphi.* Sejauh yang penulis ketahui, belum ada *library path planning* pada bahasa pemograman Labview.

C. Tujuan

Pada penelitian ini akan dirancang metode *sampling* dengan menggabungkan *sampling Goal Biassing, Gaussian,* dan *Boundary*. Kami menamakan metode *sampling* tersebut sebagai metode Integrasi. Pada penelitian ini metode integrasi akan diimplementasikan pada Labview.

Berdasarkan permasalahan diatas, maka penulis mengangkat sebuah topik pembahasan yang akan digunakan pada pelaksanaan penelitian ini. Penulis mengangkat sebuah judul "Integrasi Metode *Sampling Goal Biassing, Gaussian,* dan *Boundary* Pada Algoritma *Rapidly Exploring Random Tree** ".

D. Sistematika Pembahasan

Sistematika pembahasan yang akan diuraikan dalam laporan ini terbagi dalam bab – bab yang akan dibahas. Bagian pertama akan menjelaskan mengenai latar belakang, tinjauan *state of art*, serta tujuan dari pembuatan jurnal. Bagian kedua menjelaskan mengenai perancangan algoritma RRT* pada Labview dan pemilihan metode *sampling* yang digunakan. Bagian ketiga menyajikan hasil pengujian dan analisa. Kemudian pada bagian keempat menyajikan kesimpulan dari penelitian yang telah dilakukan.

II. METODOLOGI

A. Implementasi algoritma RRT* pada Labview

Algoritma RRT* adalah pengembangan dari algortima RRT. Perbedaan dengan algoritma RRT adalah algoritma RRT* terdapat subprogram *choose parent* dan *rewire.* **Gambar 1** Di bawah ini adalah *pseudo* dari algoritma RRT*.

Algorithm 1: $T = (V, E) \leftarrow$ **RRT***(z_{init})

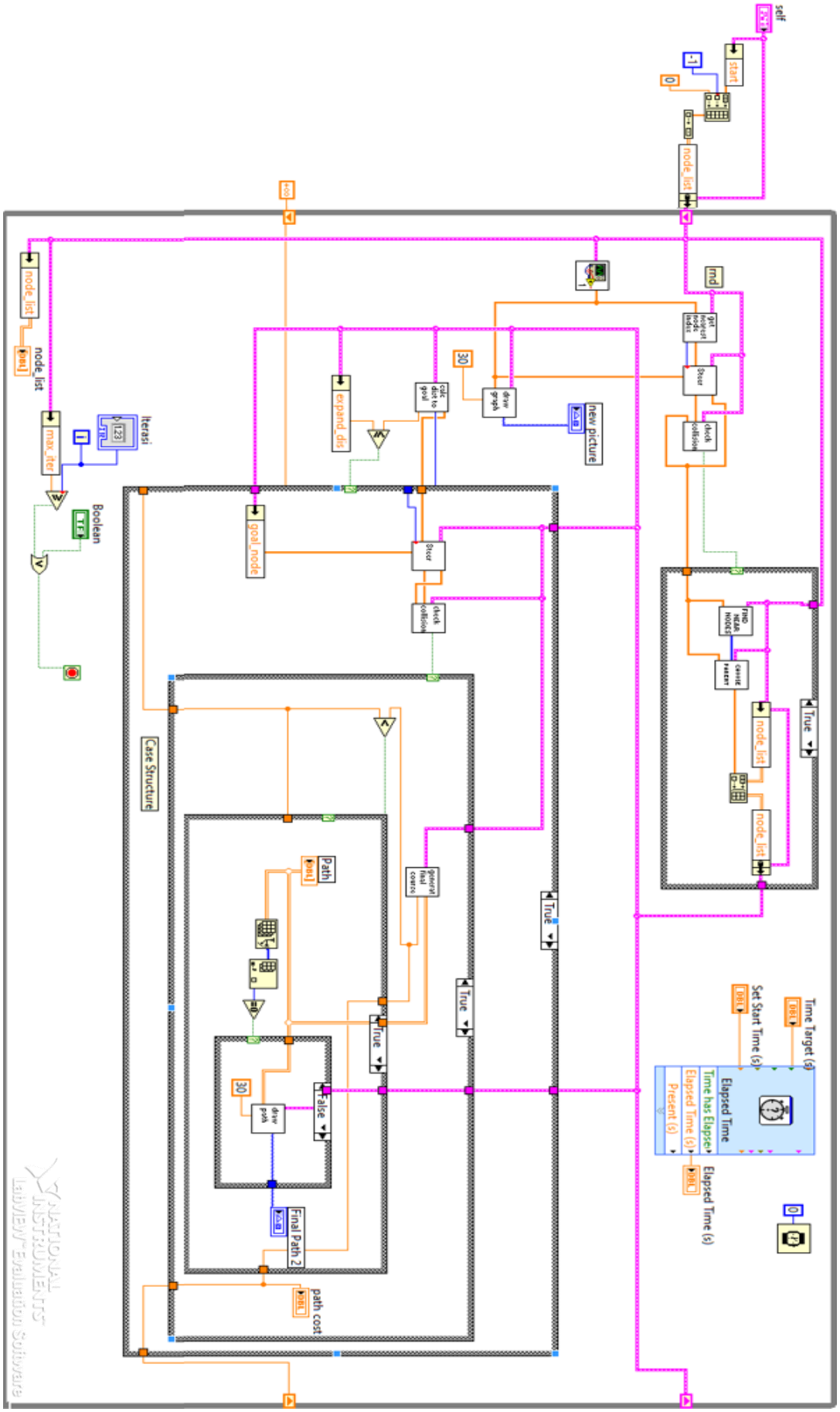
```
T← InitializeTree();
T← InsertNode(Ø, z_init, T);
for i=0 to i=N do
    z_rand ← Sample(i);
    z_nearest ← Nearest(T, z_rand);
    (x_new, u_new, T_new) ← Steer (z_nearest, z_rand);
    if Obstaclefree(x_new) then
        Z_near ← Near(T, z_new, |V|);
        z_min ← Chooseparent (Z_near, z_nearest, z_new, x_new);
        T← InsertNode(z_min, z_new, T);
        T← Rewire (T, Z_near, z_min, z_new);
return T
```

