TELEKONTRAN, VOL. 9, NO. 2, OKTOBER 2021
DOI : 10.34010/telekontran.v9i2.5625
p-ISSN : 2303 – 2901
e-ISSN : 2654 – 7384

Algoritma *Informed*-RRT* Menggunakan *Hybrid Sampling* Untuk Menemukan Solusi akhir jalur yang Cepat

Informed-RRT Using Hybrid Sampling to Finding Fast Final Path Solution*

Miftah Fauzi*, Muhammad Aria Rajasa Pohan
Program Studi Teknik Elektro, Fakultas Teknik dan Ilmu Komputer
Universitas Komputer Indonesia Jl. Dipati ukur No 112, Bandung
*Email : miftahfauzi39@mahasiswa.unikom.ac.id

Abstrak – *Informed Rapidly-exploring Random Tree** (*Informed*-RRT*) merupakan hasil pengembangan dari algoritma *Rapidly-exploring Random Tree* (RRT) yang dapat menghasilkan solusi jalur yang bersifat asimptotik optimal tetapi waktu komputasi yang dibutuhkan menjadi lebih lama. Pada awalnya algoritma *Informed*-RRT* masih menggunakan metode *random sampling* yang mana metode ini akan mengambil sampel acak pada ruang pencarian. Pengambilan sampel acak inilah yang akan membuat waktu komputasi menjadi tidak optimal. Penelitian ini bertujuan untuk merancang algoritma *Informed*-RRT* menggunakan metode *hybrid sampling*. Metode *hybrid sampling* merupakan integrasi dari beberapa metode pengambilan sampel. Pada pengujian ini, performansi metode *random sampling* akan dibandingkan dengan perfomansi metode *hybrid sampling* dalam hal waktu komputasi. Pengujian metode *hybrid sampling* pada algoritma *Informed*-RRT* ini berbasis simulasi dan dilakukan pada lingkungan *narrow, clutter, trap*. Hasil yang didapatkan dari pengujian ini adalah penggunaan metode *hybrid sampling* pada algoritma *Informed*-RRT* mampu menghasilkan performansi waktu rata- rata komputasi yang lebih cepat 26,4 detik bila dibandingkan dengan metode *random sampling* pada lingkungan *clutter*. Pada lingkungan *narrow* metode *hybrid sampling* menghasilkan waktu komputasi 24,52 detik lebih cepat bila dibandingkan dengan metode *random sampling.* Pada lingkungan *trap* metode *hybrid sampling* menghasilkan waktu komputasi 5,25 detik lebih cepat dibandingkan dengan metode *random sampling.* Dari data hasil pengujian, metode *hybrid sampling* ini dapat menjadi metode pangambilan sampel alternatif untuk digunakan pada algoritma *Informed-RRT**

Kata kunci : *Boundary sampling, goal biaing sampling, hybrid sampling, Informed-RRT*, random sampling*

Abstract *- Informed Rapidly-exploring Random Tree* (Informed-RRT*) is the result of the development of the RRT algorithm which can produce an optimal asymptotic path solution but the computation time required is longer. Initially, the Informed-RRT* algorithm was still using the random sampling method, where this method will take a random sample in the search space. This random sampling will make the computation time not optimal. This study aims to design the Informed-RRT* algorithm using a hybrid sampling method. The hybrid sampling method is an integration of several sampling methods. In this test, the performance of the random sampling method will be compared with the performance of the hybrid sampling method in terms of computation time. The test of the hybrid sampling method on the Informed-RRT* algorithm is based on simulation and is carried out in a narrow, clutter, trap environment. The results obtained from this test are that the use of the hybrid sampling method on the Informed-RRT* algorithm is able to produce a faster average computation time performance of 26.4 seconds when compared to the random sampling method in a cluttered environment. In a narrow environment, the hybrid sampling method produces a computation time of 24.52 seconds faster than the random sampling method. In the trap environment, the hybrid sampling method produces a computation time of 5.25 seconds faster than the random sampling method. From the test data, this hybrid sampling method can be an alternative sampling method to be used in the Informed-RRT* algorithm.*

Keywords : **:** *Boundary sampling, goal biaing sampling, hybrid sampling, Informed-RRT*, random sampling*

I. PENDAHULUAN

A. Latar Belakang

Perencanaan jalur merupakan salah satu proses yang terdapat di dalam pengendalian robot otonom. Proses ini sangat penting untuk memastikan bahwa jalur yang akan dilewati oleh robot otonom tersebut bebas dari tabrakan dimulai dari posisi awal hingga ke posisi tujuan. Perjalanan robot menuju ke titik tujuan tentu saja akan menemui berbagai macam hambatan. Oleh karena itu algoritma perencanaan jalur sangat perlu dilakukan untuk menghasilkan jarak yang optimal. Perencanaan jalur pada robot otonom memiliki banyak kegunaan seperti *Autonomus Vacum Cleaner Robots* [1], *Unmanned Aerial Vehicles* (UAV) menjelajahi lingkungan yang belum diketahui [2], pada bidang medis perencanaan jalur digunakan untuk mendesain obat yang lebih baik [3], kursi roda elektronik otonom di rumah sakit [4]. Pada pengaplikasian perencanaan jalur, algoritma yang sering digunakan adalah *Sampling Based Planning* (SBP). Algoritma SBP bekerja dengan mengambil sampel acak pada ruang pencarian [5], [6]. Contoh algoritma SBP yang sering digunakan yaitu *Rapidly-exploring Random Tree* (RRT) [3], [6] dan *Probabilistic Roadmap Method* (PRM). Algoritma RRT memiliki keunggulan dalam hal waktu komputasi yang rendah, tetapi solusi yang diberikan bersifat sub-optimal, sementara PRM memiliki waktu komputasi yang tinggi, tetapi menghasilkan solusi yang bersifat asimptotik optimal [7]. Karena kekurangan yang dimiliki oleh RRT, maka Karaman dan Frazzoli mengusulkan variasi dari algoritma RRT yaitu RRT* [8]. Algoritma RRT* ini menghasilkan solusi yang bersifat asimptotik optimal. Pengembangan dari algoritma RRT* menghasilkan algoritma *informed*-RRT* oleh Gammel dkk [9]. yang mana algoritma *informed*-RRT* akan melakukan pengambilan sample pada area yang berbentuk elips yang melingkupi titik awal dan titik akhir untuk memberikan jarak yang paling optimal.

Hal yang tidak kalah penting dalam SBP yaitu strategi pengambilan sample yang mana pengambilan sample merupakan inti dari SBP [6]. Pada awalnya algoritma RRT dan PRM diusulkan menggunakan strategi *random sampling* [3], [6], [10] yang mana strategi pengambilan sampel tersebut digunakan pada sebagian besar algoritma SBP termasuk algoritma *Informed*-RRT*. Strategi sampling tersebut dianggap sebagai kelemahan karena tidak menangkap konektivitas lingkungan yang sebenarnya. Hal ini dapat menyebabkan waktu komputasi dan kualitas jalur menjadi tidak optimal [10-12].

B. Tinjauan *State of art*

Permasalahan waktu komputasi yang tidak optimal dapat diatasi dengan yaitu seperti yang pernah dilakukan oleh Reza [10] yaitu pencarian solusi awal jalur dilakukan dengan menggunakan algoritma RRT karena menghasilkan waktu komputasi yang lebih rendah. Setelah solusi awal jalur jalur tersebut ditemukan maka jalur tersebut dioptimalisasi dengan menggunakan algoritma *Informed*-RRT*. Pada penelitian tersebut masih menggunakan metode *random sampling* sehingga waktu komputasi masih belum optimal. Pada penelitian ini penulis menggunakan metode *hybrid sampling* pada algoritma *informed-RRT** sebagai pengganti dari metode *random sampling* yang digunakan pada algoritma *Informed*-RRT*.

