BAB II
TINJAUAN PUSTAKA

Penelitian ini dimulai dengan mengumpulkan studi literatur terkait yang akan membantu dalam merancang pengembangan algoritma perencanaan jalur dengan menggunakan kombinasi algoritma RRT dan algoritma PSO.

2.1 Algoritma Perencanaan Jalur

Algoritma perencanaan jalur atau *path* planning merupakan suatu kegiatan proses untuk menemukan gerakan bebas tumbukan dimulai dari titik awal sampai ke titik akhir pada lingkungan tertentu. Masalah perencanaan jalur didefinisikan sebagai masalah menemukan jalur dalam ruang konfigurasi dari posisi awal ke posisi tujuan sambil mematuhi seperangkat aturan [1]. Perencanaan jalur adalah salah satu masalah mendasar dalam robotika dan merupakan masalah yang dipelajari secara luas. Perencanaan jalur memiliki aplikasi di berbagai bidang, seperti mobil self-driving, aplikasi medis, grafik gerak, transportasi sel, dan bedah robot [2]-[6]. Algoritma *path planning* dapat dibagi menjadi beberapa kelas utama, seperti metode variabel, metode pencarian grafik, dan metode pencarian inkremental [1], [7].

Metode pencarian grafik seperti Djikstra dan A* memiliki sifat *resolution-complete* dan *resolution-optimal* [8]. Artinya, kelengkapan dan optimalitas algoritma hanya bisa dijamin jika resolusi diskritisasi dari ruang konfigurasi cukup baik. Namun, resolusi yang baik mungkin sulit dicapai dalam ruang berdimensi tinggi [9]. Metode pencarian ikremental, diantaranya adalah algoritma SBP dapat menghindari kebutuhan akan diskritisasi ruang konfigurasi. Ini membuat metode SBP lebih efektif terhadap berbagai ukuran ruang konfigurasi [10]. Contoh metode

SBP adalah RRT [11] dan RRT* [12]. Keuntungan dari algoritma RRT adalah dapat memberikan solusi yang cepat dalam sistem multi-dimensi [13]. Tetapi memiliki kelemahan yaitu memberikan solusi sub-optimal [14]. Algoritma RRT kemudian dikembangkan menjadi RRT* [15], yang memberikan solusi optimal [16]. Namun algoritma tersebut memiliki kelemahan berupa kecepatan konvergensi yang lambat [17] - [19].

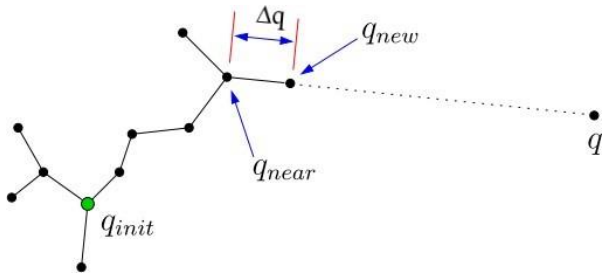
Algoritma PSO [4], ACS [23-25], dan ACO [26-27] merupakan bagian dari algoritma SI dimana algoritma ini dapat digunakan untuk menghasilkan jalur optimal dari informasi jalur sub-optimal sebelumnya. Sepengetahuan penulis, belum ada penelitian sebelumnya yang mencoba untuk meningkatkan kualitas algoritma SBP menggunakan hibridasi algoritma RRT yang merupakan bagian dari SBP dan algoritma PSO yang merupakan bagian dari SI.

