自律駐車のための探索ベース経路計画アルゴリズム：マルチヒューリスティックハイブリッドA*

Jihao Huang[1], Zhitao Liu[1], Xuemin Chi[1], Feng Hong[1], Hongye Su[1]
1. State Key Laboratory of Industrial, Control Technology, Zhejiang University, Hangzhou,310027
E-mail: jihaoh@zju.edu.cn, ztliu@zju.edu.cn, chixuemin@zju.edu.cn, hogfeg.zju.edu.cn, hysu@iipc.zju.edu.cn

概要:本論文では、自律駐車のための新しい方法を提案した。自律駐車は、その利便性から注目されているが、複雑な環境と車両の非ホロノミックな制約のため、衝突のない実現可能な経路を短時間で得ることは困難である。この問題を解決するために、本論文では、マルチヒューリスティックA*とハイブリッドA*の特性を取り入れたマルチヒューリスティックハイブリッドA*(MHHA*)と呼ばれる新しいアルゴリズムを紹介した。そのため、完全性の保証、局所最小値と準最適性の回避、実現可能な経路を短時間で生成することができる。また、本論文では、計算効率を向上させることができる座標変換に基づく新しい衝突チェック手法を提案した。提案手法の性能をシミュレーション実験でハイブリッドA*と比較し、その優位性を証明した。

Key Words: マルチヒューリスティック、経路計画、衝突回避、自律駐車、座標変換。

1 INTRODUCTION

自律走行は、自律走行駐車がこの分野で重要な問題である様々な応用シナリオ[1]により、多くの注目を集めている。自律駐車は経路計画の下位問題である。経路計画とは、スタート位置とゴール位置が与えられれば、実現可能で衝突のない経路を見つけることができるか、解決策がないことを報告することである。経路計画は、障害物や交通ルールのような環境の制約に従うだけでなく、経路の実現可能性を保証する必要があるため、複雑な問題であり、高次元の問題である。

多くの経路計画アルゴリズムが提案されており、その実装により4つのグループに分類される: グラフ探索、サンプリング、補間、数値最適化[2]。グラフ探索に基づくアルゴリズムは、順序サンプリングに基づくアルゴリズムとも呼ばれ、サンプリングアルゴリズムはランダムサンプリングに基づくアルゴリズムも指す。アルゴリズムによって特徴が異なるが、実際の使用では、Rapid-Exploring Random Tree* (RRT*)や最適化手法[3]の組み合わせなど、より良い特徴を得るために異なるアルゴリズムを組み合わせることが多い。ZhangとLiniger[4]は、ハイブリッドA*と最適化ベースの衝突回避(OBCA)アルゴリズムを組み合わせた新しい手法を提案し、この手法を階層的OBCA(HOBCA)と呼び、H-OBCAの主なアイデアは、グローバルパスプランナを使用して粗いパスを生成し、粗いパスをOBCAの初期推測として使用し、OBCAアルゴリズムが車両のダイナミクスを満たす解を生成できることである。OBCAの解の質は初期推測の質に大きく依存するため、より良い初期解を得ることが必要である。

グラフ探索とサンプリングのカテゴリに属するアルゴリズムは、A*とその変種、Rapid-Exploring Random Tree (RRT)とその変種、Probabilistic Roadmap (PRM)、および他の手法のような、グローバルパス解を生成するためによく使用される。しかし、グラフ探索に基づくアルゴリズムは、次元が6より小さい状態空間におけるランダムサンプリングに基づくアルゴリズムよりも効率的である傾向がある[5]。無人航空機(UAV)の経路計画にRRTとその亜種を使用する場合、その性能はより良くなる。そのため、グラフ探索に基づくアルゴリズムは、自律走行車の経路計画においてより一般的である。A*はヒューリスティックに基づくアルゴリズムであり、Dynamic A*、Hybrid A*、Weighted A*など多くのバリエーションがある。ハイブリッドA* [6]は、その品質の高さから自律走行車の経路計画に広く用いられているが、複雑なシナリオでは、ハイブリッドA*によって実現可能な解を得るために多くの時間を要することが多い。Multi-Heuristic A* (MHA*) [7]は最近提案されたアルゴリズムで、多くのヒューリスティックを用いて探索プロセスを高速化する。多くの許容できないヒューリスティックを使用するため、これらのヒューリスティックが提供する情報をフルに活用し、探索プロセスにおけるローカルミニマムを回避することができる。そして、MHA*は解の完全性と準最適性を保証する能力を持っている。