Gambar 119. *Pseudo code* Algoritma RRT*

Pada program utama RRT* terdapat suatu masukan *cluster self* yang di dalamnya terdapat beberapa parameter yang telah ditentukan. variabel masukan di dalam *cluster self* diantaranya adalah koordinat *start*(x,y), yaitu koordinat dari titik awal perencanaan jalur. *Goal*(x,y) yaitu tujuan yang hendak dicapai dari perencanaan jalur. *Min random* dan *maks random* yaitu *range* area pencarian minimum dan pencarian maksimum. *Expand distance,* yaitu panjang langkah dari perencanaan jalur. *Path resolution,* yaitu untuk memeriksa *obstacle* sekecil apa yang dapat dideteksi oleh program. *Max iter,* yaitu banyaknya iterasi maksimal. *Obstacle list,* yaitu parameter untuk mengatur letak, ukuran *obstacle* pada perencanaan jalur. *Goal sample rate,* yaitu secara berkala dengan kemungkinan 5% mengambil sampel baru dengan *node random* berada pada *node* tujuan. Kemudian pada program utama terdapat beberapa *sub program* sebagai langkah proses yang dilakukan pada algoritma RRT* dalam memproses variabel masukan, diantaranya adalah:

1. Subprogram "*Get random node.*vi*"*
2. Sub program "*Get Nearest Node Index.vi"*
3. Sub program "*Steer*.vi"
4. Sub program "*Choose parent*.vi"
5. Sub program "*rewire.*vi*"*
6. Sub program "*Draw graph.*vi"
7. Sub program "*Generate final course.*vi"
8. Sub program "*Draw path.*vi"

Proses *get random node* berfungsi mengambil sampel secara acak pada ruang pencarian, yang disebut sebagai *random node*. *Get nearest node index* berfungsi untuk mencari jarak *node* yang ada di pohon pencarian ke *node* yang baru untuk mencari *node* yang dekat dengan random *node* sebagai posisi cabang yang baru. *Steer* berfungsi menambah dari cabang terdekat menjadi cabang baru dimana cabang baru mendekati *random node*. *Check collison* berfungsi apabila ada cabang baru yang terlalu dekat dengan *obstacle*. Kemudian proses dua optimasi dari RRT* yaitu *choose parent* dan *rewire*, yang memeriksa cabang terdekat dari *random node* dan meminimalkan *cost* dari titik awal. Selanjutnya proses *draw graph*, *generate final course,* dan *draw path* adalah *sub program* yang berfungsi untuk memvisualisasikan hasil dari proses sebelumnya ke dalam *output* berupa gambar. Berdasarkan *pseudo code* pada **Gambar 1,** maka dibuatlah program RRT* pada bahasa pemograman Labview. Program RRT* pada Labview dapat dilihat pada **Gambar 2** di bawah.