2.2 Algortima *Rapidly-Exploring Random Tree* (RRT)

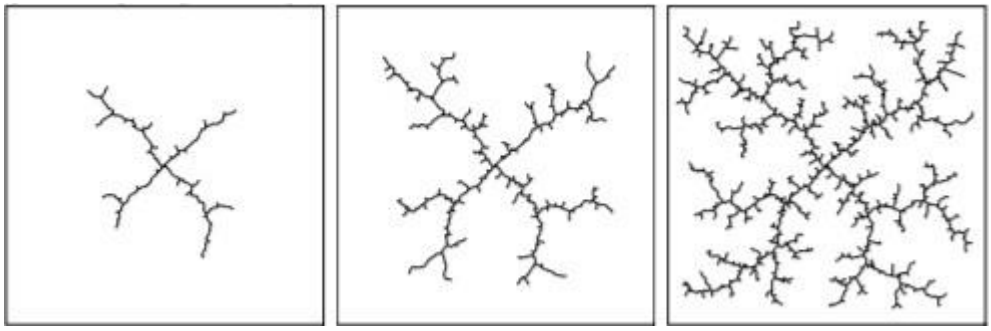
Algoritma RRT adalah salah satu metode dari SBP dimana algoritma RRT merupakan sebuah algoritma perencanaan jalur yang sering digunakan pada berbagai macam masalah perencanaan jalur. Algoritma RRT kemudian dikembangkan menjadi *informed* RRT* oleh J. D Gammell [10]. Algoritma *informed* RRT* sendiri merupakan perkembangan dari algortima RRT* yang dikembangkan oleh S.Karaman [12]. Algoritma RRT ini dikembangkan oleh S. M. Lavalle dan James pada tahun 1998 [36].

Algoritma RRT merupakan sebuah struktur data yang dirancang untuk mencari ruang bebas tabrakan secara efisien. RRT membuat cabang pada pohon pencarian, di mana algoritma ini mengembangkan setiap cabang yang akan diturunkan dari konfigurasi asli menggunakan sampel acak di ruang pencarian.

hubungan antara cabang dengan sampel acak melintasi ruang kosong dan tidak memiliki penghalang, cabang tersebut dapat ditambahkan ke cabang pohon baru. Cabang dari pohon pencarian algoritma RRT akan terus tumbuh ke segala arah tanpa hambatan tabrakan sampai menemukan tujuannya. Jalur yang ditempuh oleh cabang pertama dari pohon tersebut merupakan jalur terpendek menuju tujuan. Pohon pencarian dari algoritma RRT lebih cenderung mengeksplorasi daerah-daerah penyusun yang belum pernah dijelajahi. Pohon pencarian RRT juga membutuhkan waktu komputasi yang lebih sedikit karena algoritma RRT tidak mempertimbangkan jalur terpendek ke node target. Adapun cabang dari pohon pencarian dan ilustrasi jalur pohon pencarian pada setiap iterasi berkembang untuk mencapai tujuan dari algoritma RRT bisa dilihat pada Gambar 2.1 dan Gambar 2.2 dibawah ini.



Gambar 2. 1 Ilustrasi pohon pencarian RRT [37]



Gambar 2. 2 Ilustrasi jalur pohon pencarian algoritma RRT setiap iterasi berkembang [37]

Algoritma RRT dimulai dengan penginisialisasian *node start* pada daerah C_{free} atau daerah ruang konfigurasi bebas dimana setiap iterasi terdapat *node random* (q_{rand}) yang akan berpindah-pindah pada daerah ruang konfigurasi bebas yang mana setiap perpindahan q_{rand} akan dilakukan pengecekan *collison* di sekitar q_{rand} dimana jika terdapat halangan atau *obstacle* pada q_{rand} tersebut maka akan dipilih *node* terdekat dari pohon pencarian atau disebut degan $q_{nearest}$ atau *node* terdekat yang kemudian membuat cabang baru dari q_{rand} menuju $q_{nearest}$ sejauh *expand distance* yang mana cabang baru tersebut diberi nama *new node* (q_{new}). Algoritma ini akan terus berlanjut ke iterasi selanjutnya hingga menemukan atau sampai di titik tujuan atau *goal node*. Adapun pseudocode dasar dari algoritma RRT bisa dilihat pada Gambar 2.3 berikut ini.

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

10. $T \leftarrow InitializeTree()$
11. $T \leftarrow InsertNode(\emptyset, q_{init}, T)$
12. **for** $k \leftarrow 1$ **to** N **do**
13. $\quad q_{rand} \leftarrow RandomSample(k)$
14. $\quad q_{nearest} \leftarrow NearestNeighbor(q_{rand}, Q_{near}, T)$
15. $\quad q_{new} \leftarrow Steer\ (q_{nearest}, q_{rand}, \Delta q)$
16. $\quad$ **if** $Obstaclefree(q_{new}, q_{nearest})$ **then**
17. $\quad\quad T \leftarrow InsertNode(q_{min}, q_{new}, T)$
18. **end**

Gambar 2. 3 Pseudocode dasar algoritma RRT [37]

