

Optimalisasi Solusi Algoritma *Rapidly-exploring Random Tree* (RRT) dengan Algoritma Dijkstra pada Perencanaan Rute di Ruang Terbuka Tak-diskrit

Malik Akbar Hashemi Rafsanjani - 13520105¹

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

¹13520105@std.stei.itb.ac.id

Abstract—Algoritma *Rapidly-exploring Random Tree* (RRT) merupakan salah satu algoritma yang banyak dipakai dalam perencanaan rute. Algoritma ini sangat cepat dibandingkan algoritma perencanaan rute lainnya, tetapi solusi yang dihasilkan merupakan solusi yang sub-optimal. Tulisan ini mencoba melakukan optimalisasi pada solusi yang dihasilkan oleh algoritma RRT dengan menggabungkannya bersama algoritma Dijkstra. Berdasarkan data yang telah dikumpulkan dan dianalisis, pendekatan yang dilakukan cukup berhasil untuk menekan jarak rute yang dihasilkan, tetapi juga meningkatkan waktu Komputasi program walaupun tidak terlalu banyak. Pendekatan ini cocok dilakukan untuk perencanaan rute yang memerlukan waktu yang tetap rendah, tetapi menghasilkan solusi yang lebih optimal dibanding algoritma RRT aslinya.

Keywords—Perencanaan rute, Optimalisasi, RRT, Dijkstra

I. PENDAHULUAN

Perencanaan rute merupakan permasalahan yang sangat sering dihadapi di dunia informatika. Walaupun permasalahan ini merupakan permasalahan yang sudah cukup lama, dan sudah terdapat banyak solusi-solusi efektif dan efisien dalam penanganannya. Namun, permasalahan perencanaan rute masih sangat banyak diteliti dan banyak bermunculan solusi-solusi baru serta variasi-variasi dari solusi-solusi tersebut. Pemecahan masalah ini banyak digunakan di banyak bidang, seperti perencanaan gerak pada robot, pencarian rute terpendek pada Google Maps, dll.

Salah satu variasi permasalahan perencanaan rute yang masih banyak diteliti ialah perencanaan rute di ruang terbuka. Oleh karena itu, banyak sekali algoritma-algoritma untuk memecahkan masalah tersebut. Namun, tidak seperti perencanaan rute pada ruang diskrit yang telah ada solusi yang sangat *well established*, seperti dengan A* atau Dijkstra, bisa dibilang perencanaan rute di ruang terbuka belum ada algoritma yang *well established* untuk menangani hamper semua kasus. Juga, seringkali banyak kelemahan dari algoritma-algoritma tersebut, baik dari sisi performansi maupun efektivitas. Hal tersebut disebabkan banyak dari algoritma tersebut yang perlu melakukan diskritisasi pada ruang kerjanya sehingga menurunkan performansi, seperti pada algoritma AD* [1]. Salah satu algoritma yang tidak memerlukan diskritisasi ialah Ant Colony Optimization (ACO), tetapi metode ini dapat terjebak

pada solusi minimum lokal dan dapat berkinerja buruk pada kasus lingkungan yang memiliki celah sempit [2]. Metode-metode pendekatan lain juga sudah dicoba, seperti dengan perencanaan reaktif berbasis sensor, tetapi susah untuk digunakan pada perencanaan rute yang global atau jarak jauh [3].

Metode lain yang juga dipakai ialah *Sampling Based Planning* (SBP), seperti pada algoritma *Rapidly-exploring Random Tree* (RRT). Keunggulan dari algoritma ini ialah memiliki waktu komputasi yang rendah bila dibandingkan dengan algoritma sejenis. Namun, algoritma ini memberikan solusi yang sub-optimal [4]. Terdapat banyak pengembangan variasi-variasi dari algoritma ini, dengan yang terkenal seperti RRT* dan *Informed RRT*. Kebanyakan dari variasi-variasi tersebut bertujuan untuk menjadikan solusi yang dihasilkan algoritma RRT menjadi optimal. Namun, banyak dari variasi-variasi tersebut menambahkan waktu komputasi yang tinggi, sehingga menihilkan kelebihan algoritma RRT. Pada tulisan ini, penulis juga ingin mengoptimasi algoritma RRT dengan cara mengkombinasikannya bersama algoritma Dijkstra.

II. TEORI DASAR

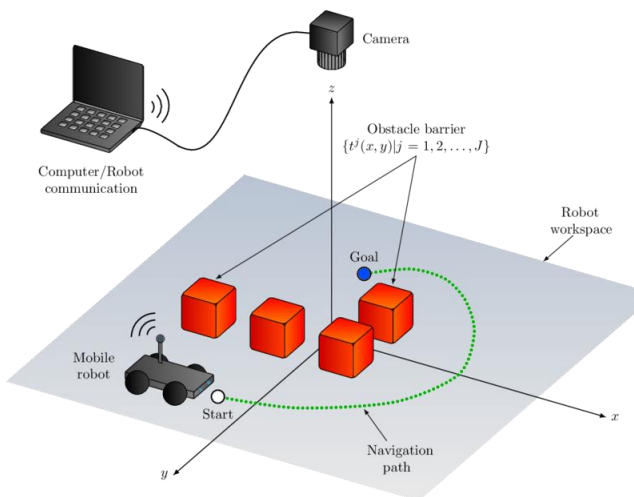
A. Perencanaan Rute (*Path Planning*)

1. Pengertian Perencanaan Rute

Perencanaan rute, disebut juga perencanaan gerak merupakan masalah komputasi untuk menemukan urutan konfigurasi yang valid yang memindahkan objek dari sumber ke tujuan. Rute ini harus dapat dilalui oleh objek dan optimal dalam hal setidaknya satu variabel sehingga dapat dianggap sebagai rute yang cocok. Untuk situasi jarak target yang berbeda, rute paling mulus, rute terpendek, atau rute di mana objek dapat bergerak dengan kecepatan tertinggi dapat menjadi rute yang paling penting. Dengan kata lain, rute optimal ditentukan mengenai karakteristik ini. Metode standar perencanaan rute adalah mendiskritkan ruang dan mempertimbangkan pusat setiap unit sebagai titik pergerakan. Setiap titik pergerakan baik memiliki rintangan yang harus dihindari atau bebas dari rintangan yang bisa dimasuki. Metode diskritisasi yang berbeda menyebabkan rute gerak yang berbeda.