Gambar 2. Program Utama RRT* pada Labview

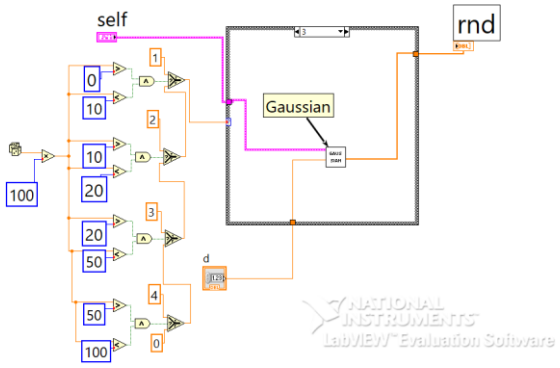
B. Metode Sampling Integrasi

Metode *sampling* integrasi merupakan gabungan dari metode *sampling Goal Biassing, Gaussian,* dan *Boundary*. *Pseudo code* dari metode sampling integrasi dapat dilihat pada **Gambar 3** di bawah

```
Algorithm : Integrasi Sampling
1 :   rand ←RandomNumber                    between 0 – 100
2 :   if rand < 10 then
3 :     qrand ← GoalBiassing Sampling
4 :       Else If 10<rand<20 then
5 :         qrand ← RandomSampling
6 :           Else if 20<rand<50 then
7 :             qrand ← GaussianSampling
8 :               Else if 50<rand<80 then
9 :                 qrand ← BoundarySampling
10 :              End
11 :          End
12 :      End
13 :    End
14 :    return qrand
```

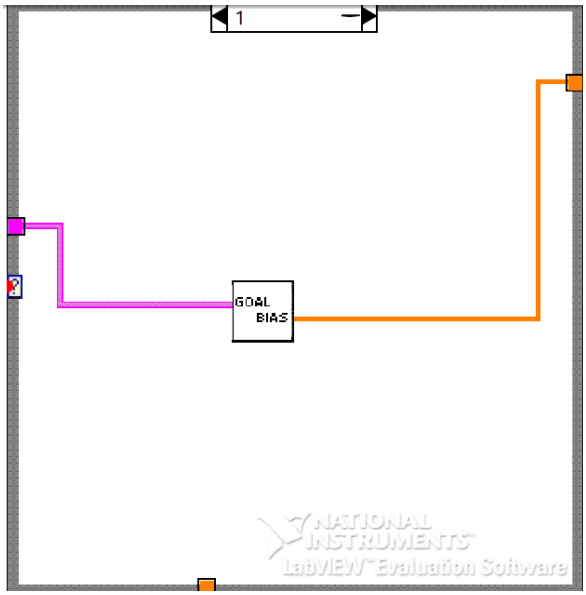
Gambar 3. *Pseudo code* integrasi *sampling*

Algoritma metode integrasi *sampling* pada **Gambar 3** bertujuan untuk memilih metode *sampling* mana yang akan digunakan berdasarkan dari nilai acak. Blok diagram labview dari integrasi *sampling* dapat dilihat pada **Gambar 4.**

