

自律的駐車のための想像力を用いたモデルベースの意思決定*。

Ziyue Feng¹, Shitao Chen², Yu Chen³ and Nanning Zheng⁴, Fellow, IEEE

概要- 自律駐車技術は、自律走行研究において重要な概念である。本論文では、駐車に関する問題を解決するための想像的な自律駐車アルゴリズムを提案する。提案アルゴリズムは3つの部分から構成される: 駐車前の結果を予測するための想像力豊かなモデル、与えられた出発点から駐車場までの実行可能な軌道を計画するための改良された迅速探索ランダムツリー(RRT)、駐車タスクの効率を最適化するための経路平滑化モジュール。我々のアルゴリズムは、実際の運動学的な車両モデルに基づいているため、実際の自律走行車へのアルゴリズム適用に適している。さらに、想像力メカニズムの導入により、本アルゴリズムの処理速度は従来の手法の10倍高速化され、同時にリアルタイムプランニングを実現することができる。アルゴリズムの有効性を評価するために、3つの異なる駐車シナリオにおいて、我々のアルゴリズムを従来のRRTと比較した。最終的に、我々のアルゴリズムは従来のRRTよりも安定しており、効率と品質の点でより良い性能を発揮することが示された。

I. INTRODUCTION

セルフパーキングは自律走行を実現するための重要な技術の一つである。自動車技術会(SAE)J3016規格は、駐車支援技術を含む5つの異なるレベルの車両自動化を定義している。最初の3つは、アシストパーキングの進行、すなわち、ドライバーアシスト、パーシャルオートメーション、コンディショナルオートメーションである。第4のレベルは、ハイオートメーションと呼ばれるほぼ自動化された技術であり、最後のレベルは、完全自動化と定義される完全自動化である。現在、道路を走る多くの車両の駐車能力は、レベル1とレベル2の間である。自動駐車は、論理的には自動運転車による自動運転の商業的応用への第一歩であるとして一般に認識されている。現在のところ、自律走行可能な駐車車が完全に実現できる段階にはまだ至っていない。しかし、多くの研究者が継続的な研究を通じてこの段階に到達しようとしている。自動化業界の自動車メーカーやテクノロジー企業の多くは、自走式自動車を主な特徴として、2020年までに完全自動化車が利用可能になると見込んでいる。

自動駐車の実現のために、駐車場が与えられた場合、車両の運動学的制約の下で、自動車を目標場所に操縦するための最適な無衝突軌道を見つける必要があることが駐車場計画タスクである。

*本研究の一部は、中国国家自然科学基金(第61773312号、第61790563号)、大学への学問分野の才能の導入プログラム(第B13043号)の支援を受けた。

Ziyue Feng¹, Shitao Chen², Yu Chen³ are with Institute of Artificial Intelligence and Robotics in Xi'an Jiaotong University, Xi'an, Shannxi, P.R.China e-mail: {brother-yue, chenshitao, yuanchen19960212}@xjtu.edu.cn
Nanning Zheng⁴ 西安交通大学人工智能机器人研究所所長、P.R.China Correspondence: nnzheng@mail.xjtu.edu.cn

しかし、このタスクにはいくつかの困難がある。駐車場計画では、前進と後退の両方が組み合わさっているため、このプロセスは路上計画よりも複雑である。少し後ろに移動すると、車が狭い場所に操縦しやすくなることがある。しかし、この行動の否定的な結果は、人間のような軌道のパフォーマンスの低下や、不快な乗客体験につながる可能性があることである。さらに、駐車軌跡の旋回角度は一般的に陸上軌跡よりも大きい場合、廊下はより狭くなり、前方に移動するときに内側の車輪間の半径の差によって引き起こされる影響が大きくなる。これは、後方に移動するときに外側の車輪で同じである。

RRTアルゴリズムは、運動計画問題を解決するための一般的な手法である。Stevenら[1]によって最初に提案され、その後、駐車計画タスクに適用された[2]。この方法は、開始点からいくつかのアクセス可能な場所を格納するツリーを構築し、その後、ターゲット点がツリーに十分近くなるまで、空間全体を探索するためにツリーをランダムに成長させる。最後に、実現可能な軌道が生成される。RRTは障害物のある複雑な環境を容易に考慮し、扱うことができる。

しかし、RRTと実用的な駐車計画タスクの間には、まだ長い道のりがある。車が目標地点に近づくと、少しのバイアス低減を数ステップで実行する必要がある。これにより、車は素早くガレージに近づいたが、その後、完璧な位置に調整するために、前後に何度も移動する。RRTによる駐車プロセス全体は時間がかかり、軌跡の最終部分が複雑になる。Bi-RRT[3]は、探索時間を短縮するためにターゲットから木を成長させるが、軌跡の最後の部分はまだ複雑である。我々は、車がすでに駐車場に駐車していると想像し、次に、様々な事前に定義された運転戦略で、駐車場から車を運転しようとする。この操作の後、それぞれ20ノードを持つ数十の実行可能なドライビング・アウト・パスが得られる。これらのドライブアウト経路をパーキングイン経路に反転させることで、数十の経路と数百のノードを持つ木が存在することになる。次に、RRTアルゴリズムにおける目標点を、このよく定義されたモデルベースの目標木に拡張する。RRTツリーがターゲットツリーの任意のノードに到達すると、最終的な軌跡が生成される。モデルベースのターゲットツリーはよく定義されているので、ターゲットツリーの軌跡は非常に滑らかになる。車は1回の動作で目標位置に入ることができる。このターゲット拡張により、RRTツリーは、より少ないステップとより少ない時間コストで、ターゲットに近づくことができる。