2. Kegunaan Perencanaan Rute

Perencanaan digunakan dalam geometri komputasi, animasi komputer, robotika, dan permainan komputer. Misalnya, pertimbangkan untuk menavigasi robot seluler di dalam gedung ke titik jalan yang jauh. Itu harus menjalankan tugas ini sambil menghindari dinding dan tidak jatuh dari tangga. Algoritma perencanaan gerak akan mengambil deskripsi tugas-tugas ini sebagai masukan, dan menghasilkan perintah kecepatan dan belokan yang dikirim ke roda robot. Algoritma perencanaan gerak mungkin menangani robot dengan jumlah sambungan yang lebih besar (misalnya, manipulator industri), tugas yang lebih kompleks (misalnya manipulasi objek), kendala yang berbeda (misalnya, mobil yang hanya dapat melaju ke depan), dan ketidakpastian (misalnya model yang tidak sempurna dari lingkungan atau robot).



Gambar 1: ilustrasi perencanaan rute pada robot

Dikutip dari: https://www.researchgate.net/figure/Optical-setup-for-obstacle-detection-for-mobile-robot-path-planning_fig1_319277436

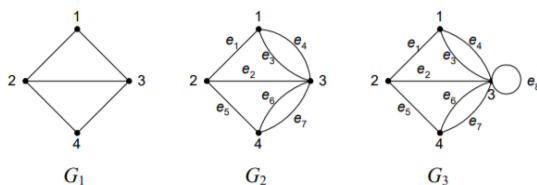
B. Graf

1. Pengertian Graf

Graf merupakan salah satu ilmu bagian dari Matematika Diskrit. Graf merupakan cara untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf (G) didefinisikan sebagai berikut.

$$G = (V, E)$$

Dalam hal ini V merupakan himpunan tidak-kosong dari simpul-simpul (vertices) $\{v_1, v_2, v_3, \dots, v_n\}$ dan E merupakan himpunan sisi (edges) yang menghubungkan sepasang simpul $\{e_1, e_2, e_3, \dots, e_n\}$.



Gambar 2: ilustrasi graf

Dikutip dari:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/matdis.htm>

Gambar di atas merupakan contoh dari graf. Graf G_1 merupakan graf sederhana, sedangkan graf G_2 merupakan graf ganda dan graf G_3 merupakan graf semu. Graf G_1 dapat dinyatakan sebagai graf dengan $V = \{1, 2, 3, 4\}$ dan $E = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 4)\}$. Graf G_2 dapat dinyatakan sebagai graf dengan $V = \{1, 2, 3, 4\}$ dan $E = \{(1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4)\} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$. Sedangkan graf G_3 dapat dinyatakan sebagai graf dengan $V = \{1, 2, 3, 4\}$ dan $E = \{(1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4), (3, 3)\} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$.

2. Jenis-jenis Graf

Graf dapat digolongkan menjadi 2 jenis menurut ada tidaknya gelang atau sisi ganda.

a. Graf sederhana

Graf yang tidak mengandung gelang maupun sisi ganda.

b. Graf tak-sederhana

Graf yang mengandung sisi ganda atau gelang. Jenis graf ini juga dibagi menjadi 2, yaitu graf ganda, graf yang mengandung sisi ganda, dan graf semu, graf yang mengandung sisi gelang.

Graf juga dapat digolongkan menjadi 2 jenis berdasarkan orientasi arah pada sisi.

a. Graf tak-berarah

Graf yang sisinya tidak mempunyai orientasi arah.

b. Graf berarah

Graf yang setiap sisinya diberikan orientasi arah.

3. Terminologi Graf

Terdapat beberapa istilah-istilah terkait graf yang akan dibahas pada tulisan ini.

a. Ketetangaan (Adjacency)

Bila dua simpul terhubung secara langsung maka keduanya bertetangga.

b. Bersisian (Incidence)

Untuk suatu sisi $e = (v_1, v_2)$, maka sisi e bersisian dengan simpul v_1 dan juga bersisian dengan simpul v_2 .

c. Derajat (Degree)

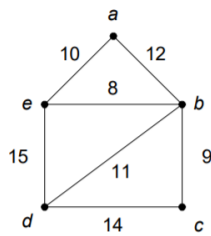
Jumlah sisi yang bersisian dengan simpul tersebut disebut dengan derajat suatu simpul.

d. Keterhubungan (Connected)

Dua buah simpul v_1 dan simpul v_2 disebut terhubung jika terdapat lintasan dari v_1 ke v_2 .

e. Graf berbobot (Weighted Graph)

Graf berbobot ialah graf yang setiap sisinya diberi suatu nilai (bobot).



Gambar 3: ilustrasi graf berbobot

Dikutip dari:

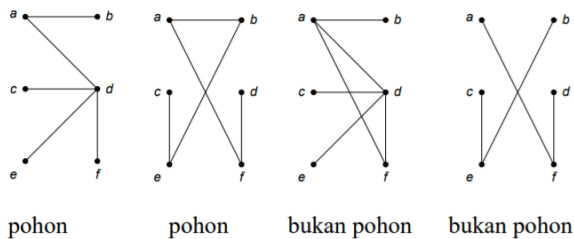
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/matdis.htm>

f. Lintasan (Path)

Suatu lintasan dengan panjang n dari simpul awal v_0 ke simpul tujuan v_n di dalam graf G merupakan barisan simpul dan sisi yang membentuk $v_0, e_1, v_1, e_2, v_2, \dots, v_{n-1}, e_n, v_n$ sedemikian sehingga $e_1 = (v_0, v_1), e_2 = (v_1, v_2), \dots, e_n = (v_{n-1}, v_n)$. Panjang lintasan pada graf tak berbobot adalah jumlah sisi dalam lintasan tersebut. Sedangkan panjang lintasan pada graf berbobot ialah jumlah bobot sisi dalam lintasan tersebut.