C. Tujuan

Penelitian ini bertujuan untuk mengusulkan metode pengambilan sampel alternatif untuk digunakan pada algoritma *Informed*-RRT*. Metode pengambilan sampel yang diusulkan yaitu metode *hybrid sampling.* Pengujian metode *hybrid sampling* ini dilakukan pada lingkungan *clutter, narrow,* dan *trap.* Performansi yang diukur dari pengujian ini adalah waktu komputasi. Hasil pengukuran performansi ini akan dibandingkan dengan algoritma *Informed-RRT** dengan metode *random sampling.*

D. Sistematika Pembahasan

Sistematika pada penulisan ini terdiri dari beberapa bagian. Pada bagian 2 akan menjelaskan perancangan metode *hybrid sampling* pada algoritma *informed-RRT**. Pada bagian 3 akan memaparkan hasil data yang didapat dari percobaan metode *hybrid sampling, random sampling, boundary sampling, goal biasing sampling* pada algoritma *informed-RRT**. Pada bagian 4 akan membahas kesimpulan yang didapat dari hasil percobaan.

II. METODOLOGI

Proses pada metode *hybrid sampling* yaitu dengan memilih salah satu metode pengambilan sampel yang telah diintegrasikan sebelumnya. Metode pengambilan sampel yang diintegrasikan

yaitu metode *goal biasing sampling*, metode *boundary sampling*, metode *random sampling*. Besarnya peluang terpilihnya salah satu metode sampling adalah tergantung dari nilai persentase yang telah ditetapkan masing-masing metode sampling yang telah diintegrasikan. Semakin besar nilai persentasenya maka peluang terpilihnya salah satu algoritma pada setiap iterasi akan semakin besar. Semakin kecil nilai persentase yang diberikan pada metode-metode sampling maka akan semakin kecil pula peluang terpilihnya salah satu algoritma pada setiap iterasinya. Algoritma metode *hybrid sampling* dapat dilihat pada **Algoritma 1.**

Algoritma 1: *Hybrid Sampling* (b_0, g_0)

```
1  x1 ← a random number; between 0 and 100
2  if x1 > 0 & x1 ≤ b0 then
3      sample ← BoundarySample
4  else if x1 > b0 & x1 ≤ b0 + g0 then
5      sample ← GoalBiasSample
6  else
7      sample ← UniformSample
8  end if
9  return sample.
```

Pada **Algoritma 1** parameter g_0 merupakan persentase dari metode *goal biasing sampling*. Parameter b_0 merupakan persentase dari metode *boundary sampling*. Baris ke 1 yang terdapat pada **Algoritma 1** merupakan proses untuk membangkitkan nilai acak (*x1*) antara 0 dan 100. Proses pada baris ke-2 yaitu untuk menguji apabila kondisi ($x_1 > 0$ & $x_1 \leq$ b0) terpenuhi maka *boundary sampling* akan digunakan. Pada baris ke-4 apabila ($x_1 > b_0$ & $x_1 \leq b_0 + g0$) terpenuhi maka *goal biasing sampling* akan digunakan. Jika kondisi pada baris ke-2 dan baris ke-4 tidak terpenuhi maka *random sampling* yang akan digunakan. Cara keja dari *random sampling* yaitu dengan mengambil nilai acak pada ruang pencarian. Algoritma *random sampling* dapat dilihat pada **Algoritma 2.**

Algoritma 2: *Random Sampling* (b_0, g_0)

```
1  q_rand ← a RandomConfig();
2  return q_rand
```

Proses pada *goal biasing sampling* yaitu dengan mengambil sampel langsung pada titik tujuan dengan frekuensi tertentu. Algoritma *goal biasing sampling* dapat dilihat pada **Algoritma 3**.

Algoritma 3: *GoalBiasSample* ()

```
1  rand ← a random number; between 0 - 100
2  if rand < k then
3      q_rand ← q_goal
4  else
5      q_rand ← RandomConfig();
6  end if
7  return q_rand
```

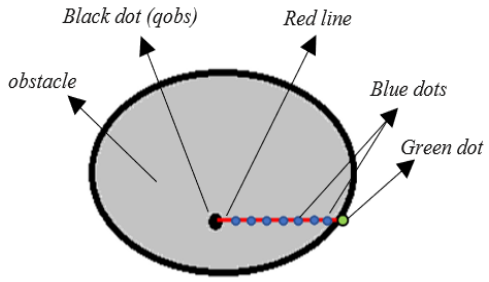
Parameter q_{goal} yang terdapat pada **Algoritma 3** merupakan titik tujuan. Pada **Algoritma 3**. Terdapat parameter *k* yang mana parameter ini merupakan besarnya persentase peluang untuk mengambil titik sampel langsung menuju ke titik tujuan. Besarnya nilai persentase yang direkomendasikan adalah 0 sampai 10%. Variabel *rand* merupakan nilai acak yang dibangkitkan mulai dari 0 sampai 100. q_{goal} merupakan titik tujuan. Jika *random number* lebih kecil dari k, maka q_{rand} akan bernilai q_{goal}, jika tidak maka nilai q_{rand} akan bernilai titik acak pada ruang pencarian. Pada perancangan ini penulis menggunakan persentase sebesar 10% yang juga digunakan oleh penelitian sebelumnya oleh Mohamed Elbanhawi [6]. Algoritma *boundary nodes* dapat dilihat pada **Algoritma 4.**

Algoritma 4: *BoundaryNodes*(*g=0, c, x=false,* arr[ ], n)

```
1  rand q_obs ← random node in obstacle
2  c= q_obs
3  while g ≤ 360 then
4      while x = false then
5          if c CollisonFree
6              arr [ ] =, c ← add c to array
7              x= true
8          else
9              c+=0,01
10     g+=n
11     x=false
12     c= q_obs
13    return arr [ ]
```

Pada **Algoritma 4** proses pada baris ke-1 yaitu membentuk titik acak di dalam *obstacle* (q_{obs}). Proses pada baris ke-2 akan memeriksa apakah sudut g pada titik q_{obs} lebih kecil sama dengan 360 derajat. Proses pada baris ke-4 akan memeriksa nilai x apakah bernilai *true* atau *false*. Proses pada baris ke-5 akan memeriksa apakah c berada berada

diluar *obstacle* atau tidak. Apabila nilai c berada di luar *obstacle* maka tambahkan nilai c ke dalam *array* (arr) dan *x* akan menjadi true. Jika c masih berada di dalam *obstacle* maka nilai c akan ditambah dengan nilai 0,01 menjauhi titik q_{obs} pada sudut g. Penggunaan nilai 0,01 bertujuan agar pencarian batas *obstacle* menjadi lebih akurat. Semakin kecil nilainya maka akan semakin akurat titik batas *obstacle* yang akan didapatkan. Nilai n merupakan penentu banyaknya titik sampel yang akan diambil pada setiap permukaan *obstacle*. Pada perancangan ini penulis mengambil titik sampel pada permukaan *obstacle* sebanyak 20 titik pada setiap *obstacle*. Pengambilan titik sampel sebanyak 20 titik setiap *obstacle* bertujuan umtuk mempermudah pohon pencarian menuju titik satu menuju ke titik lainnya. nilai n yang digunakan pada perancangan ini adalah n=18 yang di dapat dari perhitungan 360 derajat / 20 titik = 18. Ilustrasi pada **Algoritma 4** dapat dilihat pada **Gambar 1.**



Gambar 1 ilustrasi pencarian titik batas pada *obstacle*

Titik warna biru pada **Gambar 1** merupakan titik-titik pengujian untuk mencari batas dari *obstacle.* Titik biru ini merupakan variabel c yang terdapat pada **Algoritma 4**. Garis berwarna merah pada **Gambar 1** merupakan garis untuk sudut 0 derajat pada titik q_{obs} (*Black dot*). Titik warna hijau pada **Gambar 1** merupakan titik sampel pada permukaan *obstacle* yang didapatkan setelah beberapa pengujian pada titik biru dilakukan. Algoritma pemilihan titik sampel pada permukaan *obstacle* yang telah di simpan pada variabel *arr* [ ] pada **Algoritma 4** dapat dilihat pada **Algoritma 5**.