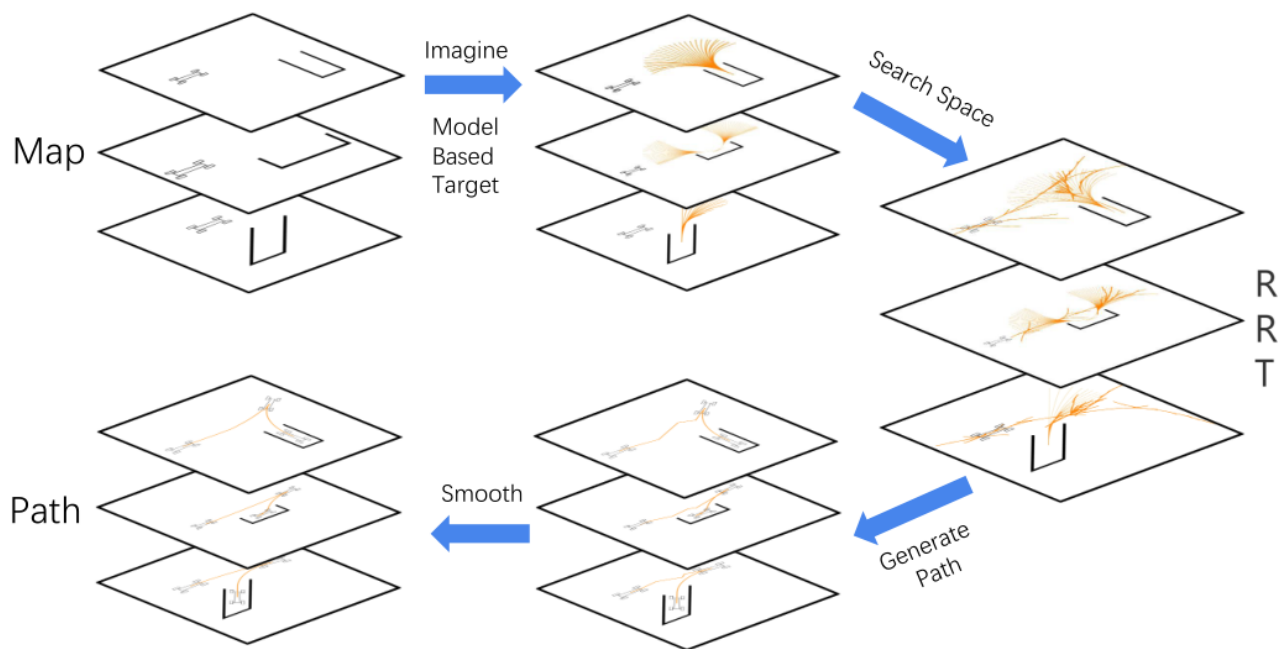


図1. 本アルゴリズムの主なワークフロー駐車場タスクには、垂直、平行、エシュロンの3つのシナリオがある。車が駐車場にすでに駐車していると想像し、特定の駐車シナリオに対応するアルゴリズムを使用してターゲットツリーを構築し、修正RRTを使用して設定空間を検索し、RRTツリーとターゲットツリーからパスを接続する。その後、軌跡が生成される。これで、我々の平滑化アルゴリズムを用いて、車両運動学的制約の下で生成された軌道を平滑化する。

実際、モデルベースのプランニングは、従来のRRTの10倍以上高速である。我々のアルゴリズムは、計画時間と品質において、従来のRRTよりも優れているだけでなく、より安定している。RRTにはまだもう一つの欠点がある。それは、その軌跡が非常に複雑になる可能性があることである。これは、木のすべての辺がランダムな方向に向かって生成されるためである。軌道を滑らかにする方法はいくつかあるが、実際の車の運動学的制約を考慮していない。これは路上計画では問題にならないかもしれないが、駐車場は路上環境よりもはるかに狭いため、平滑化された駐車場の軌跡は実現不可能かもしれない。運動学的制約に従う軌道を平滑化するアルゴリズムを提案する。我々のアルゴリズムによって平滑化された軌跡は、実現可能性が保証されている。これらの改良により、本アルゴリズムは実際の駐車計画タスクに適用することができる。

II. RELATED WORK

駐車場タスクの経路計画[4]とは、駐車場において、与えられた始点から最終目標点までの衝突のない経路を見つけることである。経路計画手法の開発の初期段階は、Dubinsら[5]の車と、ロボット工学の領域におけるReedsとSheppら[6]の車である。しかし、この2つの方法は障害物や連続した旋回角度を考慮していない。そして、モーションプランニングの分野でいくつかの研究が行われている。