本研究の一部は、中国国家重点研究開発プログラム(助成金NO.2021YFB3301000)、中国国家自然科学基金創造研究グループ科学基金(助成金NO.61621002)、中国国家自然科学基金(NSFC:62173297)、浙江省重点研究開発プログラム(助成金NO.2021C01198,2022C01035)の支援を受けた。

Search-Based Path Planning Algorithm for Autonomous Parking: Multi-Heuristic Hybrid A*

Jihao Huang[1], Zhitao Liu[1], Xuemin Chi[1], Feng Hong[1], Hongye Su[1]

1. State Key Laboratory of Industrial, Control Technology, Zhejiang University, Hangzhou,310027
E-mail: jihaoh@zju.edu.cn, ztliu@zju.edu.cn, chixuemin@zju.edu.cn, hogfeg.zju.edu.cn, hysu@iipc.zju.edu.cn

Abstract: This paper proposed a novel method for autonomous parking. Autonomous parking has received a lot of attention because of its convenience, but due to the complex environment and the non-holonomic constraints of vehicle, it is difficult to get a collision-free and feasible path in a short time. To solve this problem, this paper introduced a novel algorithm called Multi-Heuristic Hybrid A* (MHHA*) which incorporates the characteristic of Multi-Heuristic A* and Hybrid A*. So it could provide the guarantee for completeness, the avoidance of local minimum and sub-optimality, and generate a feasible path in a short time. And this paper also proposed a new collision check method based on coordinate transformation which could improve the computational efficiency. The performance of the proposed method was compared with Hybrid A* in simulation experiments and its superiority has been proved.

Key Words: Multi-heuristic, path planning, collision avoidance, autonomous parking, coordinate transformation.

1 INTRODUCTION

Autonomous driving has received a lot of attention due to its various application scenarios [1], in which autonomous parking is a key problem in this area. Autonomous parking is a sub problem of path planning. Path planning means if the start position and goal position are given, a feasible and collision-free path can be found or reporting there is no solution. Path planning is a complex problem because it not only needs to comply with the constraints of the environment such as obstacles and traffic rules but also needs to guarantee the feasibility of the path, it is a high dimensional problem.

Many path planning algorithms are proposed and they are classified into four groups according to their implementation: Graph search, Sampling, Interpolating and Numerical optimization [2]. Algorithms based on Graph search are also called orderly sampling-based algorithms and the Sampling algorithms also refer to the random sampling-based algorithms. Different algorithms have different features, in actual use, we often combine different algorithms to get better characteristics, such as the combination of Rapid-Exploring Random Tree* (RRT*) and optimization method [3]. Zhang and Liniger [4] proposed a novel method that combines Hybrid A* and the optimization-based collision avoidance (OBCA) algorithm and called this method Hierarchical OBCA (H-OBCA), the main idea of H-OBCA is to use a global path planner to generate a coarse path and then use the coarse path as the initial guess of OBCA, then the OBCA algorithm can generate solution which could meet the dynamics of the vehicle. The solution quality of OBCA depends heavily on the quality of the initial guess, so it is necessary to get a better initial solution.

The algorithm belongs to graph search and sampling category are often used to generate global path solutions, such as A* and its variants, Rapid-Exploring Random Tree (RRT) and its variants, Probabilistic Roadmap (PRM), and other methods. But the algorithm based on graph search tends to be more efficient than the algorithm based on random sampling in states spaces whose dimensions are less than six[5]. When using RRT and its variants in the path planning of Unmanned Aerial Vehicle (UAV), its performance will be better. So algorithms based on graph search are more popular in the path planning of autonomous vehicles. A* is an algorithm based on heuristic and it has many variants such as Dynamic A*, Hybrid A*, Weighted A*, and so on. Hybrid A* [6] has been widely used in path planning for an autonomous vehicle because of its good quality, but in complex scenarios, it often costs much time to get a feasible solution through Hybrid A*. Multi-Heuristic A* (MHA*) [7] is a recently proposed algorithm that uses many heuristics to speed up the search process. It uses many inadmissible heuristics so it could make full use of the information provided by these heuristics to avoid local minimum in the search process. And MHA* has the ability to guarantee the completeness and sub-optimality of the solution. In [8], a method which incorporated the characteristic of MHA* and changed the extension manner of A*

This work was partially supported by National Key R&D Program of China (Grant NO. 2021YFB3301000); Science Fund for Creative Research Group of the National Natural Science Foundation of China (Grant NO.61621002), National Natural Science Foundation of China (NSFC:62173297), Zhejiang Key R&D Program (Grant NO. 2021C01198,2022C01035).