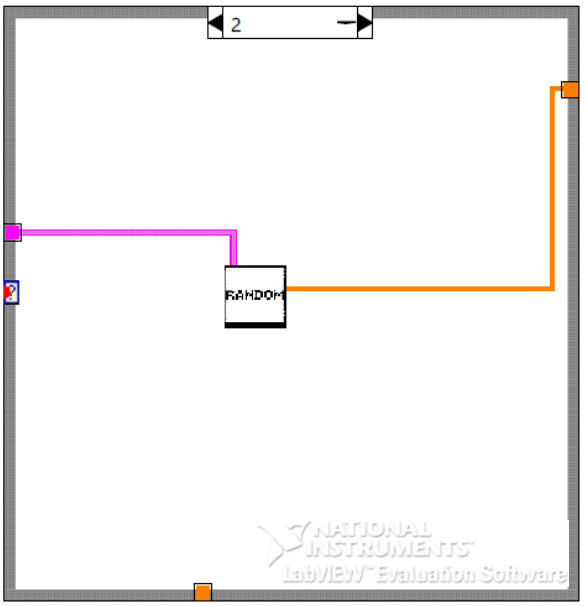


Gambar 4. Blok diagram integrasi *sampling*

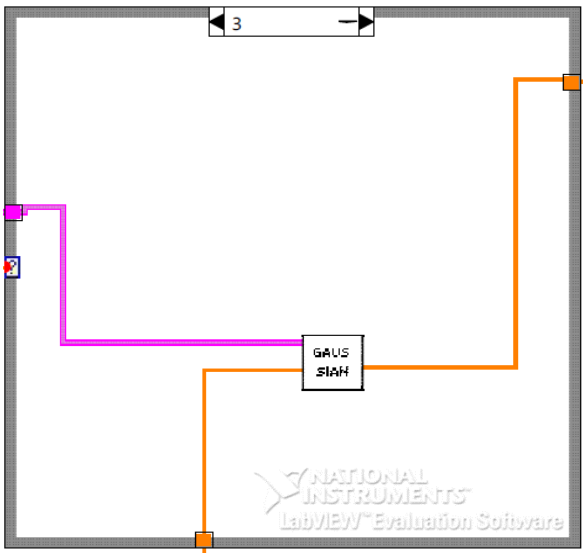
Pada **Gambar 4,** terdapat 3 metode *sampling* yang dimasukkan dalam *case structure* labview. *Case structure* pertama berisikan metode *sampling Goal Biassing* yang dapat dilihat pada **Gambar 5**. *Case structure* kedua pada metode *sampling* integrasi terdapat metode *sampling random* yang dapat dilihat pada **Gambar 6**. *Case structure* ketiga terdapat metode *sampling Gaussian* yang dapat dilihat pada **Gambar 7**. *Case structure* keempat pada metode *sampling* integrasi terdapat metode *sampling Boundary* yang dapat dilihat pada **Gambar 8.**



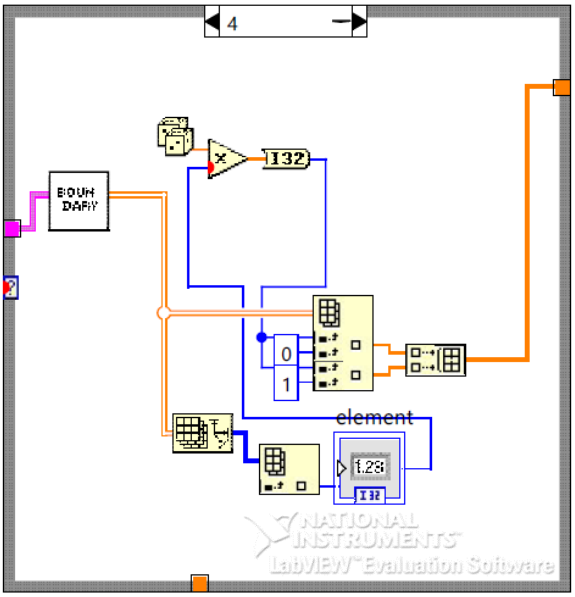
Gambar 5. *Sampling Goal Biassing*



Gambar 6. *Random sampling*



Gambar 7. *Gaussian sampling*



Gambar 8. *Boundary sampling*

III. HASIL DAN PEMBAHASAN

Skenario eksperimen pada lingkungan obstacle yang berbeda, serta hasil eksperimen dan diskusi disajikan pada bagian ini. Semua eksperimen dan analisis dilakukan menggunakan PC yang sama (AMD A9 3,00 GHz dan RAM 4GB) dengan windows 10 pro. Bahasa pemograman Labview (2014) digunakan untuk mengimplementasikan algoritma yang diusulkan. Semua eksperimen diuji dengan 20 kali percobaan dengan parameter yang sama, jumlah maksimum iterasi digunakan sebagai kondisi *stop* untuk semua algoritma. Metrik kinerja yang digunakan untuk membandingkan kinerja algoritma yang diusulkan dengan algoritma pembanding mencakup jarak jalur (path cost), waktu komputasi (time), panjang jalur rata-rata (dilambangkan sebagai mean).