自動運転車のための運動計画の方法は、例えばダイクストラアルゴリズム[8]、[9]やA-Starアルゴリズム[10]のようなグラフ探索ベースのプランナ、RRT[1]のようなサンプリングベースのプランナ、[11]、[12]、[13]のようなディープラーニングベースのプランナ、[14]のような数値最適化など、4つのグループに分けることができる[7]。我々の方法は、第2グループに属するRRTに基づいている。RRTはサンプリングに基づく計画手法の一つである。Bi-RRT[3]、RRT*[15]、CL-RRT[16]、DD-RRT[17]など、多くの改良された分岐がある。また、[2]では駐車計画タスクにも適用されている。我々の手法は、基本的なRRTに対して多くの改善を達成した。ターゲット点をモデルベースのターゲットツリーに展開する。この拡張により、生成される経路はより滑らかで人間らしくなることができる。しかし、サンプリングに基づくアプローチの欠点は、計画時間の不確実性である。我々の拡張は、計画時間をより短く、より安定させることができる。すべてのサンプルベースの計画アルゴリズムのもう一つの欠点は、パスのすべてのセグメントがランダムな方向とターゲットで生成されることであり、その結果、パスはより複雑になり、いくつかの冗長なノードを持つことになる。三次多項式[18]、五次多項式[19]、[20]、ベジェ曲線[21]、[22]、[23]、Bスプライン[24]、クロソイド[25]など、この特殊な問題に対して既にいくつかの平滑化アルゴリズムが存在する。これらの平滑化アルゴリズムには、実車のような運動学的制約を考慮することなく、滑らかな曲線への経路を激しく歪ませるという共通の欠点がある。

本論文では、車両の運動学的制約を考慮した新しい平滑化アルゴリズムを提案する。本アルゴリズムで平滑化した経路は実用的であることが保証されている。

III. METHOD

駐車場タスクを、垂直駐車場、平行駐車場、エシュロン駐車場の3つのシナリオに分ける。車が駐車場にすでに駐車していると想像し、3つの異なるシナリオで3つの異なるモデルベースのターゲットツリーを定義する。各対象木は数十のパスと数百のノードを持つ。次に、RRTアルゴリズムの目標点をこのツリーに展開する。RRTツリーがターゲットツリーの任意のノードに到達すると、最終的な軌跡が生成される。この軌跡は、RRTツリーからのパートとターゲットツリーからのパートの2つの部分から構成される。モデルベースのターゲットツリーは、後者を非常に滑らかにし、従来のRRTよりも10倍以上速く処理することができる。我々のアルゴリズムは、より効率的であるだけでなく、計画時間と品質においてより安定している。我々の平滑化アルゴリズムを用いて、車両運動学的制約の下で軌道の前段部分を平滑化する。他の平滑化アルゴリズムとは異なり、我々のアルゴリズムによって平滑化された軌跡は実現可能性が保証されている。この手順を図1に示す。

A. Point Pursuit

本論文では、基本的なアルゴリズムを「点追従」と呼び、車を特定点に近づけるために操縦する最適な旋回角度を計算する。自動車の状態を示すベクトル(X, Y, θ)を考える。 X と Y は2次元曲面における位置、 θ は方位である。図2に示すように、車の始点は $P(0, 0, \frac{\pi}{2})$ 、目標点は $T(X_t, Y_t, \theta_t)$ である。車は旋回角 ϕ を固定して $0.1m$ 移動し、 A 点(X_a, Y_a, θ_a)に到達する。したがって、 P と A の距離をできるだけ小さくするために、最適な ϕ を見つける必要がある。前述の ϕ は、外輪旋回角 ϕ_o と内輪旋回角 ϕ_i から計算される等価旋回角である。

$$X_a = R(1 - \cos \frac{Vt}{R}) \quad (1)$$

$$Y_a = R \sin \frac{Vt}{R} \quad (2)$$

$$\theta_a = \frac{\pi}{2} - \frac{Vt}{R} \quad (3)$$

$$R = \frac{L}{\tan \phi} \quad (4)$$

$$V = 0.1m/s, t = 1s \quad (5)$$

$$distance = (X_a - 0)^2 + (Y_a - 0)^2 + (\theta_a - \frac{\pi}{2})^2 \quad (6)$$

アルゴリズム「点追従」は、距離を最小化するために最適な ϕ を計算する。この動きが無衝突でない場合、 ϕ を小さくして衝突のない ϕ_o を小さくし、 ϕ を大きくして無衝突の ϕ_i を大きくする。点 A を点 P に近づけるために、2つの ϕ の間でより良い ϕ を選択する。このアルゴリズムをアルゴリズム1に示す。

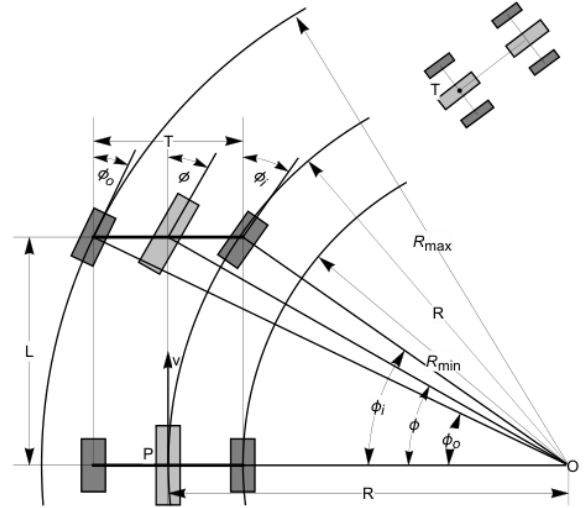


図2. 点追従」アルゴリズムにおける運動モデル。これはアッカーマンモデルに基づいている。車は4つの車輪と2つの等価な車輪を持つ。後方等価ホイールは、車の座標を決定する。この図では、車は開始点 $(0, 0, \frac{\pi}{2})$ に配置されている。