Algoritma RRT kemudian dikembangkan menjadi algortima RRT* yang dikembangkan oleh S. Karaman [12] memiliki perbedaan mendasar dengan algoritma RRT dikarenakan algoritma RRT* mempertimbangkan semua node yang paling dekat dengan q_{new} dan selalu memeriksa nilai terkecil pada setiap q_{near}. Algoritma RRT* dikembangkan dengan menambahan 2 proses yaitu *choose parent* dan *rewire* dimana 2 proses ini bertujuan untuk mencapai jalur terpendek dengan

waktu komputasi dalam mencapai tujuan yang cepat. Adapun *pseudocode* dari algoritma RRT* dengan 2 proses tambahan yaitu proses *choose parent* dan *rewire* bisa dilihat pada gambar berikuti ini.

Algorithm 2 : $T = (V, E) \leftarrow RRT^*(q_{init})$

13. $T \leftarrow InitializeTree()$
14. $T \leftarrow InsertNode(\emptyset, q_{init}, T)$
15. **for** $k \leftarrow 1$ **to** N **do**
16. $q_{rand} \leftarrow RandomSample(k)$
17. $q_{nearest} \leftarrow NearestNeighbor(q_{rand}, Q_{near}, T)$
18. $q_{new} \leftarrow Steer\ (q_{nearest}, q_{rand}, \Delta q)$
19. **if** $Obstaclefree(q_{new}, q_{nearest})$ **then**
20. $Q_{near} \leftarrow Near(T, z_{new})$
21. $q_{parent} \leftarrow ChooseParent(q_{new}, Q_{near}, q_{nearest})$
22. $T \leftarrow InsertNode(q_{parent}, q_{new}, T)$
23. $T \leftarrow Rewire(T, Q_{near}, q_{parent}, q_{new})$
24. **end**

Gambar 2. 4 Pseudocode dasar algoritma RRT* [37]

Algorithm 3 :
$q_{min} \leftarrow ChooseParent(q_{rand}, Q_{near}, q_{nearest}, \Delta q)$

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

Gambar 2. 5 Pseudocode dari proses choose parent [37]

Algorithm 4 : $T \leftarrow Rewire(T, Q_{near}, q_{min}, q_{rand})$

7. **for** $q_{near} \in Q_{near}$ **do**
8. $q_{path} \leftarrow Steer(q_{near}, q_{rand}, \Delta q)$
9. **if** $ObstacleFree(q_{path})$ **and** $Cost(q_{rand}) + c(q_{path}) < Cost(q_{near})$ **then**
10. $T \leftarrow ReConnect(q_{rand}, q_{near}, T)$
11. **end**
12. **return** T

Gambar 2. 6 Pseudocode dari proses choose rewire [37]

Algorithm 1: Informed RRT*$(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$

$V \leftarrow \{\mathbf{x}_{\text{start}}\}$;
$E \leftarrow \emptyset$;
$X_{\text{soln}} \leftarrow \emptyset$;
$\mathcal{T} = (V, E)$;
for iteration $= 1 \dots N$ **do**
  $c_{\text{best}} \leftarrow \min_{\mathbf{x}_{\text{soln}} \in X_{\text{soln}}} \{\texttt{Cost}(\mathbf{x}_{\text{soln}})\}$;
  $\mathbf{x}_{\text{rand}} \leftarrow \texttt{Sample}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}, c_{\text{best}})$;
  $\mathbf{x}_{\text{nearest}} \leftarrow \texttt{Nearest}(\mathcal{T}, \mathbf{x}_{\text{rand}})$;
  $\mathbf{x}_{\text{new}} \leftarrow \texttt{Steer}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{rand}})$;
  if $\texttt{CollisionFree}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$ **then**
    $V \leftarrow \cup \{\mathbf{x}_{\text{new}}\}$;
    $X_{\text{near}} \leftarrow \texttt{Near}(\mathcal{T}, \mathbf{x}_{\text{new}}, r_{\text{RRT*}})$;
    $\mathbf{x}_{\min} \leftarrow \mathbf{x}_{\text{nearest}}$;
    $c_{\min} \leftarrow \texttt{Cost}(\mathbf{x}_{\min}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$;
    for $\forall \mathbf{x}_{\text{near}} \in X_{\text{near}}$ **do**
      $c_{\text{new}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{near}}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})$;
      if $c_{\text{new}} < c_{\min}$ **then**
        if $\texttt{CollisionFree}(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})$ **then**
          $\mathbf{x}_{\min} \leftarrow \mathbf{x}_{\text{near}}$;
          $c_{\min} \leftarrow c_{\text{new}}$;
    $E \leftarrow E \cup \{(\mathbf{x}_{\min}, \mathbf{x}_{\text{new}})\}$;
    for $\forall \mathbf{x}_{\text{near}} \in X_{\text{near}}$ **do**
      $c_{\text{near}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{near}})$;
      $c_{\text{new}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{new}}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})$;
      if $c_{\text{new}} < c_{\text{near}}$ **then**
        if $\texttt{CollisionFree}(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})$ **then**
          $\mathbf{x}_{\text{parent}} \leftarrow \texttt{Parent}(\mathbf{x}_{\text{near}})$;
          $E \leftarrow E \setminus \{(\mathbf{x}_{\text{parent}}, \mathbf{x}_{\text{near}})\}$;
          $E \leftarrow E \cup \{(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})\}$;
    if $\texttt{InGoalRegion}(\mathbf{x}_{\text{new}})$ **then**
      $X_{\text{soln}} \leftarrow X_{\text{soln}} \cup \{\mathbf{x}_{\text{new}}\}$;
return $\mathcal{T}$;