A. Perbandingan Performa antara *RRT** metode Integrasi *sampling* dengan Metode *Gaussian sampling*

Pengujian dilakukan sebanyak 20 kali dengan parameter yang telah ditentukan, pengujian bertujuan untuk membandingkan jarak jalur (*path cost*), serta waktu yang ditempuh (*time*). Pengujian pertama dilakukan menggunakan lingkungan *obstacle clutter*, pengujian pertama adalah membandingkan performa antara metode integrasi *sampling* dengan metode *Gaussian sampling*. Data hasil pengujian dapat dilihat pada **Tabel I.**

Tabel I Perbandingan jarak jalur dan waktu pada *obstacle clutter*

*Rapidly Exploring Random Tree**					
Integrasi *sampling*			*Gaussian sampling*		
no.	*path cost (m)*	*Time (s)*	no.	*path cost (m)*	*Time (s)*
1	14,37	56,62	1	14,76	102,31
2	14,02	52,87	2	14,02	104,7
3	13,76	64,43	3	13,95	110,56
4	13,9	50,72	4	14,83	121,26
5	13,95	58,38	5	13,95	112,51
6	14,11	55,71	6	14,01	100,82
7	14,16	57,86	7	13,51	106,73
8	14,04	60,2	8	14,19	124,1
9	13,84	63,75	9	14,26	115,85
10	14,16	53,46	10	13,72	110,29
11	14,47	60,23	11	13,5	123,44
12	13,65	60,5	12	13,91	115,72
13	14,1	54,26	13	14,07	113,78
14	13,87	52,49	14	13,69	102,26
15	13,22	56,17	15	13,88	126,27
16	13,62	65,64	16	13,99	122,74
17	13,7	57,92	17	14,18	119,06
18	14,07	63,71	18	14,01	110,32
19	14,18	59,43	19	13,82	120,7
20	13,76	58,27	20	14,17	118,45
mean	13,94	58,13		14,02	114,09

Pada **Tabel I** terlihat bahwa jarak jalur metode integrasi *sampling* lebih pendek dibandingkan dengan jarak jalur *Gaussian sampling*. Metode integrasi *sampling* memiliki jarak rata-rata 13,94 m dengan waktu komputasi 58,131 detik, sedangkan metode *Gaussian sampling* memiliki jarak rata-rata 14,02 m dengan waktu komputasi 114,09 detik. Maka dari itu, dapat dibuktikan bahwa metode integrasi lebih baik dibandingkan dengan metode *Gaussian sampling* meskipun tidak ada perbedaan yang signifikan pada jarak jalur.

Kemudian pengujian selanjutnya antara metode integrasi *sampling* dan metode *Gaussian sampling* dilakukan pada lingkungan *obstacle SquareField BW*, data hasil pengujian dapat dilihat pada **Tabel II.**

Tabel II Perbandingan jarak jalur dan waktu pada *obstacle SquareField*

*Rapidly Exploring Random Tree**					
Integrasi *sampling*			*Gaussian sampling*		
no.	*path cost (m)*	*time*	no.	*path cost (m)*	*time*
1	12,67	40,82	1	12,51	51,21
2	12,83	43,73	2	12,98	56,92
3	12,95	41,92	3	12,2	53,5
4	13	47,55	4	13,01	59,26
5	13,02	41,21	5	13,85	54,87
6	12,75	43,98	6	13,32	58,49
7	11,52	48,01	7	12,9	51,16
8	11,75	50,82	8	12,65	57,82
9	12,82	45,64	9	12,22	55,81
10	12,16	55,92	10	12,58	53,81
11	13,25	50,2	11	13,19	58,49
12	13,16	50,87	12	13,98	60,05
13	12,7	42,41	13	13,25	58,77
14	12,74	49,89	14	12,32	56,12
15	13,37	48,2	15	12,86	54,85
16	12,8	50,62	16	12,77	54,71
17	12,12	56,12	17	12,61	51,95
18	12,81	48,09	18	13,21	53,8
19	12,21	47,43	19	12,9	58,93
20	13,18	42,82	20	12,76	57,86
min	11,52	40,82		12,2	51,16
mean	12,69	47,31		12,90	55,91

Pada **Tabel II** dapat dilihat bahwa dalam lingkungan *obstacle SquareField BW*, metode Integrasi *sampling* memiliki jarak rata-rata 12,69m dengan waktu komputasi 47,31 detik. Sedangkan, metode *Gaussian sampling* memiliki jarak rata-rata 12,90m dengan waktu komputasi 55,91 detik. Pada pengujian ini juga dapat dibuktikan bahwa metode integrasi *sampling* lebih baik dibandingkan dengan metode *Gaussian sampling.*