B. Basic RRT

RRTはStevenら[1]によって開発されたもので、空間充填木をランダムに構築することで、複雑な高次元の運動計画問題を解くように設計されたアルゴリズムである。まず、始点が木の唯一の点であり、成長段階は、自由空間内の点 P をランダムにサンプリングし、次に、サンプリングされた点 P に最も近い木の点 N を選択することである。その後、点 N から点 A への実行可能な移動を見つけ、 A と P をできるだけ近づける。次に、点 A と移動をノードとエッジとしてツリーに追加する。このプロセスを何度も繰り返すと、目標点が木に十分近くなるはずである。次に、木をバックトラックすることで、始点から目標点までの実現可能な経路を見つける。このアルゴリズムはアルゴリズム2に示されている。

C. モデルベース目標

基本的なRRTアルゴリズムでは、車と目標との距離が小さいにもかかわらず、許容値 θ より大きい場合、距離をわずかに縮めるのに多大な労力を要する。生成される軌道は複雑になり、車は完璧な位置に調整するために前後に何度も移動する。この問題を解決するために、モデルベースのターゲットを提唱し、ターゲット点をモデルベースの木に展開する。図3に示すように、このモデルベースツリーは数百のノードを持つ。RRTツリーは、ターゲットツリーのノードの誰に近いかが必要である。次に、RRT木とターゲット木のパスを接続し、最終的な軌跡を生成する。この拡張により、アルゴリズムはより簡単にタスクを終了することができ、得られた軌道はより滑らかになる。ターゲットツリーはよく定義されたモデルによって生成されるため、軌跡の最後の部分は完璧になる。車は何の調整もせずに目標位置まで移動する。

Algorithm 1 Point Pursuit

```
function POINT PURSUIT( $N, P$ )  
  movement  $\leftarrow \text{argmin\_distance}(N, P)$   
  simulate_movement(movement)  
  if collide then  
    smaller  $\leftarrow \text{movement.turning\_angle}$   
    larger  $\leftarrow \text{movement.turning\_angle}$   
    while collide do  
      smaller  $\leftarrow \text{smaller} - \text{little}$   
    end while  
    while collide do  
      larger  $\leftarrow \text{larger} + \text{little}$   
    end while  
    ds  $\leftarrow \text{distance}(\text{moving}(\text{smaller}), P)$   
    dl  $\leftarrow \text{距離}(\text{移動}(\text{大きい}), P)$   
    if ds < dl then  
      movement.turning_angle  $\leftarrow \text{smaller}$   
    else  
      movement.turning_angle  $\leftarrow \text{larger}$   
    end if  
  end if  
  A  $\leftarrow \text{movement.destination}$   
  E  $\leftarrow \text{movement.path}$   
  return A, E  
end function
```

Algorithm 2 Basic RRT

```
Tree  $T \leftarrow \text{startPoint}$   
while distance( $T, \text{target\_point}$ ) > tolerance_value do  
  P  $\leftarrow \text{random.choice}()$   
  N  $\leftarrow \text{nearest\_in\_tree}$   
  E, A  $\leftarrow \text{最適移動}(N, P)$   
  T.add(E, A)  
end while  
T.backtrack()
```

このアルゴリズムを実装するためには、ターゲットツリーを生成し、基本的なRRTの「ランダム選択」関数を再定義する必要がある。アルゴリズム3に示す。ターゲットツリーを生成するために、駐車シナリオを垂直駐車、平行駐車、エシュロン駐車の3つのグループに分ける。各モデルは、あらかじめ定義されたいくつかの駐車ルートで構成されている。モデルを生成するために、車が駐車場内にすでに駐車していると想像し、次に、あらかじめ定義された様々な運転戦略で、駐車場から車を運転しようとする。この操作の後、数十の実行可能なドライビング・アウト・パスが得られ、各パスには20のノードがある。これらのドライブアウト経路をパーキングイン経路に逆変換すると、数十の経路と数百のノードを持つツリーが存在することになる。

- 垂直駐車：垂直駐車：垂直な駐車場から車を走らせる場合、しばらく直進した後、一定の旋回角度を保つことを考える。この角度は 2° あたり -30° から 30° に変化するので、31種類の線が得られる。