C. Pohon

Pohon merupakan bagian khusus dari graf. Pohon merupakan graf yang tidak berarah dan terhubung yang tidak memuat sirkuit sederhana. Diagram pohon dapat digunakan sebagai alat untuk memecahkan masalah dengan menggambarkan semua alternatif pemecahan. Pohon telah digunakan sejak tahun 1857 oleh matematikawan Inggris yang bernama Arthur Cayley untuk menghitung jumlah senyawa kimia.



Gambar 4: ilustrasi pohon

Dikutip dari:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/matdis.htm>

D. Algoritma Rapidly-exploring Random Tree (RRT)

1. Pengertian Algoritma RRT

Pohon acak penjelajahan cepat (RRT) adalah algoritma yang dirancang untuk secara efisien mencari pada ruang berdimensi tinggi yang tidak cembung dengan membangun pohon pengisi ruang secara acak. Pohon dibangun secara bertahap dari sampel yang diambil secara acak dari ruang pencarian dan secara inheren bias tumbuh menuju area masalah yang besar dan belum ditelusuri. Algoritma ini telah banyak digunakan dalam perencanaan gerak robot otonom. RRT dikembangkan oleh Steven M. LaValle dan James J. Kuffner Jr.

RRT dapat dilihat sebagai teknik untuk menghasilkan lintasan loop terbuka untuk sistem nonlinier dengan batasan keadaan. RRT juga dapat dianggap sebagai metode Monte-Carlo untuk membiaskan pencarian. RRT dapat digunakan untuk menghitung perkiraan kebijakan kontrol untuk mengontrol sistem nonlinier dimensi tinggi dengan batasan status dan tindakan.

2. Pseudocode Algoritma RRT

Berikut merupakan pseudocode dari algoritma RRT.

```

Algorithm 1 Rapidly-exploring Random Tree
1: function RAPIDLYEXPLORINGRANDOMTREE(img, start, goal, seed,
   numVertices,  $\Delta q$ )
   ▷ Mengembalikan rute terpendek dari titik awal sampai titik tujuan,
   berdasarkan input data peta, titik awal, titik tujuan, biji pembangkit angka
   acak, banyak simpul dan jarak antar simpul
2:   T.init(start)
3:   while goal not in T do
4:     xRand ← randomState(seed, img)
5:     xNear ← nearestNeighbor(xRand, T)
6:     u ← selectInput(xRand, xNear)
7:     xNew ← newState(xNear, u,  $\Delta q$ , img)
8:     T.addVertex(xNew)
9:     T.addEdge(xNear, xNew, u)
   return T
  
```

3. Variasi Algoritma RRT

Terdapat banyak sekali variasi dari algoritma RRT. Variasi-variasi tersebut umumnya dikembangkan untuk mengatasi permasalahan bahwa solusi algoritma RRT bukanlah solusi yang optimal. Meskipun algoritma RRT jauh lebih cepat daripada algoritma lainnya, seperti A* [5], tetapi solusi yang dihasilkan merupakan rute yang lebih jauh. Maka dari itu, banyak dikembangkan variasi dari algoritma ini. Salah satu varian RRT yang konvergen ke solusi optimal, ialah RRT*. Berikut ini adalah daftar metode berbasis RRT* (dimulai dengan RRT* itu sendiri) [6]. Namun, tidak semua metode turunan itu sendiri konvergen ke solusi optimal.

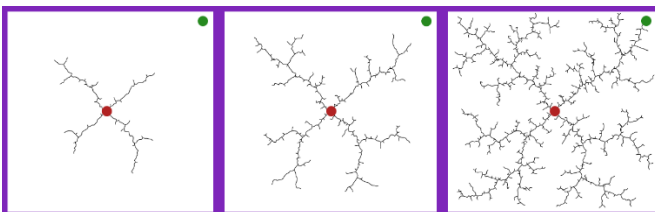
- a. Rapid-exploring random graph (RRG) dan RRT*, varian dari RRT yang konvergen menuju solusi optimal
- b. RRT*-Smart, sebuah metode untuk mempercepat laju konvergensi RRT* dengan menggunakan optimasi rute (dengan cara yang mirip dengan Theta*) dan pengambilan sampel cerdas (dengan membiaskan pengambilan sampel terhadap simpul rute, yang – setelah optimasi rute – kemungkinan besar untuk menjadi dekat dengan rintangan)
- c. RRT*FN, RRT* dengan jumlah simpul yang tetap, yang secara acak menghapus simpul daun di pohon dalam setiap iterasi
- d. RRT*-AR, perencanaan rute alternatif berbasis pengambilan sampel
- e. Informed RRT*, meningkatkan kecepatan konvergensi RRT* dengan memperkenalkan heuristik, mirip dengan cara A* meningkatkan algoritma Dijkstra
- f. RRT* Real-Time* (RT-RRT*), varian RRT* dan Informed RRT* yang menggunakan strategi pengkabelan ulang pohon online yang

memungkinkan akar pohon bergerak dengan agen tanpa membuang rute sampel sebelumnya, untuk dapatkan perencanaan rute waktu nyata dalam lingkungan yang dinamis seperti permainan komputer