Gambar 2. 7 Pseudocode algoritma informed RRT* [10]

Algoritma *informed* RRT* merupakan pengembangan lebih lanjut dari algoritma RRT*. Algoritma *informed* RRT* bekerja dengan prinsip yang sama dengan algoritma RRT*, tetapi ada beberapa perbedaan dikarenakan algoritma *informed* RRT* mempersempit area pencarian dengan berfokus pada target yang membentuk wilayah elips dan mencari solusi yang ditemukan. Sepengetahuan penulis, belum ada penelitian sebelumnya yang mencoba untuk meningkatkan kualitas algoritma perencanaan jalur menggunakan hibridasi algoritma RRT dan algoritma PSO.

2.3 Algoritma *Particle Swarm Optimization* (PSO)

Algoritma PSO [4], ACS [23-25], dan ACO [26], [27] merupakan bagian dari algoritma SI dimana sistem yang memanfaatkan SI biasanya merupakan sebuah populasi yang terdiri atas anggota agen-agen yang sederhana, yang berinteraksi sesama anggota, dan juga berinteraksi dengan lingkunga. Pada permasalahan *path planning*, algortima ini dapat digunakan untuk menghasilkan jalur optimal dari informasi jalur sub-optimal sebelumnya [26],[27].

Algoritma PSO merupakan teknik optimasi stokastik berbasis populasi yang relatif baru dan dikembangkan oleh Kennedy dan Eberhart [4]. Algoritma PSO meniru perilaku kawanan serangga, hewan penggembalaan, kawanan burung dan ikan, dimana pencarian makanan yang dilakukan secara bersama-sama menghasilkan sebuah model komputasi yang potensial. Setiap anggota kawanan menyesuaikan pola pencariannya dengan belajar dari pengalamannya sendiri dan pengalaman anggota lainnya. Hasil penyelidikan pada fenomena ini kemudian menghasilkan sebuah model matematika. Anggota gerombolan, yang disebut partikel, mewakili solusi yang mungkin, yang merupakan titik dalam ruang pencarian. Setiap partikel memiliki nilai fitness dan kecepatan untuk menyesuaikan arah terbangnya sesuai dengan pengalaman terbaik kawanan dalam menemukan global optimum dalam ruang solusi n-dimensi.

Ada beberapa prosedur yang harus dilakukan pada algoritma PSO yang dimulai dengan melakukan inisialisasi pada parameter dengan populasi untuk setiap partikel. Inisialisasi yang dilakukan pada setiap partikel akan membangkitkan kecepatan secara random. Setelah proses penginisialisasian terjadi, maka proses evaluasi akan terjadi dimana evaluasi partikel dilakukan dengan cara