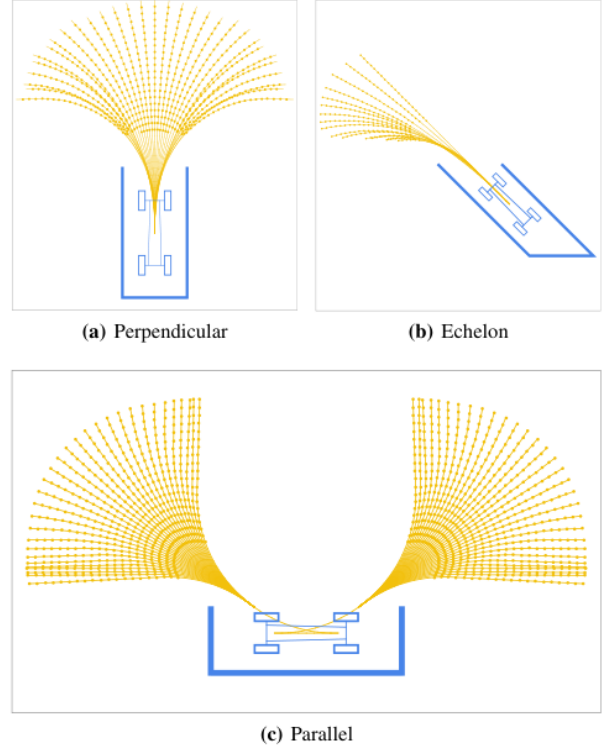


図3. 数十のパスと数百のノードを持つモデルベースのターゲットツリー。

無衝突駐車を前提に、直線部分をできるだけ短く設定する。最後に、1行あたり20点をモデルのノードとして選ぶと、垂直モデルで620個のノードが得られる。

- 平行駐車：平行駐車：平行駐車場の駐車場から車が追い出され、しばらく後方に移動した後、車体が駐車場から半分になるまで最大旋回角度を保つ。そして、残りの経路で -2° あたり -30° から 30° までの一定の旋回角度を選ぶ。このパートでも31行620ノードが得られるが、平行駐車場の駐車場に駐車した場合、2つの可能な向き方向があることを考慮し、得られたデータは62行1240ノードまで2倍になる。
- エシュロン駐車場：垂直駐車と同様に、エシュロン駐車場の車はしばらく直進し、その後一定の旋回角度を保つ。この状況下では、動き全体が後方にあるという違いがある。固定旋回角は 2° あたり 0° から 30° に変化する。最終的に16行、320ノードが得られる。

D. 運動学的制約の下での平滑化

RRTツリーノードのランダムサンプリングのため、パスを複雑にする冗長なノードが常に存在する。したがって、それを平滑化できるアルゴリズムが必要である。この問題には、三次多項式[18]、五分位多項式[19]、[20]、ベジェ曲線[21]、[22]、[23]、Bスプライン[24]、クロソイド[25]など、すでにいくつかの平滑化アルゴリズムがある。

アルゴリズム3 ランダム選択

```
 $r \leftarrow \text{random}(0,1)$ 
if  $r < 0.5$  then
  return random_point_in_free_space
else
  ターゲットツリーのノードを返す(適切な順序) end if
```

Long HanらもRRTに基づく駐車計画のための経路平滑化アルゴリズム[2]を提案している。これらのアルゴリズムはいずれも車の運動学的制約を無視しており、平滑化された経路が実行不可能になる可能性がある。これに対して、我々は、実現可能な経路の生成を保証する我々の提案する平滑化アルゴリズムにおいて、車両の運動学的制約を考慮し、取り組んだ。我々のアルゴリズムは、分割と征服の考え方に基づいている。まず、例えば、パスが N 個のノードを持つ場合、 N ステップ以内に「点追跡」アルゴリズムを使用して、始点から目標点への新しいパスを見つけようとする。成功した場合、新しいパスは元のパスより短くなる。成功しない場合は、パスを中間ノードから2つのパスに分割し、成功またはノードが1つしかないまで、各パスに対して同じ処理を実行する。得られたすべてのパスを接続すると、より滑らかなパスが得られる。点追跡」アルゴリズムは車の運動学的制約下にあるため、生成された経路は実行可能である。

Algorithm 4 Smooth

```
function SMOOTH(path)
  if path just have one node then
    return path
  else
    new_path.add(path.start)
    while new_path is shorter than path do
      new_path.add( point_pursuit to path.end )
      if new_path reached path.end then
        return new_path
      end if
    end while
    path_1, path_2  $\leftarrow$  divide(path)
    path_1  $\leftarrow$  smooth(path_1)
    path_2  $\leftarrow$  smooth(path_2)
    return connect(path_1,path_2)
  end if
end function
```

E. 駐車場計画

駐車場計画に関しては、基本的なRRTアルゴリズムを、前述のモデルベースツリーとスムーズアルゴリズムで修正した。まず、我々のアルゴリズムは駐車タスクを3つの駐車シナリオのいずれかに割り当て、対応する関数を呼び出してターゲットツリーを成長させる。対象木のノードのリストはアルゴリズム3に送られる。最後に、RRTアルゴリズムを実行して、駐車経路を生成する。

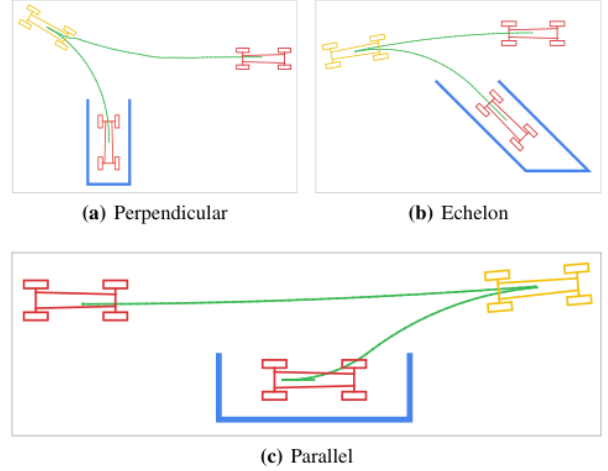


図4. 図4. 3つの異なる駐車シナリオにおいて、我々のアルゴリズムが生成した駐車経路。

RRTの各ステップにおいて、アルゴリズム3はターゲット点を選択する。RRTは1000ステップごとにバイアスが許容値以下かどうかをチェックする。つまり、RRTは少なくとも1000ステップ実行されることになる。バイアスが許容値以下であれば、RRTアルゴリズムを終了し、パスを出力する前にパスを平滑化する平滑化アルゴリズムを開始する。