Selanjutnya, pengujian antara metode integrasi *sampling* dengan metode *Gaussian sampling* dilakukan pada lingkungan *obstacle trap*. Data hasil pengujian dapat dilihat pada **Tabel III.**

Tabel III Perbandingan jarak jalur dan waktu pada *obstacle trap*

*Rapidly Exploring Random Tree**					
Integrasi *sampling*			*Gaussian sampling*		
no.	*path cost (m)*	*time*	no.	*path cost (m)*	*time*
1	20,26	38,92	1	19,51	51,21
2	20,53	29,03	2	21,14	53,88
3	21,1	29,67	3	20,19	53,72
4	20,92	28,64	4	19,48	50,09
5	19,25	29,87	5	20,09	60,81
6	19,61	28,45	6	21,1	55,94
7	20,87	35,64	7	20,36	56,92
8	20,84	35,28	8	21,35	50,46
9	19,36	38,12	9	21,08	55,67
10	19,47	25,78	10	20,9	51,82
11	19,69	20,5	11	20,87	59,36
12	20,67	29,65	12	20,42	55,67
13	20,98	27,87	13	21,53	58,63
14	20,13	22,63	14	20,87	55,56
15	20,44	29,53	15	20,64	56,45
16	20,23	25,87	16	20,56	50,41
17	19,93	29,63	17	20,96	52,89
18	19,3	32,8	18	21,75	56,7
19	20,64	30,71	19	20,65	50,54
20	20,53	27,65	20	20,16	50,49
min	19,25	20,5		19,48	50,09
mean	20,23	29,81		20,68	54,3

Tabel III di atas menunjukkan bahwa *Rapidly Exploring Random Tree** metode integrasi *sampling* memiliki jarak jalur 20,23m dengan waktu komputasi 29,81 detik. sedangkan metode *Gaussian sampling* memiliki jarak jalur 20,68m dengan waktu komputasi 54,36 detik.

B. Perbandingan Performa antara *RRT** metode Integrasi *sampling* dengan Metode *Boundary sampling*

Pengujian yang dilakukan sama seperti sebelumnya, melakukan 20 kali percobaan dengan parameter yang telah ditentukan. Bertujuan untuk mencari jarak jalur dan waktu komputasi. Pengujian pertama dilakukan pada lingkungan obstacle clutter, data hasil pengujian dapat dilihat pada **Tabel IV**.

Tabel IV perbandingan jarak jalur dan waktu pada obstacle clutter antara metode integrasi sampling dengan *Boundary* sampling

*Rapidly Exploring Random Tree**					
Integrasi *sampling*			*Boundary sampling*		
no.	*path cost (m)*	*Time (s)*	no.	*path cost (m)*	*Time (s)*
1	14,37	56,62	1	15,5	31,72
2	14,02	52,87	2	14,75	43,9
3	13,76	64,43	3	14,82	40,68
4	13,9	50,72	4	15,49	30,09
5	13,95	58,38	5	14,87	47,38
6	14,11	55,71	6	14,55	43,54
7	14,16	57,86	7	15,95	44,36
8	14,04	60,2	8	15,62	49,5
9	13,84	63,75	9	14,6	39,15
10	14,16	53,46	10	14,81	35,28
11	14,47	60,23	11	14,09	30,18
12	13,65	60,5	12	15,68	39,53
13	14,1	54,26	13	14,92	42,81
14	13,87	52,49	14	15,71	38,6
15	13,22	56,17	15	14,28	39,4
16	13,62	65,64	16	14,92	31,88
17	13,7	57,92	17	15,08	39,04
18	14,07	63,71	18	14,72	32,78
19	14,18	59,43	19	14,15	35,67
20	13,76	58,27	20	15,53	30,82
min	13,22	50,72		14,09	30,09
mean	13,94	58,13		15,00	38,31

Pada **Tabel IV** dapat dilihat perbandingan antara metode integrasi *sampling* dengan metode *Boundary sampling.* Metode integrasi *sampling* mempunyai jarak rata-rata 13,94m dengan waktu komputasi 58,13 detik. Sedangkan, metode *Boundary sampling* mempunyai jarak rata-rata 15,02m dengan waktu komputasi 38,31 detik. Pada pengujian ini metode integrasi *sampling* lebih baik dibandingkan dengan metode *Boundary sampling.*