Algoritma 5: *BoundarySample(arr [ ])*

```
1  rand ← random number; between 0 and 100
2  if rand ≤ 10 then
3      sample ← a RandomConfig().
4  else
5      randarr ← random number; between 0
       and arr length
6      sample ← arr [randarr]
7  return sample
```

Proses pada **Algoritma 5** baris ke-1 yaitu membangkitkan angka acak (*rand*) dari 0 sampai 100. Kemudian proses pada baris ke-2 yaitu menguji apabila *rand* < 10 maka ambil sampel acak pada ruang pencarian. Proses pada baris ke-4 Apabila *rand > 10 maka* pilih salah satu titik sampel pada permukaan *obstacle* yang terdapat pada variabel *arr* [ ]. besarnya pemilihan titik sampel di permukaan obstacle pada perancangan ini yaitu 90% dan persentase pengambilan titik acak pada ruang pencarian yaitu 10%. Besarnya persentase yang digunakan pada perancangan ini bertujuan agar proses pada **Algoritma 5** tidak hanya memilih titik sampel pada permukaan *obstacle* saja, tetapi juga sesekali dapat mengambil titik sampel pada ruang bebas apabila ternyata titik tujuan berada di ruang pencarian bebas.

III. HASIL DAN PEMBAHASAN

A. Pengujian metode *goal biasing sampling, boundary sampling, random sampling* pada algoritma *Informed*-RRT*

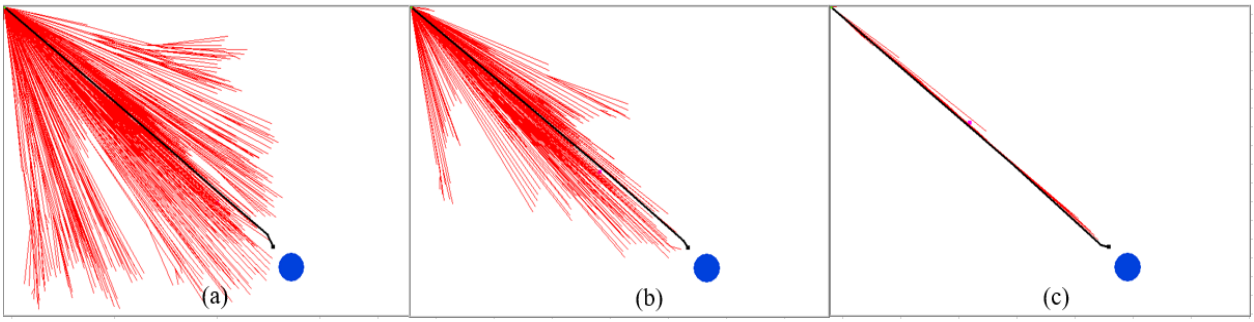
Agar analisa pada percobaan metode *hybrid sampling* semakin mudah dipahami, maka penulis melakukan percobaan untuk melihat kelebihan dari metode pengambilan sampel *goal biasing sampling* dan *boundary sampling*. Metode Percobaan pertama dilakukan dengan mengambil data sebanyak 30 data. Setiap data akan dilakukan sebanyak 1000 iterasi. Percobaan pertama yaitu dengan menempatkan titik tujuan dekat dengan *obstacle* dan berhadapan langsung dengan titik awal. Percobaan pertama dapat dilihat pada **Gambar 2.** Kemudian percobaan kedua yaitu dengan menempatkan titik tujuan sedikit lebih jauh dari *obstacle*. Percobaan kedua dapat dilihat pada **Gambar 3.** Kemudian percobaan ketiga yaitu dilakukan hanya untuk menguji metode *boundary sampling* yaitu dengan menempatkan titik tujuan yang membelakangi *obstacle.* Percobaan ketiga dapat dilihat pada **Gambar 4.**

Pada **Gambar 2** pohon pencarian pada metode *random sampling* akan menyebar kesegala arah karena pengambilan titik sampel pada metode *random sampling* dilakukan secara acak pada ruang pencarian. Hal ini menjadi penyebab waktu komputasi pada metode *random sampling* menjadi sangat lama. Bentuk pohon pencarian pada metode *goal biasing sampling* hampir sama seperti bentuk pohon pencarian pada metode *random sampling*. Perbedaannya terletak pada pengambilan titik sampel pada metode *goal biasing sampling* yang

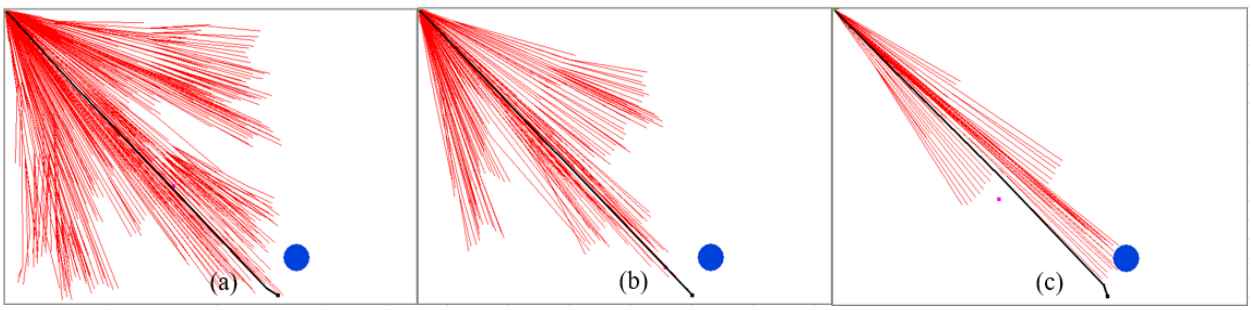
sesekali mengarah langsung ke titik tujuan, sehingga mempercepat waktu komputasi. Bentuk pohon pencarian pada metode *boundary sampling* akan mengarah langsung menuju *obstacle* karena pengambilan titik sampel diambil pada permukaan *obstacle*, sekaligus menemukan titik tujuan yang berada didekat *obstacle*. Sifat dari *boundary sampling* ini sama seperti percobaan yang dilakukan oleh Nancy [13]. Hal ini menyebabkan waktu komputasi pada metode *sampling boundary* menjadi sangat cepat.

Pada **Gambar 3** pohon pencarian pada metode *random sampling* berada menyebar ke semua ruang pencarian, karena metode tersebut akan mengambil titik sampel secara acak diruang pencarian. Sehingga waktu komputasi pada metode *random sampling* menjadi lebih lama tetapi tidak lebih lama bila dibandingkan dengan metode *boundary sampling.* pada metode *goal biasing sampling* pengambilan sampel sama seperti metode *random sampling*, hanya saja pada metode *goal biasing sampling* akan menempatkan titik sampelnya langsung pada titik tujuan. Sehingga pohon pencarian pada metode *goal biasing sampling* dapat mengarah langsung ke titik tujuan. Sifat dari *goal biasing sampling* ini sama seperti yang dilakukan oleh Mohamed Elbanhawi [6]. Pada metode *boundary sampling* pohon pencarian akan mengarah langsung menuju ke permukaan *obstacle* dan sesekali pohon pencarian mengarah acak pada ruang pencarian. Hal ini menyebabkan pohon pencarian pada metode *boundary sampling* menjadi sulit menemukan titik tujuan bahkan gagal menemukan titik tujuannya.