- g. RRTX dan RRT#, optimalisasi RRT* untuk lingkungan yang dinamis
- h. Theta*-RRT, metode perencanaan gerak dua fase yang mirip dengan A*-RRT* yang menggunakan kombinasi hierarki pencarian sudut apa pun dengan perencanaan gerak RRT untuk pembangkitan lintasan cepat di lingkungan dengan kendala nonholonomik yang kompleks
- i. RRT* FND, perpanjangan RRT* untuk lingkungan dinamis
- j. RRT-GPU, implementasi RRT tiga dimensi yang memanfaatkan akselerasi perangkat keras
- k. APF-RRT, kombinasi perencana RRT dengan metode Bidang Potensi Buatan yang menyederhanakan tugas perencanaan ulang
- l. CERRT, ketidakpastian pemodelan perencana RRT, yang dikurangi dengan mengeksploitasi kontak
- m. RRT-Blossom, perencana RRT untuk lingkungan yang sangat terbatas.
- n. TB-RRT, algoritma RRT berbasis waktu untuk perencanaan pertemuan dua sistem dinamis.
- o. RRdT*, perencana berbasis RRT* yang menggunakan beberapa pohon lokal untuk secara aktif menyeimbangkan eksplorasi dan eksploitasi ruang dengan melakukan pengambilan sampel lokal.

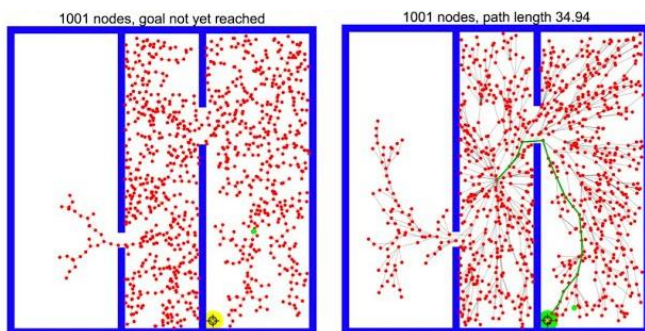
4. Ilustrasi Algoritma RRT

Berikut merupakan ilustrasi dari solusi yang dihasilkan algoritma RRT beserta variasinya.



Gambar 5: ilustrasi cara algoritma RRT menemukan rute dari titik asal sampai titik tujuan

Dikutip dari: <https://towardsdatascience.com/how-does-a-robot-plan-a-path-in-its-environment-b8e9519c738b>

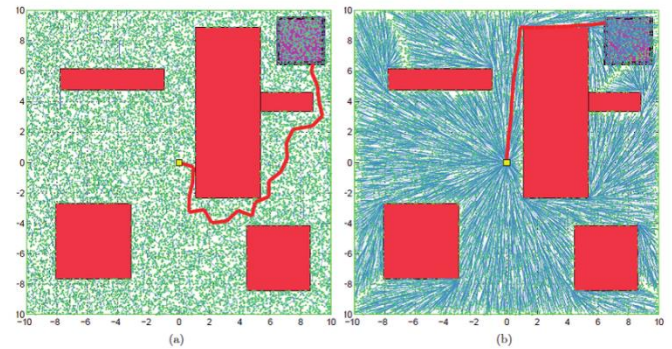


Gambar 6: ilustrasi pembangkitan titik acak dan solusi

algoritma RRT

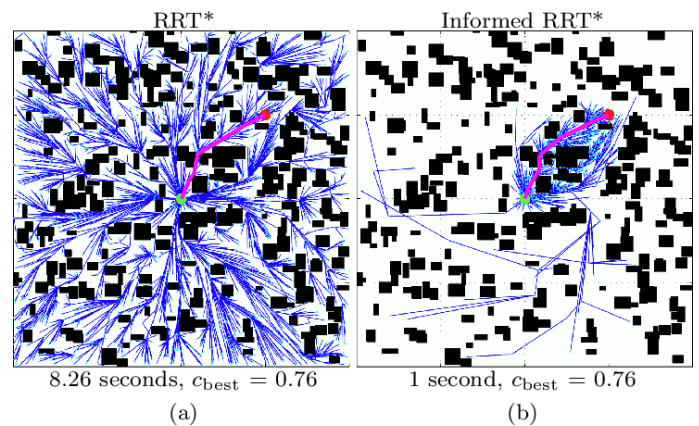
Dikutip dari:

<https://i.ytimg.com/vi/Ob3BIJkQJEw/sddefault.jpg>



Gambar 7: ilustrasi solusi yang dihasilkan algoritma RRT dengan algoritma RRT*

Dikutip dari: https://www.researchgate.net/figure/A-Comparison-of-the-RRT-a-and-RRT-b-algorithms-on-an-example-Karaman-and-Frazzoli_fig2_289366294



Gambar 8: ilustrasi solusi yang dihasilkan algoritma RRT* dengan algoritma Informed RRT*

Dikutip dari: https://www.researchgate.net/figure/Solutions-of-equivalent-cost-found-by-RRT-and-Informed-RRT-on-a-random-world-After-an_fig1_261512737

E. Algoritma Dijkstra

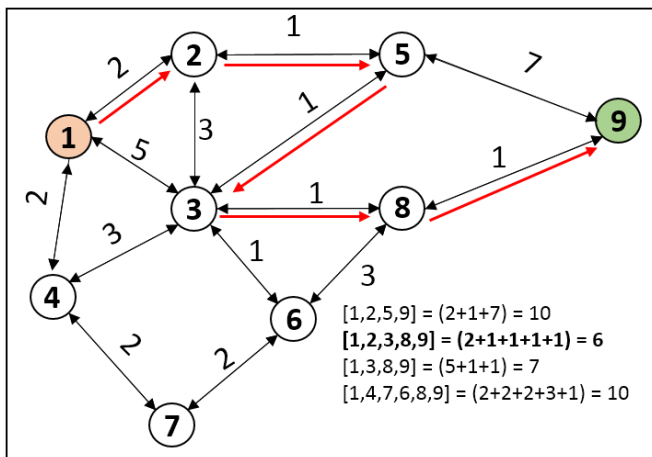
1. Pengertian Algoritma Dijkstra

Algoritma Dijkstra merupakan algoritma untuk menemukan rute terpendek antara simpul-simpul dalam graf, yang dapat mewakili, misalnya, jaringan jalan. Algoritma ini disusun oleh ilmuwan komputer Edsger W. Dijkstra pada tahun 1956 dan diterbitkan tiga tahun kemudian. Algoritma ini memiliki banyak varian. Algoritma asli Dijkstra menemukan rute terpendek antara dua simpul yang diberikan, [7] tetapi varian yang lebih umum memperbaiki satu node sebagai simpul awal dan menemukan rute terpendek dari simpul awal ke semua node lain dalam grafik, menghasilkan rute terpendek pohon.