Pengujian selanjutnya antara metode integrasi *sampling* dengan metode *Boundary sampling* adalah pada lingkungan *obstacle SquareField BW*, data hasil pengujian dapat dilihat pada **Tabel V.**

Tabel V Perbandingan jarak jalur dan waktu pada *obstacle SquareField BW* antara metode integrasi *sampling* dan metode *Boundary sampling*

*Rapidly Exploring Random Tree**					
Integrasi *sampling*			*Boundary sampling*		
no.	*path cost (m)*	*Time (s)*	no.	*path cost (m)*	*Time (s)*
1	12,67	40,82	1	13,69	22,46
2	12,83	43,73	2	13,66	28,18
3	12,95	41,92	3	13,94	27,98
4	13	47,55	4	14,36	19,76
5	13,02	41,21	5	14,01	27,54
6	12,75	43,98	6	14,28	28,19
7	11,52	48,01	7	13,67	23,05
8	11,75	50,82	8	13,87	29,81
9	12,82	45,64	9	15,22	21,71
10	12,16	55,92	10	14,25	24,65
11	13,25	50,2	11	15,27	22,92
12	13,16	50,87	12	14,92	28,74
13	12,7	42,41	13	14,54	25,04
14	12,74	49,89	14	14,69	26,71
15	13,37	48,2	15	14,15	25,28
16	12,8	50,62	16	14,98	22,45
17	12,12	56,12	17	13,95	27,11
18	12,81	48,09	18	14,54	24,56
19	12,21	47,43	19	14,28	22,17
20	13,18	42,82	20	14,76	26,54
min	11,52	40,82		13,66	19,76
mean	12,69	47,31		14,35	25,24

Seperti yang terlihat pada **Tabel V** di atas, jarak jalur rata-rata yang dihasilkan oleh metode integrasi *sampling* lebih pendek yaitu 12,69m dengan waktu komputasi 47,31 detik. Dibandingkan dengan yang dihasilkan oleh *Boundary sampling* yaitu 14,35m dengan waktu komputasi 25,24 detik.

Selanjutnya pengujian antara metode integrasi *sampling* dengan metode *Boundary sampling* dilakukan pada lingkungan *obstacle trap*. Jarak jalur (*path cost)* dan waktu komputasi (*cost*) dilakukan dalam pengujian ini data hasil pengujian dapat dilihat pada **Tabel VI.**

Tabel VI. Perbandingan jarak jalur dan waktu pada *obstacle trap* antara metode integrasi *sampling* dan metode *Boundary sampling.*

*Rapidly Exploring Random Tree**					
Integrasi *sampling*			*Gaussian sampling*		
no.	*path cost (m)*	*Time (s)*	no.	*path cost (m)*	*Time (s)*
1	20,26	38,92	1	21,21	19,87
2	20,53	29,03	2	21,14	23,01
3	21,1	29,67	3	20,19	27,92
4	20,92	28,64	4	19,48	20,81
5	19,25	29,87	5	20,09	25,29
6	19,61	28,45	6	21,1	24,76
7	20,87	35,64	7	20,36	21,83
8	20,84	35,28	8	21,35	28,63
9	19,36	38,12	9	21,08	25,64
10	19,47	25,78	10	20,9	28,2
11	19,69	20,5	11	20,87	31,15
12	20,67	29,65	12	20,42	27,65
13	20,98	27,87	13	21,53	23,68
14	20,13	22,63	14	20,87	24,65
15	20,44	29,53	15	20,64	21,4
16	20,23	25,87	16	20,56	28,76
17	19,93	29,63	17	20,96	20,19
18	19,3	32,8	18	21,75	22,35
19	20,64	30,71	19	20,65	29,41
20	20,53	27,65	20	20,16	30,03
min	19,25	20,5		19,48	19,87
mean	20,23	29,81		20,76	25,26

Jika dibandingkan dengan penelitian yang dilakukan oleh [15] maka didapatkan bahwa RRT* metode *random sampling* memiliki waktu komputasi yang lebih cepat dibandingkan dengan metode *sampling* lainnya. Hal ini disebabkan oleh pencarian jalur yang tidak memperhatikan optimalisasi pada jarak jalur, akibatnya jarak jalur yang dihasilkan menjadi lebih jauh.