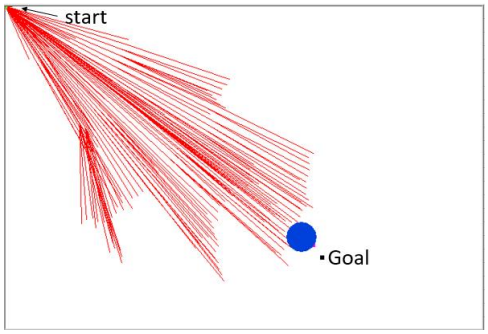
Pada **Gambar 4** Pengujian dilakukan sebanyak 1000 iterasi. Kondisi tersebut merupakan kondisi di mana metode *boundary sampling* tidak dapat menemukan titik tujuannya. Metode *boundary sampling* tetap akan mengambil sampel merata pada permukaan *obstacle* akan tetapi pohon pencarian tidak dapat langsung menuju kebagian sisi belakang yang berdekatan dengan titik tujuan karena terhalang oleh *obstacle* itu sendiri. Pada metode *boundary sampling* ini juga melakukan sebagian kecil pengambilan titik sampel acak pada ruang pencarian tetapi tetap tidak mampu menemukan titik tujuannya karena sebagian besar pengambilan titik sampel diambil pada permukaan *obstacle*. Tabel hasil perbadingan waktu komputasi dari metode *goal biasing sampling*, *boundary sampling*, dan *random sampling* pada kondisi titik tujuan dekat dan sedikit lebih jauh dari *obstacle* dapat dilihat pada **Tabel I-II. Tabel I** merupakan perbandingan waktu komputasi dari metode *goal biasing sampling*, *boundary sampling*, dan *random sampling* pada kondisi titik tujuan dekat dari *obstacle*. **Tabel II** merupakan perbandingan waktu komputasi dari metode metode *goal biasing sampling*, *boundary sampling*, dan *random sampling* pada kondisi titik tujuan sedikit lebih jauh dari *obstacle*.



Gambar 2. Perbandingan pohon pencarian pada kondisi titik tujuan berada di dekat *obstacle* dan berhadapan langsung dengan titik awal: *random sampling* (a), *goal biasing sampling* (b), *boundary sampling* (c).



Gambar 3. Perbandingan pohon pencarian pada kondisi titik tujuan berada sedikit lebih jauh dari *obstacle* dan berhadapan langsung dengan titik awal: *random sampling* (a), *goal biasing sampling* (b), *boundary sampling* (c).



Gambar 4. Pohon pencarian pada kondisi titik tujuan membelakangi *obstacle* menggunakan *boundary sampling*

Pada **Tabel II** metode *boundary sampling* mengalami 14 kali kegagalan dari 30 data yang didapatkan. dalam pencarian untuk menemukan titik tujuan. Hal ini disebabkan karena karena sebagian besar titik sampel yang diambil berada pada permukaan *obstacle* dan hanya sebagian kecil mengambil titik sampel acak pada ruang pencarian.

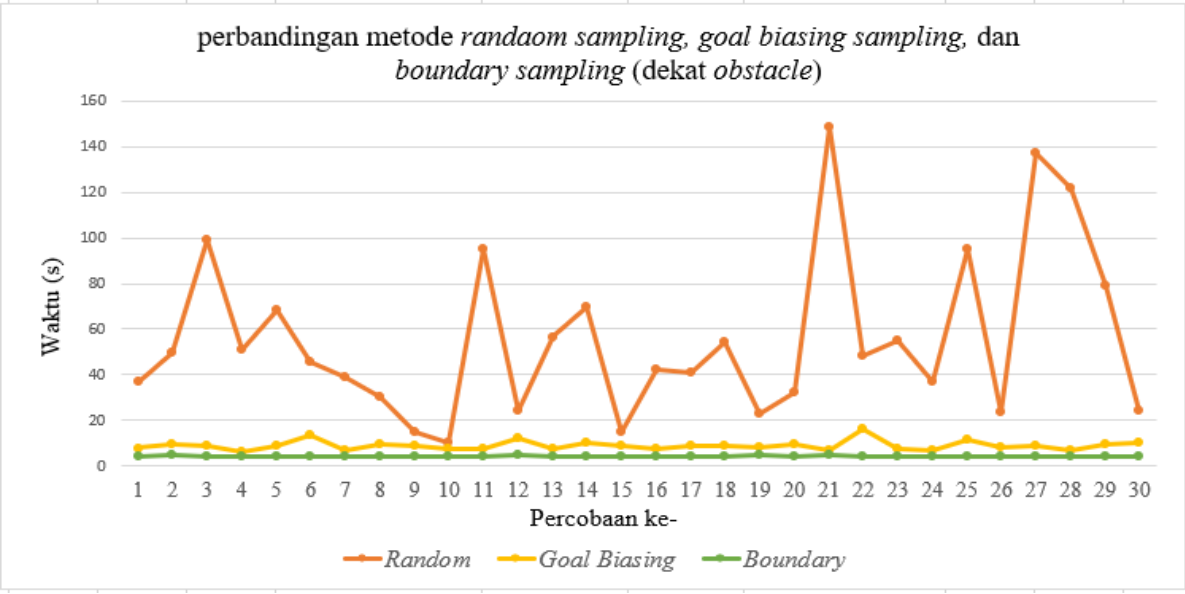
Grafik perbandingan metode *randaom sampling*, *goal biasing sampling*, dan *boundary sampling* saat titik tujuan berada dekat dengan *obstacle* dapat dilihat pada **Gambar 5.** Grafik perbandingan metode *random sampling*, *goal biasing sampling*, dan *boundary sampling* saat titik tujuan berada sedikit lebih jauh dengan *obstacle* dapat dilihat pada **Gambar 6.**

Tabel I Perbandingan waktu komputasi dari metode *goal biasing sampling*, *boundary sampling*, dan *random sampling* pada kondisi titik tujuan dekat dari *obstacle*.

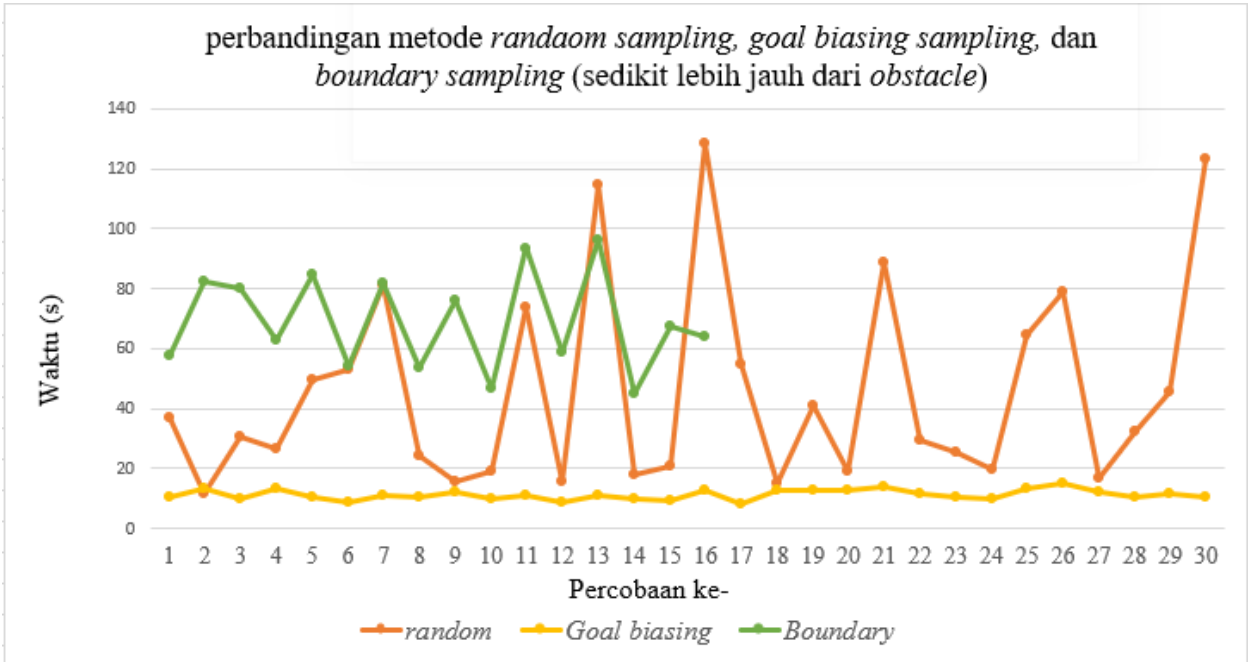
data	*random*	*Goal biasing*	*Boundary*
	Waktu (s)	Waktu (s)	Waktu (s)
Minimum	9,93	6,42	3,91
Maksimum	148,33	16,05	4,81
Rata-rata	55,53	8,99	4,27
STD	35,53	2,1	0,21