Untuk simpul awal yang diberikan dalam graf, algoritma menemukan rute terpendek antara node itu dan setiap node lainnya. Ini juga dapat digunakan untuk menemukan rute terpendek dari satu simpul ke satu simpul tujuan dengan menghentikan algoritma setelah rute terpendek ke simpul

tujuan telah ditentukan. Misalnya, jika simpul graf mewakili kota dan biaya rute tepi mewakili jarak mengemudi antara pasangan kota yang dihubungkan oleh jalan langsung (untuk menyederhanakan, mengabaikan lampu merah, rambu berhenti, jalan tol, dan penghalang lainnya), algoritma Dijkstra dapat digunakan untuk menemukan rute terpendek antara satu kota dan semua kota lainnya.

Algoritma Dijkstra menggunakan struktur data untuk menyimpan dan mendapatkan solusi parsial yang diurutkan berdasarkan jarak dari awal. Algoritma ini secara asimtotik merupakan algoritma rute terpendek sumber tunggal tercepat yang diketahui untuk graf berarah arbitrer dengan bobot non-negatif tak terbatas. Namun, kasus khusus (seperti bobot terbatas/bilangan bulat, grafik asiklik terarah, dll.) memang dapat ditingkatkan lebih lanjut.



Gambar 9: ilustrasi solusi yang dihasilkan algoritma Dijkstra
Dikutip dari:

https://www.researchgate.net/figure/Illustration-of-Dijkstras-algorithm_fig1_331484960

2. Pseudocode Algoritma Dijkstra

Berikut merupakan pseudocode dari algoritma Dijkstra.

Algorithm 1 Rapidly-exploring Random Tree

```

1: function DIJKSTRA(graph, start)
  > Mengembalikan rute terpendek dan kumpulan pasangan suatu simpul
  dengan simpul sebelumnya berdasarkan input graf dan titik awal

2: createVertexSet(Q)
3: i ← 1
4: while i ≤ graph.vertexNum do
5:   dist[graph.vertex[i]] ← INFINITY
6:   prev[graph.vertex[i]] ← UNDEFINED
7:   Q.add(graph.vertex[i])
8:   i ← i + 1
9: while not isEmpty(Q) do
10:  u ← getVertexWithMinDistance(Q)
11:  Q.remove(u)
12:  i ← 1
13:  while i ≤ neighborOf(u).length do
14:    v ← neighborOf(u)[i]
15:    alt ← dist[u] + length(u, v)
16:    if then alt < dist[v]
17:      dist[v] ← alt
18:      prev[v] ← u
19:    i ← i + 1
  return dist, prev

```

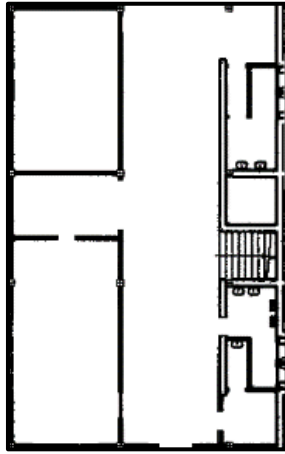
III. ANALISIS DAN PEMBAHASAN

A. Pemecahan Masalah

Hasil algoritma RRT ialah kumpulan simpul-simpul yang mengarahkan dari titik awal ke titik tujuan. Namun, seperti yang telah dijelaskan sebelumnya, solusi yang dihasilkan merupakan solusi sub-optimal dan seringkali banyak simpul yang dapat dilompati untuk mencapai solusi optimal. Pendekatan yang dipilih penulis ialah menjadikan pohon hasil algoritma ini menjadi suatu graf berbobot. Graf ini disusun oleh simpul-simpul dari hasil algoritma RRT dengan ditambahkan sisi-sisi baru jika sisi tersebut tidak dihalangi oleh dinding atau penghalang. Selain penambahan sisi-sisi baru, semua sisi-sisi yang ada juga perlu dihitung bobot yang berasal dari jarak masing-masing simpulnya. Lalu, graf berbobot yang dihasilkan akan dipilih simpul-simpul yang paling efisien dari titik awal sampai titik akhir menggunakan algoritma Dijkstra.

B. Pengumpulan Data

Pada makalah ini, penulis membuat program yang dapat mengambil dan memproses file gambar. Program disediakan gambar bawaan, tetapi pengguna dapat memilih gambar sendiri dengan argumen pada terminal. Gambar bawaan merupakan denah suatu rumah yang menggambarkan ruang terbuka dengan dilengkapi tembok, penghalang berliku-liku, dan ruangan tertutup tanpa akses masuk dan keluar untuk menggambarkan situasi asli. Lalu, program menyediakan antarmuka untuk pengguna memilih titik awal dan titik tujuan. Fitur tersebut digunakan penulis untuk tes manual. Namun, untuk kebutuhan pengumpulan data, penulis mengotomatisasikan pemilihan titik awal dan titik tujuan menggunakan angka acak yang telah dibatasi sesuai ukuran gambar. Program lalu menjalankan fungsi RRT yang mengembalikan kumpulan simpul-simpul jalan. Simpul-simpul tersebut lalu diolah untuk membuat suatu graf seperti yang telah dijelaskan sebelumnya. Lalu graf tersebut diproses dengan algoritma Dijkstra untuk mendapatkan kumpulan simpul-simpul terpendek dari titik awal ke titik tujuan. Untuk mengukur performansi dari program, penulis juga menghitung waktu yang dibutuhkan untuk menjalankan fungsionalitas algoritma RRT dan fungsionalitas optimalisasi berupa pembuatan graf dan algoritma Dijkstra. Lalu program menambahkan data-data yang didapatkan berupa jarak rute hasil algoritma RRT, jarak rute hasil optimalisasi, waktu fungsionalitas algoritma RRT, dan waktu fungsionalitas optimalisasi ke berkas csv. Program juga menyimpan gambar hasil rute algoritma RRT dan hasil optimalisasi dalam format png.