アルゴリズム5 駐車場計画

```
target_tree  $\leftarrow$  model_based_tree()
node_list  $\leftarrow$  target_tree.nodes
Tree T  $\leftarrow$  start_Point
while distance(T, target_point) > tolerance_value do
  for 1000 times do
    P  $\leftarrow$  random_choice(node_list)
    N  $\leftarrow$  nearest_node_in_tree
    E, A  $\leftarrow$  point_pursuit(N, P)
    T.add(E, A)
  end for
end while
RRT_path  $\leftarrow$  T.backtrack()
RRT_path  $\leftarrow$  smooth(RRT_path)
model_path  $\leftarrow$  target_tree.backtrack()
path  $\leftarrow$  RRT_path + model_path
return path
```

IV. EXPERIMENT

我々のアルゴリズムはC++でプログラムされ、Ubuntuオペレーティングシステムを搭載したパーソナルコンピュータ上で実行される。我々のアルゴリズムは駐車軌跡を生成することができる。垂直、平行、エシュロン3つの典型的なシナリオでアルゴリズムをテストする。その結果を図4に示す。この結果を基本的なRRTと比較する。

表1: ステップと時間コスト

Scenarios	垂直的パーキング		Parallel Parking		Echelon Parking	
Target Type	Target Point	モデルベースのターゲット	Target Point	Model-based Target	Target Point	Model-based Target
Max Steps	9000	1000	5000	1000	13000	1000
Min Steps	4000	1000	2000	1000	6000	1000
Average Steps	6250	1000	3000	1000	8750	1000
Max Time	4.85s	0.20s	2.28s	0.23s	10.30s	0.20s
Min Time	1.46s	0.12s	0.13s	0.17s	2.98s	0.17s
Average Time	2.54s	0.17s	0.92s	0.20s	5.28s	0.18s

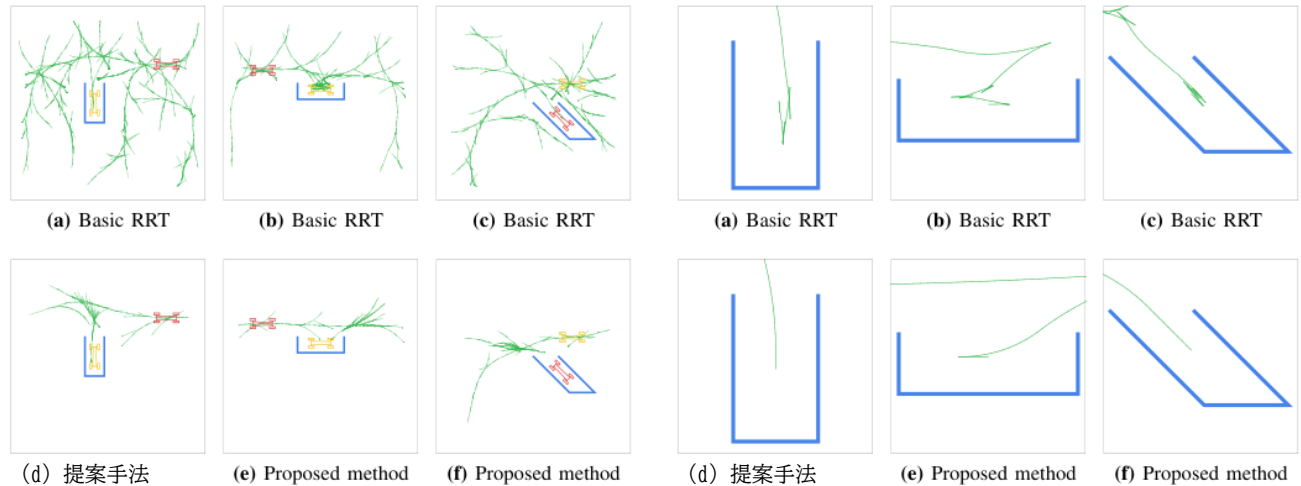


図5. 2つのアルゴリズムによる3つの異なる駐車シナリオでの探索空間。(a), (b), (c)は基本RRTの探索空間、(d), (e), (f)は本アルゴリズムの探索空間である。この検索された空間には、ターゲットツリーの枝は含まれていない。

図6. パスの最終部分。(a), (b), (c)は基本RRTによって生成され、非常に複雑である。車は完璧な位置に調整するために、前方および後方へ複数回移動する。(d), (e), (f)は我々のアルゴリズムによって生成される。非常に滑らかで、車も1回だけ移動して目標位置に到達できる。