Tabel II. Perbandingan waktu komputasi dari metode metode *goal biasing sampling*, *boundary sampling*, dan *random sampling* pada kondisi titik tujuan sedikit lebih jauh dari *obstacle*.

data	*random*	*Goal biasing*	*Boundary*
	Waktu (s)	Waktu (s)	Waktu (s)
Minimum	11,84	8,42	45,21
Maksimum	128,45	13,93	96,21
Rata-rata	43,19	11,19	69,54
STD	33,32	1,6	15,68



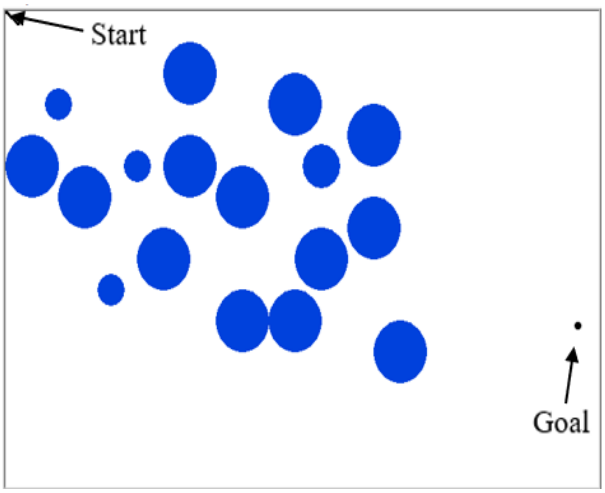
Gambar 5. Perbandingan metode *random sampling*, *goal biasing sampling*, dan *boundary sampling* (dekat *obstacle*)



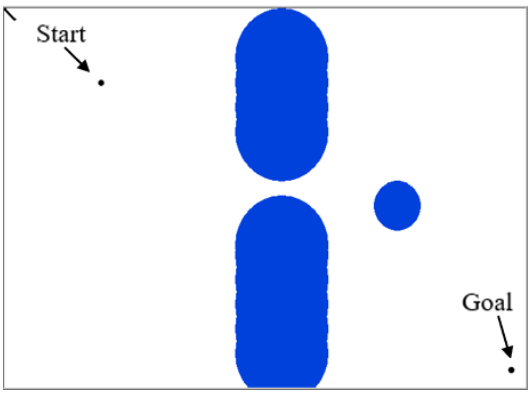
Gambar 6. Perbandingan metode *random sampling*, *goal biasing sampling*, dan *boundary sampling* (sedikit lebih jauh dari *obstacle*)

B. Pengujian metode *random sampling* dan *hybrid sampling* pada algortima *Informed-RRT**

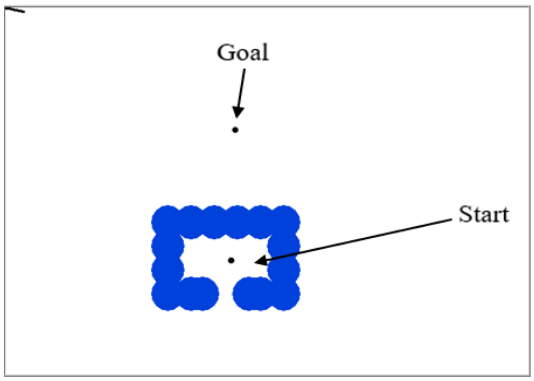
Pengujian ini dilakukan pada lingkungan yang umum digunakan yaitu *clutter*, *narrow*, dan *trap*. Parameter yang akan dibandingkan adalah waktu komputasi. Pada percobaan ini dilakukan pengambilan data sebanyak 30 data. Setiap data dilakukan percobaan dengan iterasi sebanyak 1000 iterasi. Waktu komputasi yang terdapat pada **Tabel III-VIII** merupakan waktu komputasi jalur terpendek yang didapatkan dari masing-masing metode *random sampling* dan *hybrid sampling* setelah iterasi berakhir. Lingkungan *clutter*, *narrow*, dan *trap* dapat dilihat pada **Gambar 7-9.**



Gambar 7. Contoh lingkungan pengujian *clutter*



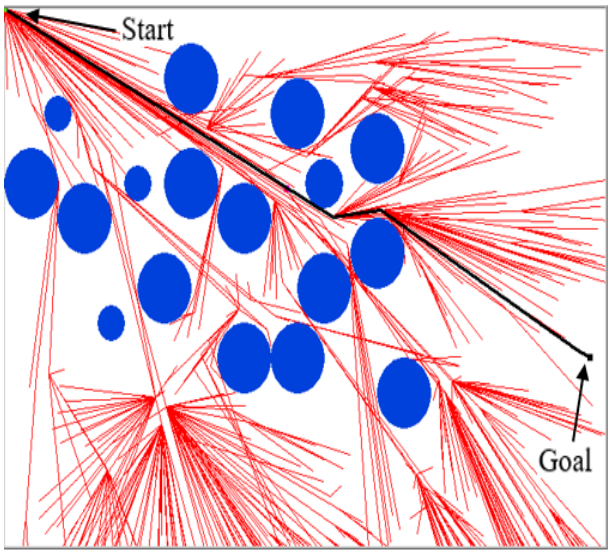
Gambar 8. Contoh lingkungan pengujian *Narrow*



Gambar 9. Contoh lingkungan pengujian *Trap*

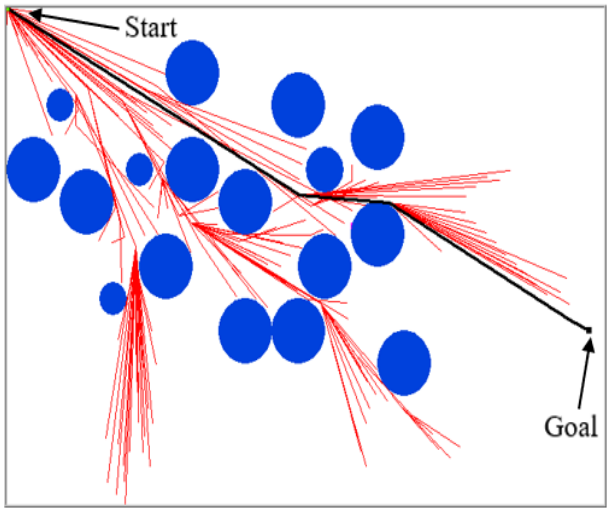
Pengujian metode *hybrid sampling* dan metode *random sampling* menggunakan algoritma *Informed-RRT** pada lingkungan *clutter* ditunjukkan pada **Gambar 10 - 11.** Pada **Gambar 10** merupakan bentuk pohon pencarian pada metode *random sampling*. **Gambar 11** merupakan

bentuk pohon pencarian pada metode *hybrid sampling*.



Gambar 10. Bentuk pohon pencarian pada metode *random sampling* pada lingkungan *clutter*

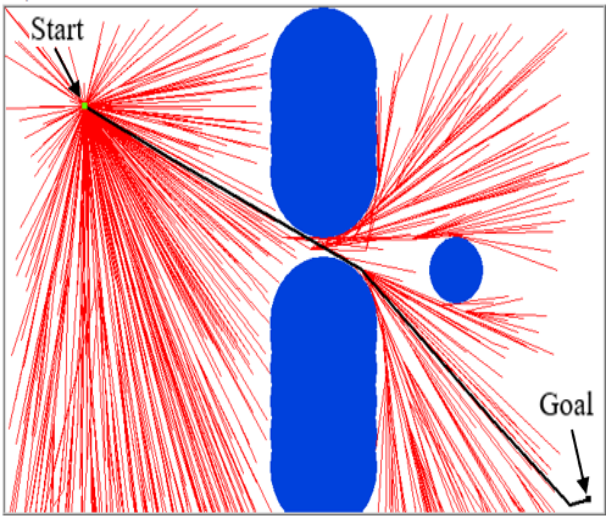
Pada **Gambar 10** terlihat bahwa cabang pada pohon pencarian menyebar ke segala arah. Hal ini disebabkan karena metode *random sampling* akan selalu mengambil sampel acak pada ruang pencarian sehingga beberapa cabang bergerak menjauhi titik tujuan. Hal ini menyebabkan waktu komputasi menjadi lebih lama.