Gambar 10: peta yang digunakan dalam percobaan
Diambil dari: <https://github.com/ArianJM/rapidly-exploring-random-trees>

Berikut merupakan pseudocode dari keseluruhan program yang penulis buat.

```

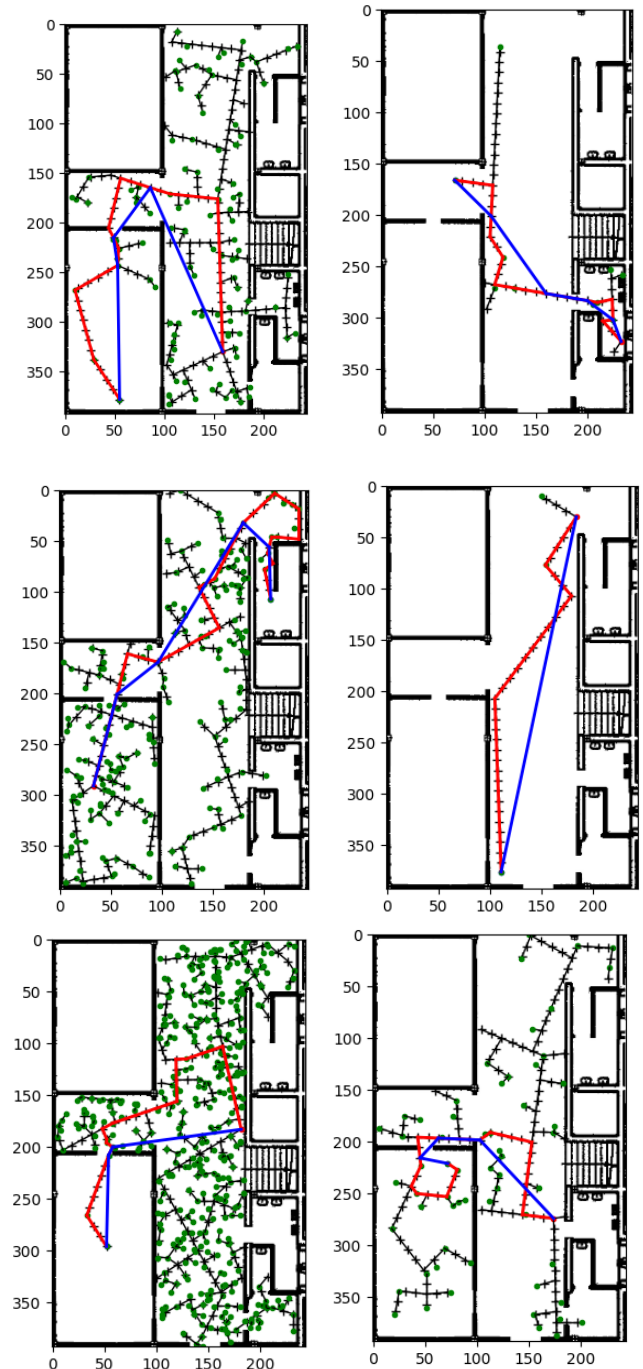
Algorithm 1 Main Program
1: procedure MAIN
  ▷ Initial State: imgFile, seed, dan csvFile terdefinisi
  ▷ Final State: Dituliskan ke dalam csvFile data berupa jarak rute hasil algoritma RRT, jarak rute hasil optimalisasi, waktu fungsionalitas algoritma RRT, dan waktu fungsionalitas optimalisasi
2:   t0 ← time.now
3:   img ← readImage(imgFile)
4:   start, goal ← selectStartGoalPoints(img)
5:   graphRRT ← rapidlyExploringRandomTree(img, start, goal, seed, numVertices, Δq)
6:   path ← extractPath(graphRRT)
7:   lengPath ← totalDistance(path)
8:   t1 ← time.now
9:   initGraph ← {}
10:  i ← 1
11:  while i ≤ path.length do
12:    initGraph[i] ← {}
13:    i ← i + 1
14:  i ← 1
15:  while i ≤ path.length do
16:    initGraph[i][i+1] ← distance(path[i], path[i+1])
17:    j ← i + 2
18:    while j ≤ path.length do
19:      if notHittingWall(path[i], path[j]) then
20:        initGraph[i][j] ← distance(path[i], path[j])
21:        j ← j + 1
22:    i ← i + 1
23:  graph ← Graph(initGraph)
24:  shortestPath, prevNodes ← dijkstraAlgorithm(graph, start)
25:  resPathDijkstra ← findDijkstraResult(prevNodes, shortestPath, start, goal)
26:  lengOptimized ← totalDistance(resPathDijkstra)
27:  t3 ← time.now
28:  plot.draw(img, path, shortestPath)
29:  fileName ← secrets.random
30:  plot.save(fileName)
31:  write(csvFile, lengPath, lengOptimized (t1-t0), (t3-t2))
  
```

Penulis mengimplementasikan program ini menggunakan bahasa Python. Penulis juga menggunakan berbagai *library*, antara lain:

1. openCV untuk pemrosesan gambar.
2. time untuk pengukuran performansi.
3. math untuk membantu perhitungan.
4. secrets untuk pemilihan nama berkas hasil yang aman.
5. random untuk pembangkitan angka acak.
6. sys dan os untuk validasi dan pengambilan berkas gambar.
7. matplotlib untuk menggambar ilustrasi rute yang dihasilkan baik dari algoritma RRT maupun hasil

optimalisasi, serta menyimpan gambar tersebut menjadi suatu berkas.