membandingkan nilai *fitness* yang mana nilai *fitness* tersebut akan digunakan untuk dijadikan acuan dalam menentukan *Gbest* dan *Pbest* . Langkah selanjutnya adalah proses pencarian *Pbest* dimana proses ini dilakukan dengan cara membandingkan nilai *Pbest* sebelum dan sesudah iterasi dimana jika nilai *fitness* partikel baru yang didapatkan lebih besar dari nilai *fitness* Pbest sebelumnya maka partikel tersebut akan di-*update* atau dijadikan sebagai *Pbest* terbaru. Setelah proses pencarian *Pbest*, prosedur selanjutnya adalah proses pencarian nilai *Gbest* dimana nilai *Gbest* didapatkan dari nilai *fitness Pbest* tertinggi. Prosedur selanjutnya adalah *update velocity score* atau memperbaharui nilai kecepatan dan juga posisi partikel. Nilai kecepatan ini bisa didapatkan melalui penjumlahan momentum dan pengalaman dari *Pbest* dan juga *Gbest*. Algoritma ini akan terus berlanjut dan kembali ke proses evaluasi yang terjadi setelah proses inisialisasi pada parameter dengan populasi untuk setiap partikel sampai dengan *stopping condition* terpenuhi. Adapun Pseudocode dasar dari algoritma PSO tertera pada Gambar 2.8 berikut ini**.**

```
 1 Procedure PSO
 2  Initialize Parameters
 3  Initialize Population
 4  For each particle
 5   Evaluate
 6   Update local best
 7  Update global best
 8  Do
 9   For each particle
10     Update velocity and position
11     Evaluate
12     Update local best
13   Update global best
14  While (Not Terminated)
15 End Procedure
```

Gambar 2. 8 Pseudocode dasar algoritma PSO [38]

Algoritma PSO yang merupakan bagian dari SI memiliki kemampuan untuk untuk menghasilkan jalur optimal dari informasi jalur sub-optimal sebelumnya [26],[27]. Dengan kemampuan tersebut, performansi yang dihasilkan dengan mengkombinasikan algoritma RRT dengan algoritma PSO akan memberikan

performansi yang lebih baik. Namun sepengetahuan penulis, belum ada penelitian sebelumnya yang mencoba untuk meningkatkan kualitas algoritma perencanaan jalur menggunakan hibridasi algoritma RRT dan algoritma PSO.

2.4 Penelitian Terdahulu

Algoritma perencanaan jalur dapat dibagi menjadi beberapa kelas seperti metode variabel, metode pencarian grafik, metode pencarian inkremental. Metode pencarian grafis seperti Djikstra dan A* memiliki sifat resolusi sempurna dan resolusi. Ini berarti bahwa kelengkapan dan optimalitas algoritma dijamin hanya jika resolusi diskritisasi ruang konfigurasi cukup baik. Namun, dalam ruang berdimensi tinggi mungkin sulit untuk mencapai resolusi yang baik. Salah satu teknik pencarian inkremental adalah algoritma SBP dimana algoritma SBP dapat menghindari kebutuhan untuk diskritisasi ruang konfigurasi. Hal ini membuat metode SBP lebih efektif untuk ukuran ruang konfigurasi yang berbeda. Contoh metode SBP adalah RRT dan RRT*. Algoritma RRT yang dikembangkan oleh S. M. Lavalle dan James pada tahun 1998 sebelum dikembangkan menjadi RRT* oleh S. Karaman dan dikembangkan lagi menjadi *informed* RRT* oleh J. D Gammell. Algoritma RRT adalah dapat memberikan solusi yang cepat dalam sistem multi-dimensi. Tetapi memiliki kelemahan yaitu memberikan solusi yang kurang optimal. Algoritma RRT kemudian dikembangkan menjadi RRT* yang memberikan solusi optimal namun dengan kecepatan konvergensi yang lambat.

Terdapat pengembangan yang terjadi secara bertahap pada algortima RRT dan bahkan RRT*. Salah satu pengembangan pada algoritma RRT* antara lain yang dilakukan oleh Kuffner dan Lavalle (2000) yang memperkenalkan algortima RRT-*Connect.* Kumar dkk (2010) kemudian memperkenalkan algoritma *Goal-*

Bias-RRT. Karaman dan Frazzoli (2011) melakukan penelitian dengan menggunakan algoritma RRT* yang kemudian mendapatkan hasil bahwa algoritma yang digunakan mendapatkan hasil asimptotik optimal namun nilai konvergensi yang didapatkan masih terbilang lambat.