Gambar 11. Bentuk pohon pencarian pada metode *hybrid sampling* pada lingkungan *clutter*

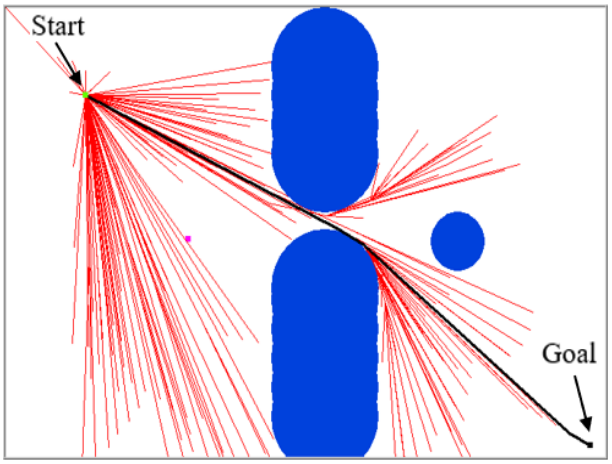
Bentuk pohon pencarian pada metode *hybrid samping* Pada **Gambar 11**, pohon pencarian akan langsung menuju ke arah *obstacle* hal itu disebabkan karena pengaruh dari metode *boundary sampling* yang mengambil titik sampel pada permukaan *obstacle*. *Goal biasing sampling* dan *boundary sampling* pada lingkungan *clutter* membuat pohon pencarian lebih terarah menuju titik tujuan sehingga waktu komputasi pada metode *hybrid sampling* menjadi lebih cepat bila dibandingkan dengan metode *random sampling.*

Pengujian metode *hybrid sampling* dan metode *random sampling* menggunakan algoritma *Informed-RRT** pada lingkungan *narrow* ditunjukkan pada **Gambar 12-13.** Pada **Gambar 12** merupakan bentuk pohon pencarian pada metode *random sampling*. **Gambar 13** merupakan bentuk pohon pencarian pada metode *hybrid sampling*.



Gambar 12. Bentuk pohon pencarian pada metode *random sampling* pada lingkungan *narrow*

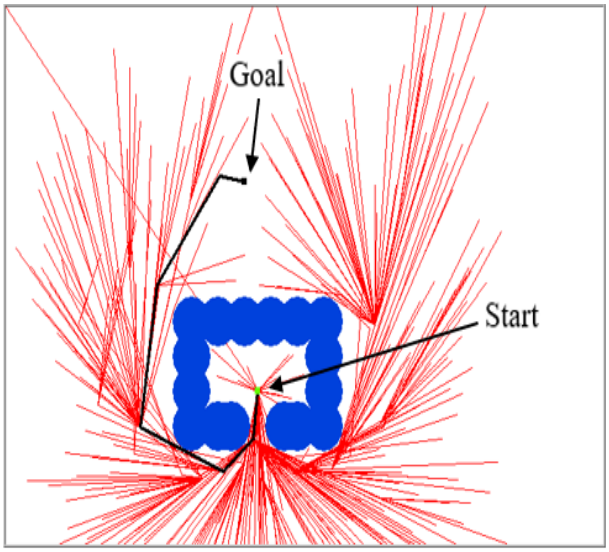
Pada **Gambar 12** pohon pencarian pada metode *random sampling* sebelum melewati celah sempit akan mengarahkan cabang pohon pencarian ke segala arah. Setelah berhasil melewati celah, cabang pohon pencarian pada *random sampling* akan tetap menuju ke segala arah bahkan sebagian cabang bergerak menjauhi titik tujuan. Hal ini yang menyebabkan waktu komputasi menjadi lebih lama.



Gambar 13 Bentuk pohon pencarian pada metode *hybrid sampling* pada lingkungan *narrow*

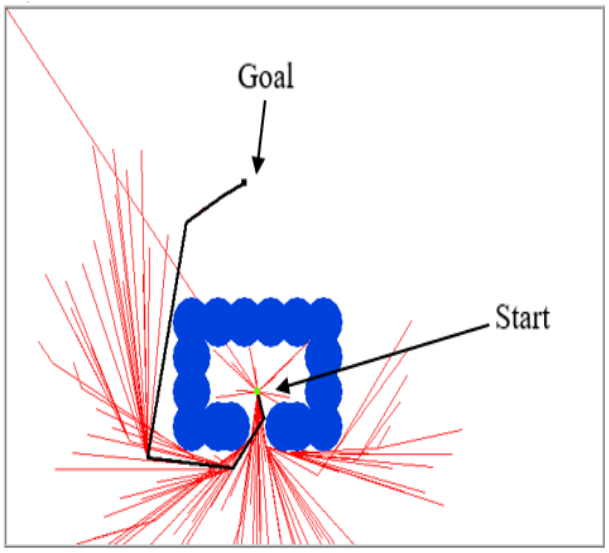
Pada **Gambar 13** cabang pohon pencarian sebelum melawati celah sempit terlihat lebih sedikit bila dibandingkan dengan metode *random sampling* pada **Gambar 12.** Hal ini disebabkan karena sebagian cabang pohon pencarian yang dilakukan pada metode *hybrid sampling* akan menuju langsung ke permukaan *obstacle* yang menyebabkan peluang untuk dapat melewati celah sempit menjadi lebih besar. Cabang pohon pencarian setelah melewati celah sempit akan cenderung mengarah langsung ke titik tujuan sehingga waktu komputasi menjadi lebih cepat bila dibandingkan dengan metode *random sampling*.

Pengujian metode *hybrid sampling* dan metode *random sampling* menggunakan algoritma *Informed-RRT** pada lingkungan *trap* ditunjukkan pada **Gambar 14-15.** Pada **Gambar 14** merupakan bentuk pohon pencarian pada metode *random sampling*. **Gambar 15** merupakan bentuk pohon pencarian pada metode *hybrid sampling*.



Gambar 14. Bentuk pohon pencarian pada metode *random sampling* pada lingkungan *trap*

Pada **Gambar 14** cabang pohon pencarian pada metode *random sampling* memiliki peluang yang besar untuk keluar dari *obstacle trap.* Karena pengambilan titik sampel pada metode tersebut diambil secara acak di seluruh ruang pencarian. Setelah cabang pada pohon pencarian tersebut berhasil keluar dari *obstacle trap,* maka cabang pohon pencarian tersebut cenderung mengarah ke segala arah pada ruang pencarian. Hal ini menyebabkan beberapa cabang yang tidak mengarah ke titik tujuan menjadi tidak berguna.



Gambar 15 bentuk pohon pencarian pada metode *hybrid sampling* dilingkungan *trap*

Pada **Gambar 15** cabang pohon pencarian pada metode *hybrid sampling* memiliki peluang yang lebih kecil untuk keluar dari celah sempit bila dibandingkan dengan metode *random sampling.* Hal ini disebabkan karena beberapa pengambilan titik sampel pada metode ini berada pada permukaan *obstacle* yang menyebabkan waktu komputasi menjadi lebih lama. Setelah cabang pada pohon pencarian berhasil keluar dari celah sempit maka cabang pohon pencarian cenderung menuju ke titik tujuan. Pada kasus pengujian dilingkungan *trap* ini metode *boundary sampling* tidak terlalu dibutuhkan pada proses pencarian jalur karena pohon pencarian menjadi sulit menemukan jalan keluar dari dalam *trap*. Tabel perbandingan waktu komputasi menggunakan metode *random sampling* dan *hybrid sampling* pada lingkungan *clutter*, *narrow*, dan *trap* ditunjukkan pada **Tabel III-V**.