A. モデルベース目標

我々の実験では、モデルベースのターゲットには主に2つの利点がある。まず、ターゲットツリーが駐車場の障害物からターゲットポイントを拡張し、ターゲットポイントを1点から数百ノードに拡張することで、RRTツリーがターゲットに到達しやすく、かつ迅速に到達できるようになる。そして、RRTツリーは、ターゲットツリーに十分に近づくために、より少ないステップでより小さな空間を探索する必要がある。検索された空間を図5に示す。一般に、対象木の枝が多ければ多いほど、探索に必要な空間は小さくなる。探索空間とステップが縮小されるため、本アルゴリズムは高速化を実現する。基本RRTと比較した我々のアルゴリズムのステップと時間コストを表1に示す。数百回のテストの結果、基本RRTは数千ステップを要し、いくつかの悪い条件では1万ステップ以上かかることが確認された。これに対し、我々のアルゴリズムは常に1000ステップを要し、決して失敗することはない。前述したように、我々のアルゴリズムは1000ステップまでに停止しない。通常、基本的なRRTの計算には0.13秒～10秒かかるが、我々の計算には常に0.2秒程度かかる。

次に、駐車経路の最終部分はよりスムーズで、安全なマージンも大きくなる。その理由は以下の通りである:すべてのRRTアルゴリズムはパスに接続ゾーンを持つ。RRTツリーは接続ゾーン内のターゲットポイントに接続する。基本的なRRTでは、接続ゾーンはターゲット点の周囲にあるが、我々のアルゴリズムでは、RRTツリーがモデルベースのターゲットツリーに接続する場所である。基本的なRRTにおける駐車経路の最後の部分には接続ゾーンが含まれるため、非常に複雑である。車両は駐車場内で複数回前進・後退し、適切な位置に調整する。通常、駐車場内では狭いので、調整によって衝突を起こすことがある。我々のアルゴリズムの最後の部分は、よく定義されたモデルから生成され、非常に滑らかで人間らしい。車両は1歩で完璧な位置に移動する。

パスの最後の部分を図6に示す。我々のアルゴリズムの接続ゾーンは駐車場の外にあり、そこではより広く、より安全である。ターゲットツリーには数百のノードがあり、接続が非常に簡単である。

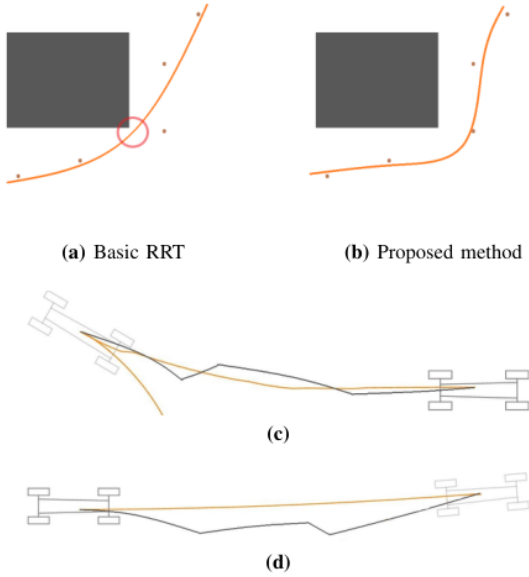


図7. 平滑化アルゴリズムのデモ (a)は従来の平滑化アルゴリズムで平滑化した経路を示す。カーブを滑らかにする経路を激しく歪めると、赤いゾーンで衝突する。(b)は我々のアルゴリズムによって平滑化された経路を示す。(c), (d)において、黒線は平滑化前の経路、オレンジ線は我々の平滑化アルゴリズムによって平滑化された経路である。

接続ゾーンも非常に滑らかになる。本アルゴリズムの接続ゾーンを図4に示す。

B. 車両の運動学的制約による平滑化

サンプルベース計画アルゴリズムの欠点は、パスの各セグメントがランダムな方向にあるため、パスが複雑になり、冗長なノードが発生することである。前述したように、このアルゴリズムにはいくつかの平滑化アルゴリズムが既に存在する。これらの平滑化アルゴリズムにはどちらも共通の欠点があり、運動学的制約を考慮しない一方で、滑らかな曲線への経路を激しく歪める。図7に示すように、歪みはある点で衝突を引き起こす可能性がある。我々の平滑化アルゴリズムは、点追従アルゴリズムを用いて、RRTによって生成された経路に基づいて経路を再計画した。第III部で述べたように、我々のアルゴリズムによって平滑化された経路は実現可能である。平滑化前後の経路を図7に示す。この手順の後、経路がより滑らかになっていることがわかる。図8において、自動車が予め平滑化された経路で操縦しているとき、前輪の旋回角度は -30° から 30° の間で揺れ、車の顔角度も揺れる。しかし、この2つの角度は、平滑化された経路では、はるかに安定で人間的である。

C. 複雑な駐車シナリオ

我々は、様々な複雑な駐車シナリオで我々のアルゴリズムをテストした。限られたスペースの狭いスロットを含む。図9に示すように、我々のアルゴリズムはこのような環境では非常によく機能する。