Algoritma perencanaan jalur kemudian banyak dikombinasikan untuk memperoleh performansi yang lebih baik. Salah satu hibridasi yang dilakukan adalah dengan algoritma SI seperti ACS dan ACO. Adapun kemampuan dari algortima SI pada perencanaan jalur adalah dapat digunakan untuk menghasilkan jalur optimal dari informasi jalur sub-optimal sebelumnya. P. Guo dan L. Zhu (2012) mengusulkan algoritma perencanaan jalur dengan semut untuk domain kontinu menggunakan RRT* dan $ACO_{\mathbb{R}}$. $ACO_{\mathbb{R}}$ sendiri merupakan perpanjangan dari algoritma ACO untuk domain kontinu.

Nasir, dkk (2013) kemudian mengembangkan algoritma RRT* dengan mengkombinasikan algoritma tersebut menjadi algoritma *RRT*-SMART*. Algoritma hasil hibridasi ini bertujuan untuk meningkatkan atau mempercepat laju konvergensi agar mendapatkan peningkatan efisiensi baik pada waktu maupun pada beban komputasi. Kemudian Gammel, dkk (2014) memperkenalkan algoritma *informed* RRT* yang menggunakan pengambilan sampel berdasarkan informasi pada RRT* setelah solusi pertama ditemukan. Proses pengambilan sampel dilakukan di area elips yang mengelilingi node awal dengan node tujuan.

Kemudian A Viseras, dkk (2016) yang mengimplementasikan algoritma ACS pada permasalahan *path planning* namun implementasi tersebut memerlukan diskritisasi dari ruang konfigurasi namun menggunakan diskritisasi di lingkungan

pencarian akan mengurangi kinerja algoritma dalam ruang konfigurasi yang berdimensi tinggi.

Penelitian yang di lakukan M. Aria (2020) dengan judul "*New Sampling Based Planning Algorithm for Local Path Planning for Autonomous Vehicles"* menghasilkan sebuah usulan berupa algoritma hibridasi antara RRT dan PRM walaupun pada penelitian ini bisa ditingkatkan kecepatan konvergensi, tetapi algoritma tidak dapat bersifat optimal asimptotik. Di tahun yang sama, M. Aria kembali melakukan penelitian "*Path Planning Algorithm Using Informed Rapidly Exploring Random Tree*"* dan menghasilkan sebuah usulan berupa hibridasi antara algoritma RRT* dan algoritma *local search.* Adapun hasil dari pengusulan algoritma ini diperoleh peningkatan pada kecepatan konvergensi yang lebih baik dari algoritma RRT*. Lalu sebuah penelitian lain yang dilakukan oleh M. Aria (2020) yaitu penelitian terkait dengan perencaan jalur dengan judul "Algoritma Perencanaan Jalur Kenderaan Otonom Berbasis Hibridasi Algoritma BFS dan *Path Smoothing*". Hasil dari penelitian ini mengusulkan algoritma baru berdasarkan penggabungan algoritma BFS dan *path smoothing* dimana hasil algoritma yang diusulkan berhasil memiliki sifat optimal asimptotik namun memiliki kecepatan konvergensi yang lambat.

Dari hasil beberapa penelitian yang disebutkan sebelumnya, kombinasi antar dua algoritma seperti hibridasi diantara algoritma RRT dan PRM, BFS yang dikombinasikan dengan *path smoothing,* lalu RRT* dan *local search* memberikan performansi yang lebih baik. Dan bisa disimpulkan bahwa hibridasi antar dua algoritma selalu menghasilkan perbaikan dari penelitian-penelitian sebelumnya. Namun, dari penelitian yang disebutkan sebelumnya, belum ada algoritma yang

bisa menghasilkan sifat asimptotik optimal dan nilai konvergensi yang didapatkan relatif lambat.

Maka dari itu, dinilai akan lebih baik jika terdapat sebuah algoritma yang bisa digunakan untuk perencanaan jalur yang mampu menghasilkan sifat asimptotik optimal dan bisa menghasilkan nilai konvergensi yang cepat. Seperti yang disebutkan sebelumnya bahwa kelebihan dari algoritma SBP seperti RRT dan RRT* adalah kemampuannya untuk menghasilkan jalur dengan cepat, namun memiliki kelemahan yaitu jalur yang dihasilkan kurang optimal sedangkan algoritma SI seperti ACS dan ACO dapat digunakan untuk menghasilkan jalur yang optimal dari informasi jalur sebelumnya yang kurang optimal, sehingga jika digabungkan dinilai bisa mendapatkan nilai konvergensi yang cepat dan memiliki sifat asimptotik yang optimal.