Tabel III. Perbandingan waktu komputasi pada lingkungan *clutter*

Data	*Random sampling*	*Hybrid sampling* (*goal biasing* 50%, *boundary* 30%, *random* 20%)
	Waktu (s)	Waktu (s)
Minimum	13,01	10,15
Maksimum	108,81	21,23
Rata-rata	39,00	14,03
STD	21,17	3,02

Tabel IV. Perbandingan waktu komputasi pada lingkungan *narrow*

Data	*Random sampling*	*Hybrid sampling* (*goal biasing* 50%, *boundary* 30%, *random* 20%)
	Waktu (s)	Waktu (s)
Minimum	23,53	10,11
Maksimum	53,50	47,55
Rata-rata	40,34	21,46
STD	7,97	10,54

Tabel V. Perbandingan waktu komputasi pada lingkungan *trap*

Data	*Random Sampling*	*Hybrid sampling* (*goal biasing* 50%, *boundary* 30%, *random* 20%)
	Waktu (s)	Waktu (s)
Minimum	15,25	8,23
Maksimum	42,15	35,76
Rata-rata	23,20	19,45
STD	6,24	5,38

Pada **Tabel III-V** menunjukkan bahwa penggunaan metode *hybrid sampling* pada algoritma *Informed*-RRT* dapat memberikan waktu komputasi yang lebih cepat bila dibandingkan metode *random sampling* pada algoritma *Informed*-RRT*. Penulis mencoba melakukan beberapa percobaan dengan mengubah nilai persentase pada metode *hybrid sampling* untuk mengetahui apakah waktu komputasi metode *hybrid sampling* pada algoritma *informed*-RRT* dapat menjadi lebih baik atau lebih buruk. Perbandingan waktu komputasi dari kombinasi *hybrid sampling* pada lingkungan *clutter*, *narrow*, dan *trap* dapat dilihat pada **Tabel VI-VIII. Tabel VI** merupakan perbandingan waktu komputasi dari dua kombinasi *hybrid sampling* yang dilakukan pada lingkungan *clutter*.

Tabel VI. perbandingan waktu komputasi dari dua kombinasi *hybrid sampling* yang dilakukan pada lingkungan *clutter*.

Data	*Hybrid sampling (goal biasing* 70%, *boundary* 10%, *random* 20%)	*Hybrid sampling (goal biasing* 90%, *boundary* 5%, *random* 5%)
	Waktu (s)	Waktu (s)
Minimum	9,73	9,28
Maksimum	25,44	19,1
Rata-rata	13,04	12,6
STD	3,07	2,3

Tabel VII merupakan perbandingan waktu komputasi dari dua kombinasi *hybrid sampling* yang dilakukan pada lingkungan *narrow*. **Tabel VIII** merupakan perbandingan waktu komputasi dari dua kombinasi *hybrid sampling* yang dilakukan pada lingkungan *trap*.

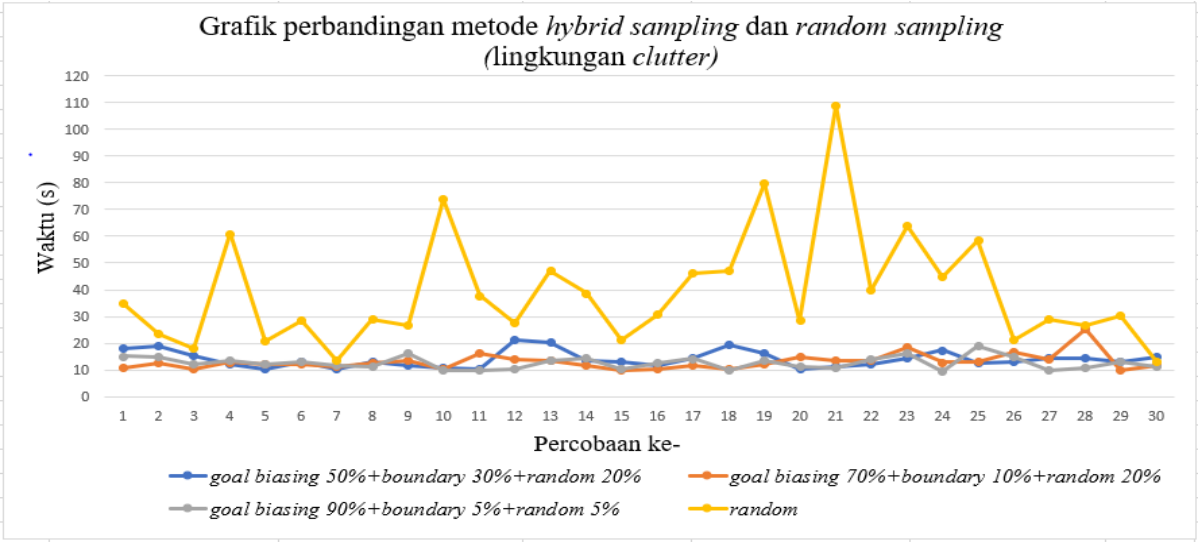
Tabel VII. Perbandingan waktu komputasi dari dua kombinasi *hybrid sampling* yang dilakukan pada lingkungan *narrow*.

Data	*Hybrid sampling (goal biasing* 70%, *boundary* 10%, *random* 20%)	*Hybrid sampling (goal biasing* 90%, *boundary* 5%, *random* 5%)
	Waktu (s)	Waktu (s)
Minimum	10,17	9,31
Maksimum	34,78	41,32
Rata-rata	16,84	15,82
STD	5,8	7,45

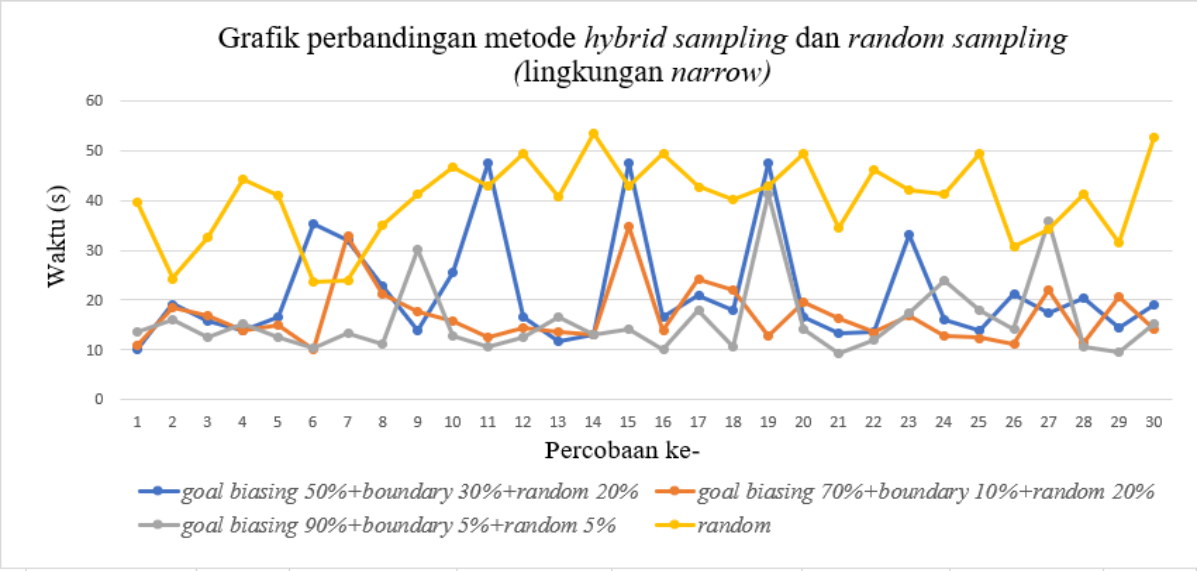
Tabel VIII. Perbandingan waktu komputasi dari dua kombinasi *hybrid sampling* yang dilakukan pada lingkungan *trap*.