IV. KESIMPULAN

Berdasarkan pengamatan dan pengujian yang telah dilakukan, maka diperoleh kesimpulan akhir. Metode integrasi sampling yang merupakan gabungan dari metode *Goal Biassing* sampling, *Gaussian* sampling, dan *Boundary* sampling telah dapat ditambahkan pada algoritma RRT*. Implementasi algoritma RRT* pada bahasa pemograman Labview juga telah berhasil dilakukan. Hasil perbandingan dari metode integrasi sampling dan *Gaussian* sampling menunjukkan keunggulan jarak jalur dari metode integrasi sampling. Jarak minimal 13,95 (integrasi) dan 14,14 (*Gaussian*), waktu komputasi 115,73 detik (integrasi) dan 189,15 (*Gaussian*) pada obstacle clutter. Jarak minimal 12,8 (integrasi) dan 13,09 (*Gaussian*), waktu komputasi 80,15 detik (integrasi) dan 103,12 detik (*Gaussian*) pada obstacle squareField BW. Jarak minimal 20,38 (integrasi) dan 21,14 (*Gaussian*), waktu komputasi 51,37 detik (integrasi) dan 75,06 (*Gaussian*) pada obstacle trap. Hasil perbandingan dengan metode *Boundary* sampling menunjukkan keunggulan jarak dari integrasi sampling. Jarak minimal 13,95 (integrasi) dan 14,77 (*Boundary*), waktu komputasi 115,73 detik (integrasi) dan 35,87 (*Boundary*) pada obstacle clutter. Jarak minimal 12,8 (integrasi) dan 15,55 (*Boundary*), waktu komputasi 80,15 detik (integrasi) dan 13,98 detik (*Boundary*) pada obstacle squareField BW. Jarak minimal dan waktu komputasi *Boundary* sampling pada obstacle trap tidak dapat ditemukan jalur. Pengembangan lebih lanjut dari penelitian ini dapat dilakukan pada *Informed* RRT* yang menggunakan metode Integrasi sampling atau pun algoritma RRT lainnya yang lebih baik dari RRT*.

DAFTAR PUSTAKA

[1] M. Elbanhawi dan M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, no. February, hal. 56–77, 2014.

[2] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, dan S. Teller, "Anytime motion planning using the RRT," *Proc. - IEEE Int. Conf. Robot. Autom.*, hal. 1478–1483, 2011.

[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, dan E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, hal. 33–55, 2016.

[4] D. González, J. Pérez, V. Milanés, dan F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, hal. 1135–1145, 2016.

[5] Liu, Ying, dan Norman I. Badler. "Real-time reach planning for animated characters using hardware acceleration." In *Proceedings 11th IEEE international workshop on program comprehension*, hal. 86-93. IEEE, 2003.

[6] S. Zhang, J. Pu, dan Y. Si, "An Adaptive Improved Ant Colony System Based on Population Information Entropy for Path Planning of Mobile Robot," *IEEE Access*, vol. 9, hal. 24933–24945, 2021.

[7] LaValle, S.M. Rapidly-exploring random trees: A new tool for path planning. Comput. Sci. Dept. Oct. 1998.

[8] Lavalle, S.M. dan Kuffner, J.J., Jr. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics: New Directions*, hal.293-308, 2000.

[9] J. 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.

[10] S. Karaman, dan E. Frazzoli. "Incremental sampling-based algorithms for optimal motion planning." *Robotics Science and Systems VI* 104, no. 2 (2010).

[11] L. G. D. O. Véras, F. L. L. Medeiros, dan L. N. F. Guimarães, "Rapidly exploring Random Tree* with a sampling method based on Sukharev grids and convex vertices of safety hulls of obstacles," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 1, hal. 1–19, 2019.

[12] J. Dong, M. Mukadam, F. Dellaert, dan B. Boots, "Motion planning as probabilistic inference using *Gaussian* processes and factor graphs," *Robot. Sci. Syst.*, vol. 12, 2016.

[13] J. Wang, X. Li, dan M. Q. H. Meng, "An improved RRT algorithm incorporating obstacle *Boundary* information," 2016 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2016, hal. 625–630, 2016.

[14] O. Hachour, "Path planning of Autonomous Mobile robot," vol. 2, no. 4, hal. 178–190, 2008.

[15] Noreen, Iram, Amna Khan, dan Zulfiqar Habib. "A comparison of RRT, RRT* and RRT*-smart path planning algorithms." *International Journal of Computer Science and Network Security (IJCSNS)* 16. Hal.10-20 (2016).