Namun sepengetahuan penulis, belum ada penelitian sebelumnya yang mencoba untuk meningkatkan kualitas algoritma perencanaan jalur menggunakan hibridasi algoritma SBP seperti RRT dan algoritma SI seperti PSO. Penelitian terdahulu menjadi salah satu faktor yang bisa untuk dijadikan sebagai acuan agar dapat memperbanyak teori yang nantinya bisa digunakan sebagai referensi dalam mengkaji setiap penelitian yang akan dilakukan. Adapun penelitian terdahulu yang membahas terkait dengan permasalahan perencanaan jalur bisa dilihat pada Tabel 2.1.

Tabel 2. 1 Penelitian Terdahulu

No	Peneliti	Algoritma	Deskripsi
1.	S.M. LaValle (1998)	Algoritma RRT	Algoritma RRT yang dihasilkan oleh S.M LaValle cukup sederhana sehingga dapat memberikan solusi yang cepat dalam namun terdapat kelemahan yaitu memberikan yaitu memberikan solusi sub-optimal
2.	J.J Kuffner dan S.M LaValle (2000)	RRT-*Connect*	Algoritma RRT-*Connect* ini merupakan versi dua arah dari algoritma RRT
3.	Kumar dkk (2010)	*Goal-Bias-RRT*	Kualitas jalur yang dihasilkan dari algoritma ini lebih baik menggunakan RRT dan mampu meminimalkan kebutuhan komputasi dibandingkan algoritma RRT
4.	S. Karaman dan E. Frazolli (2011)	Algoritma RRT*	Algoritma RRT* yang diusulkan menghasilkan sifat asimptotik optimal dan memperbaiki kualitas jalur yang dihasilkan namun waktu konvergensi yang didapatkan oleh algoritma RRT* ini masih lambat
5.	P. Guo dan L. Zhu (2012)	Algoritma RRT* dan $ACO_{\mathbb{R}}$.	Pengusulan algoritma perencanaan jalur dengan semut untuk domain kontinu menggunakan RRT* dan $ACO_{\mathbb{R}}$.
5.	Nasir dkk (2013)	Kombinasi antara algoritma RRT dan *SMART A*	Memberikan solusi baru dalam hal mempercepat laju konvergensi yang mana akan menyebabkan peningkatan efisiensi pada waktu maupun pada beban komputasi
6.	J.D. Gammell, dkk (2014)	Algoritma *informed* RRT*	Didapatkn hasil yang memperlihatkan bahwa algoritma *informed* RRT* mendapatkan hasil yang lebih baik atau dapat ditingkatkan laju konvergensi dari algoritma RRT* dimana proses pengambilan *sampling* pada algoritma ini dilakukan pada daerah elips yangmelingkupi node awal dengan node akhir
8.	A. Viseras dkk (2016)	Algoritma RRT dan ACO	Implementasi algoritma RRT dan ACO memerlukan diskritisasi dari ruang konfigurasi. Tapi, menggunakan diskritisasi di lingkungan pencarian akan mengurangi kinerja algoritma dalam ruang konfigurasi yang berdimensi tinggi.
9.	S.D. Pendleton, dkk. (2017)	Metode Pencarian Grafik (Djikstra dan A*)	Memiliki sifat *resolution-complete* dan *resolution-optimal* yang artinya, kelengkapan dan optimalitas algoritma hanya bisa dijamin jika resolusi diskritisasi dari ruang konfigurasi cukup baik. Namun, resolusi yang baik mungkin sulit dicapai dalam ruang berdimensi tinggi
10.	M. Aria (2020)	Algoritma *informed* RRT*-Connect dan *local search*	Diperoleh nilai konvergensi yang lebih baik daripada Algoritma RRT*
12.	M. Aria (2020)	Hibridasi algoritma BFS dan *path smoothing*	Diperoleh sifat asimptotik yang optimal namum memiliki kecepatan konvergensi yang lambat.
13.	M. Aria (2020)	Kombinasi antara algoritma RRT* dan Algoritma PRM.	Diperoleh kecepatan konvergensi yang bisa ditingkatkan namun algoritma yang dikombinasikan tidak dapat bersifat asimptotik yang optimal.