Data	*Hybrid sampling (goal biasing* 70%, *boundary* 10%, *random* 20%)	*Hybrid sampling (goal biasing* 90%, *boundary* 5%, *random* 5%)
	Waktu (s)	Waktu (s)
Minimum	9,4	9,34
Maksimum	40,55	24,64
Rata-rata	17,2	14,2
STD	6,14	3,03

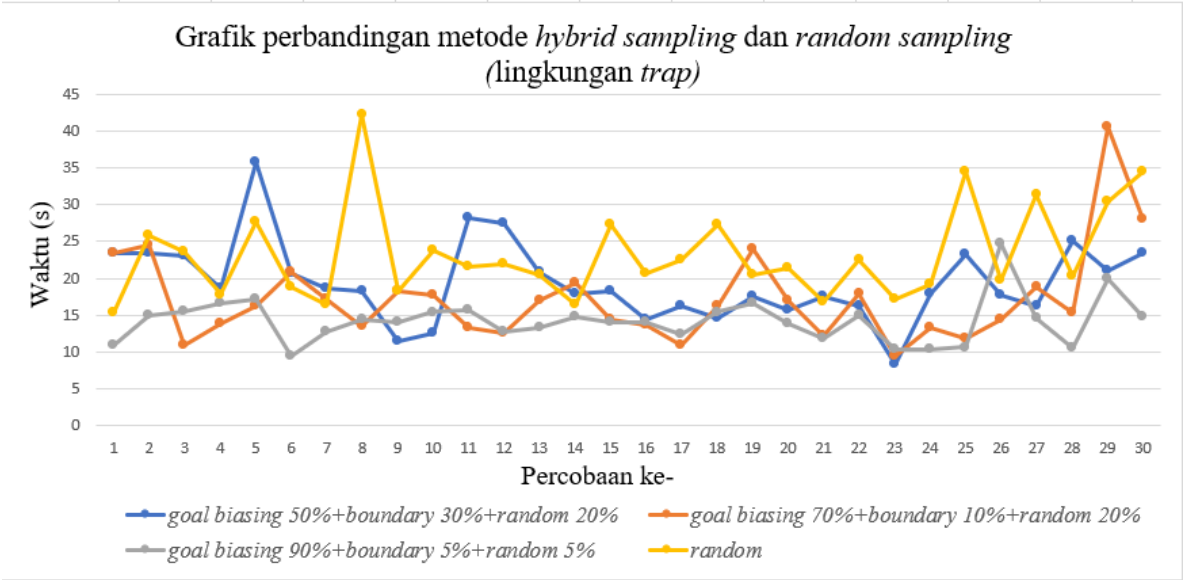
Hal yang menarik dari hasil persentase kombinasi metode *hybrid sampling* percobaan yang dilakukan pada **Tabel III-VIII** yaitu semakin besar persentase *goal biasing sampling* maka waktu komputasi yang dihasilkan akan semakin cepat. Persentase kombinasi metode *hybrid sampling* yang menghasilkan waktu komputasi paling cepat yaitu *goal biasing sampling* 90% + *boundary sampling* 5% + *random sampling* 5%. Grafik perbandingan metode hybrid sampling dapat dilihat pada **Gambar 16-18.** Pada **Gambar 16** merupakan grafik perbandingan metode *hybrid sampling* dan *random sampling* pada lingkungan *clutter*. Pada **Gambar 17** merupakan grafik perbandingan metode *hybrid sampling* dan *random sampling* pada lingkungan *narrow*. Pada **Gambar 18** merupakan grafik perbandingan metode *hybrid sampling* dan *random sampling* pada lingkungan *trap*.



Gambar 16. Grafik perbandingan metode *hybrid sampling* dan *random sampling* pada lingkungan *clutter*



Gambar 17. Grafik perbandingan metode *hybrid sampling* pada lingkungan narrow



Gambar 18. Grafik perbandingan metode *hybrid sampling* pada lingkungan narrow

IV. KESIMPULAN

Data dari hasil pengujian menunjukkan bahwa metode *hybrid sampling* dapat menemukan titik tujuan lebih cepat bila dibandingkan dengan metode *random sampling.* Nilai persentase kombinasi pada metode *hybrid sampling* akan mempengaruhi waktu komputasi pada algoritma *Informed*-RRT*. Dari hasil pengujian pada lingkungan *clutter, narrow,* dan *trap* nilai persentase kombinasi pada metode *hybrid sampling* yang menghasilkan waktu komputasi tercepat yaitu *goal biasing sampling* 90% + *boundary sampling* 5% + *random sampling* 5%. Metode *hybrid sampling* dengan persentase kombinasi *goal biasing sampling* 90% + *boundary sampling* 5% + *random sampling* 5% menghasilkan waktu komputasi 26,4 detik lebih cepat bila dibandingkan dengan metode *random sampling* pada lingkungan *clutter.* Pada lingkungan *narrow* metode *hybrid sampling* menghasilkan waktu komputasi 24,52 detik lebih cepat bila dibandingkan dengan metode *random sampling*. Pada lingkungan *trap* metode *hybrid sampling* menghasilkan waktu komputasi 5,25 detik lebih cepat bila dibandingkan dengan metode *random sampling*. Data-data yang telah didapatkan dari hasil pengujian pada lingkungan *clutter, narrow,* dan *trap* menunjukkan bahwa metode *hybrid sampling* dapat digunakan sebagai metode pengambilan sampel alternatif pada algoritma *informed*-RRT*.

DAFTAR PUSTAKA

[1] K. M. Hasan, Abdullah-Al-Nahid and K. J. Reza, "Path planning algorithm development for autonomous vacuum cleaner robots," in *2014 International Conference on Informatics, Electronics \& Vision (ICIEV)*, hal. 1-6, 2014.

[2] J. A. Cobano, D. Alejo and G. Heredia, "Efficient collision-free trajectory planning for WSN data collection with Unmanned Aerial Vehicles," *IFAC Proceedings Volumes*, vol. 46, no. 30, hal. 220-225, 2013.

[3] S. M. LaValle, Planning Algorithm, Cambridge university press, 2006.

[4] Y. Jung, Y. Kim, W. H. Lee, M. S. Bang and Y. Kim, "Path Planning Algorithm for an Autonomous Electric Wheelchair in Hospitals," *IEEE Access*, vol. 8, hal. 208199-208213, 2020.

[5] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli and S. Teller, "Anytime motion planning using the RRT," in *2011 IEEE International Conference on Robotics and Automation*, hal. 1478-1483, 2011.

[6] M. Elbanhawi and M. Simic, "Sampling-Based Robot Motion Planning : A Review," *IEEE access*, vol. 2, hal. 56-77, 2014.

[7] M. A. Rajasa Pohan, B. R. Trilaksono, S. P. Santosa and A. S. Rohman, "Algoritma Perencanaan Jalur Kendaraan Otonom di Lingkungan Perkotaan dari Sudut Pandang Filosofi Kuhn dan Filosofi Popper Path Planning Algorithm for Autonomous Urban Vehicles from the Viewpoint of Kuhn's Philosophy and Popper's Philosophy," vol. 7, no. 2, hal. 145-156, 2019

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, hal. 846-894, 2011.

[9] J. D. Gammel, S. S. Srinivasa and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, hal. 2997-3004, 2014.

[10] R. Mashayekhi, M. Y. Idna Idris and M. H. Anisi, "Hybrid rrt: A semi-dual-tree rrt-based motion planner," *IEEE Access*, vol. 8, hal. 18658-18668, 2020.

[11] C. Zhong and H. Liu, "A region-specific hybrid sampling method for optimal path planning," *International Journal of Advanced Robotic Systems*, vol. 13, p. 71, 2016.

[12] A. Kannan, P. Gupta, R. Tiwari, S. Prasad, A. Khatri and R. Kala, "Robot motion planning using adaptive hybrid sampling in probabilistic roadmaps," *Electronics*, vol. 5, no. 2, p. 16, 2016.

[13] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proceedings of IEEE international conference on robotics and automation*, hal. 113-120, 1996.