Program ini dijalankan secara otomatis secara berkali-kali sehingga didapatkan 249 data nilai-nilai yang akan diolah serta gambar-gambar hasil perencanaan rute menggunakan algoritma RRT dan hasil rute optimalasinya. Berikut beberapa gambar perbandingannya.



Gambar 11-16: Beberapa contoh solusi yang dihasilkan oleh algoritma RRT dan hasil optimalisasinya.

Keterangan:

1. Titik-titik hijau merupakan simpul-simpul yang dibangkitkan oleh algoritma RRT.
2. Garis hitam merupakan sisi-sisi dari pohon yang dibuat

oleh algoritma RRT.

3. Garis merah merupakan rute jalan yang dihasilkan oleh algoritma RRT.
4. Garis biru merupakan rute jalan yang dihasilkan oleh algoritma RRT yang ditambahkan fungsionalitas optimasi.

C. Pengolahan dan Analisis Data

Data yang didapatkan ialah data panjang rute algoritma RRT, panjang rute hasil optimalisasi, waktu algoritma RRT, dan waktu fungsionalitas optimalisasi jarak. Oleh karena itu, dibutuhkan kolom data baru berupa waktu total, gabungan antara waktu algoritma RRT ditambah waktu fungsionalitas optimalisasi, serta rasio antara panjang teroptimisasi dengan panjang algoritma RRT, dan rasio antara waktu total dengan waktu algoritma RRT. Kedua data rasio tersebut akan digunakan untuk perbandingan pengaruh optimasi terhadap hasil jarak rute dan performansi program.

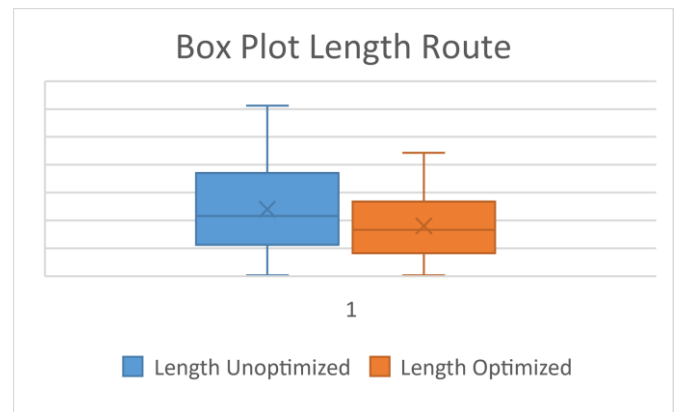
Penulis menggunakan *library* pandas dari bahasa pemrograman Python untuk melakukan pengolahan dan analisis data dan aplikasi Excel untuk visualisasi data. Didapatkan hasil sebagai berikut.

	Length Unoptimized (unit)	Length Optimized (unit)	Time RRT (ms)	Time Total (ms)
mean	240.5158	180.3707	303.9054	431.5261
std	148.4762	110.3365	427.8731	517.5234
min	3.605551	3.605551	56.1835	57.2233
25%	115.1577	84.11896	96.699	118.9248
50%	215.8455	167.3708	161.2979	269.0357
75%	370.091	266.9209	335.5978	568.7476
max	612.2341	442.9939	4493.154	4985.384

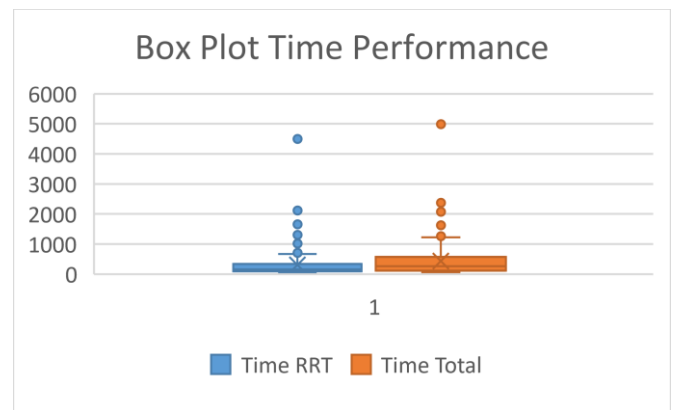
Tabel 1: Statistik deskriptif dari data jarak rute tak teroptimasi, jarak rute teroptimasi, waktu komputasi algoritma RRT, dan waktu komputasi algoritma keseluruhan

	Ratio Length	Ratio Time
mean	0.767436	1.520543
std	0.130232	0.667449
min	0.264604	1.010388
25%	0.68834	1.108499
50%	0.774344	1.316318
75%	0.855411	1.630328
max	1	5.380422

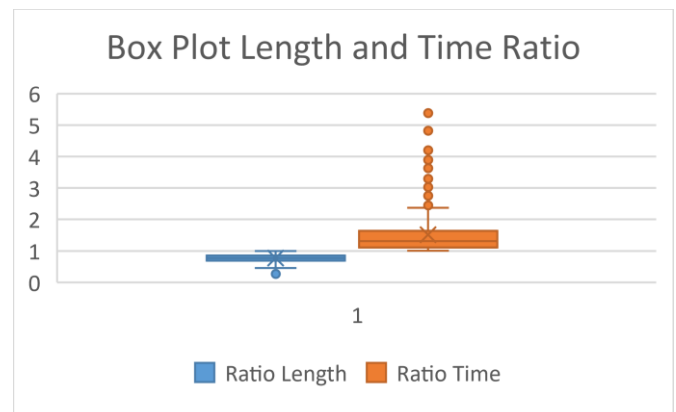
Tabel 2: Statistik deskriptif dari data rasio jarak rute teroptimasi dibandingkan jarak rute tak teroptimasi dan rasio waktu Komputasi algoritma RRT dibandingkan dengan waktu Komputasi algoritma keseluruhan.