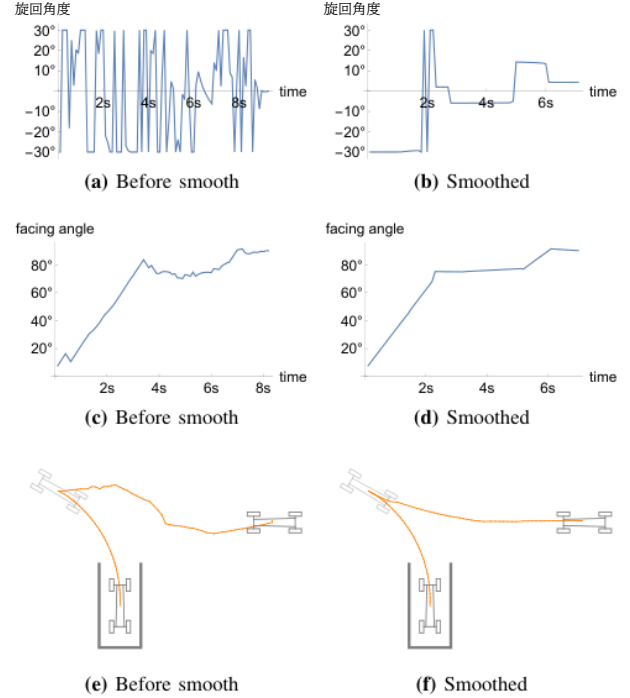


図8. 駐車場における車の旋回角度と対向角度。(a), (b)は平滑化処理前後のフロントホイールの旋回角度を示す。(c), (d)は平滑化処理前後の車の向き角を示す。(e), (f)は対応する経路を示す。ターゲットツリーから生成されるパスは平滑化する必要がないことに注意。(b)のように回転角の急激な変化を制御するために、将来的にいくつかの制約を加える予定である。

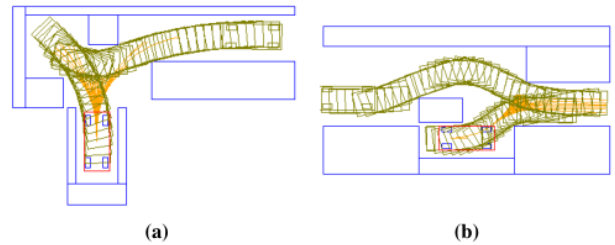


図9. この図は、超狭帯域駐車シナリオにおいて、我々のアルゴリズムが計画したいくつかの経路を示している。赤い長方形が目的地である。

V. CV. 結論と今後の課題

本論文では、RRTアルゴリズムに想像力を導入し、ターゲット点をモデルベースのターゲットツリーに拡張することで、アルゴリズムをより迅速に、生成された軌跡をより滑らかにする。また、実現可能な軌道を生成するために、車両の運動学的制約を考慮した平滑化アルゴリズムを提案した。

我々の方法にはまだいくつかの欠点がある。例えば、我々の車は一定の速度で運転しなければならない。

将来的には、深層強化学習を用いて車速を連続値に一般化し、未知の空間を探索して、利用可能な駐車場を特定する予定である。また、我々のアルゴリズムを実世界でテストする。

REFERENCES

- [1] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.
- [2] L. Han, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on rrt," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5622–5627.
- [3] S. M. Lavalle, J. J. Kuffner, and Jr., "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000, pp. 293–308.
- [4] Y. Song and C. Liao, "Analysis and review of state-of-the-art automatic parking assist system," in *Vehicular Electronics and Safety (ICVES), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.
- [5] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.
- [6] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific journal of mathematics*, vol. 145, no. 2, pp. 367–393, 1990.
- [7] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.
- [8] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.
- [9] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholdt, D. Hong, A. Wicks, T. Alberi, D. Anderson *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.
- [10] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.
- [11] P. Petrov and F. Nashashibi, "Modeling and nonlinear adaptive control for autonomous vehicle overtaking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1643–1656, 2014.
- [12] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain-inspired cognitive model with attention for self-driving cars," *IEEE Transactions on Cognitive and Developmental Systems*, 2017.
- [13] S. Chen, J. Shang, S. Zhang, and N. Zheng, "Cognitive map-based model: Toward a developmental framework for self-driving cars," in *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE, 2017, pp. 1–8.
- [14] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.
- [15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
- [16] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.
- [17] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3856–3861.
- [18] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," in *The 2005 DARPA Grand Challenge*. Springer, 2007, pp. 1–43.
- [19] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," in *The DARPA Urban Challenge*. Springer, 2009, pp. 1–59.
- [20] E. Papadopoulos, I. Poulakakis, and I. Papadimitriou, "On path planning and obstacle avoidance for nonholonomic platforms with manipulators: A polynomial approach," *the International Journal of Robotics research*, vol. 21, no. 4, pp. 367–383, 2002.
- [21] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010.
- [22] K. Jolly, R. S. Kumar, and R. Vijayakumar, "A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 23–33, 2009.
- [23] K. Yang, D. Jung, and S. Sukkarieh, "Continuous curvature path-smoothing algorithm using cubic bezier spiral curves for non-holonomic robots," *Advanced Robotics*, vol. 27, no. 4, pp. 247–258, 2013.
- [24] T. Maekawa, T. Noda, S. Tamura, T. Ozaki, and K.-i. Machida, "Curvature continuous path generation for autonomous vehicle using b-spline curves," *Computer-Aided Design*, vol. 42, no. 4, pp. 350–359, 2010.
- [25] Y. Kanayama and B. I. Hartman, "Smooth local path planning for autonomous vehicles," in *Autonomous robot vehicles*. Springer, 1990, pp. 62–67.