Grafik 1: *box plot* dari data jarak rute tak teroptimasi dan jarak rute teroptimasi



Grafik 2: *box plot* dari data waktu komputasi algoritma RRT dan waktu komputasi algoritma keseluruhan



Grafik 3: *box plot* dari data rasio jarak rute teroptimasi dibandingkan jarak rute tak teroptimasi dan rasio waktu Komputasi algoritma RRT dibandingkan dengan waktu Komputasi algoritma keseluruhan.

Dari hasil tersebut, dapat dilihat bahwa penambahan fungsionalitas optimalisasi berhasil menurunkan jarak rute. Jarak rute yang dihasilkan rata-rata menjadi sekitar 76.74% jarak hasil awal algoritma RRT. Penurunannya juga cukup bervariasi mulai dari rasio 1, jarak tidak berubah sama sekali sampai rasio 0.264604, jarak turun menjadi 26.65% jarak semula. Penurunan tersebut merupakan penurunan yang sangat drastis, bahkan terdapat beberapa pencilan bawah seperti terlihat pada Grafik 3. Rasio tetap 1 terjadi ketika solusi yang dihasilkan

algoritma RRT merupakan solusi yang sudah optimal, biasa terjadi ketika jarak titik asal dan titik tujuan pendek, seperti terlihat pada Tabel 1 baris "min". Walaupun bervariasi, rasio jarak rute cukup stabil terlihat dari nilai standard deviasinya bernilai 0.130232, yaitu cukup kecil.

Dari hasil analisis data tersebut juga, dapat disimpulkan bahwa optimalisasi hasil dari algoritma RRT ini mengurangi performansi waktu seperti yang terlihat pada Grafik 2. Pada Tabel 2, didapatkan rata-rata kenaikan waktu yang diperlukan adalah sekitar 52.05%. Berbeda dengan data jarak rute, standard deviasi dari data waktu cukup tinggi. Hal tersebut juga terlihat pada Grafik 2, banyak terdapat pencilan pada data waktu. Hal tersebut menyebabkan data rasio waktu juga memiliki banyak data pencilan. Rasio waktu terbesar cukup berbeda dengan rata-rata, waktu yang diperlukan menjadi 5.38 kali waktu biasa. Walaupun begitu, secara keseluruhan, kenaikan data ini masih sangat dapat ditolerir karena kompleksitas waktu dari algoritma keseluruhan tidak naik secara drastis.

IV. KESIMPULAN

Berdasarkan data yang telah dikumpulkan dan dianalisis, disimpulkan bahwa penambahan fungsionalitas optimasi jarak rute pada algoritma RRT cukup berhasil. Hal ini ditunjukkan dengan rata-rata jarak rute menjadi 76.74% dari jarak rute sebelumnya, sedangkan waktu program rata-rata naik sebesar 52.05%. Hal ini menunjukkan bahwa optimalisasi ini dapat mengurangi jarak rute tanpa menambah waktu program secara drastis. Optimalisasi ini cukup berguna untuk digunakan pada perencanaan rute yang memerlukan solusi cukup optimal dan waktu program yang tetap rendah. Optimalisasi ini juga membantu dalam pengambilan rute yang lebih alami dengan memangkas titik-titik belok yang tidak perlu sebagaimana yang terlihat pada Gambar 11-16 dengan belokan dan titik-titik yang perlu dituju jauh lebih sedikit.

V. UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena dengan rahmat dan berkah-Nya, makalah ini dapat diselesaikan tepat waktu. Penulis juga mengucapkan terima kasih kepada kedua orangtua, serta teman-teman yang telah memberikan dukungan selama pengerjaan makalah ini. Tidak lupa rasa hormat dan terima kasih kepada dosen-dosen pengampu mata kuliah IF2120 Matematika Diskrit yaitu, Pak Rinaldi, Ibu Ulfa, terutama Ibu Harlili selaku dosen pengampu pada kelas penulis, yang telah memberi materi kepada penulis pada semester ketiga Tahun Ajar 2020/2021. Penulis meminta maaf jika terdapat kesalahan kata-kata dalam makalah ini, penulis berharap makalah ini dapat digunakan sebaik-baiknya.

REFERENSI

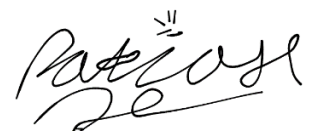
- [1] Nugraha, Eka Setia. "Simulator Edukatif untuk Pembelajaran Algoritma Rapidly-exploring Random Tree (RRT) Educational Simulator for Teaching of Rapidly-exploring Random Tree (RRT) Algorithms."
- [2] Garcia, MA Porta, et al. "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation." *Applied Soft Computing* 9.3 (2009): 1102-1110. H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

- [3] Belkhouche, Fethi, and B. Bendjilali. "Reactive path planning for 3-D autonomous vehicles." *IEEE transactions on control systems technology* 20.1 (2011): 249-256.
- [4] Karaman, Sertac, and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning." *The international journal of robotics research* 30.7 (2011): 846-894.
- [5] Lajevardy, Pooria & mousavian, Azam & Oskoei, Mohammadreza. "A Comparison Between RRT* and A* Algorithms for Motion Planning in Complex Environments". (2015).
- [6] https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree. Diakses pada 14 Desember 2021.
- [7] Dijkstra, Edsger Wybe. "A note on two problems in connexion with graphs:(Numerische Mathematik, 1 (1959), p 269-271)." (1959).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 14 Desember 2021



Malik Akbar Hashemi Rafsanjani / 13520105