8]では、MHA*の特性を取り入れ、A*の拡張方法を変化させ、経路をより実現可能なものにした手法である。

本論文では、MHA*とハイブリッドA*を組み合わせたマルチヒューリスティックハイブリッドA*(MHHA*)という手法を提案する。本稿の構成は以下の通りである。第2節では、問題定式化を紹介する。第3節では、MHHA*の実装全体を紹介し、ヒューリスティック関数の効果を分析する。セクション4では、本研究のシミュレーション結果を示し、ハイブリッドA*との比較を行う。結論はセクション5で述べる。

2 問題の定式化

本節では、すべてのシナリオ情報は、センサーが多少の誤差をなくして得ることができると仮定する。そして、スタート位置とゴール位置が与えられた場合、自律駐車アルゴリズムは、車両の非ホロノミック制約を満たし、環境と衝突しない経路を計画したり、そのような経路が存在しないことを報告したりすることができる。本研究では、主に古典的なパスプランナの改良に焦点を当て、この問題に必要な情報を紹介する。

2.1 Environment

駐車場問題は、最も一般的な運転行動の1つと考えられている。一般的な駐車シナリオには、並列駐車、逆駐車、斜め駐車がある。正直なところ、このアルゴリズムによってさまざまな駐車場配置を解決できる可能性があるが、本研究では、並行駐車場を研究対象として選択する。図1はこのシナリオを表している。

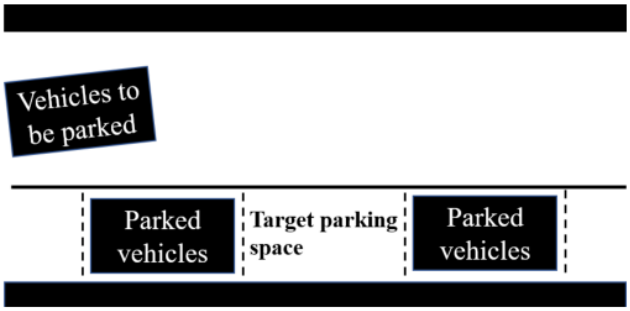

図1: 並列駐車シナリオ

2.2 車両の運動学的制約

この小節では、車両の運動学的制約に焦点を当て、車両運動学モデルを図2に示す。周知のように、駐車のような非構造化シナリオでは、車両は通常低速で走行するため、ダイナミクス制約を考慮せずに車両モデルを単純化することは合理的である。車両の運動学モデルは、ż(t) = f (z(t), u(t))で定式化でき、それを単一トラックモデルで表現することができる[9]:

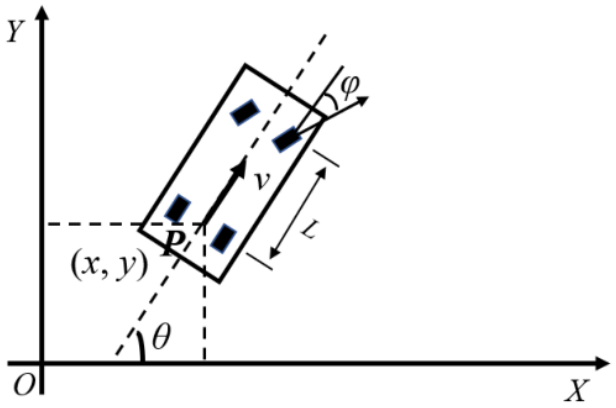

図2:車両の運動モデル。

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \nu(t) \\ \varphi(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} \nu(t) \cdot \cos\theta(t) \\ \nu(t) \cdot \sin\theta(t) \\ a(t) \\ \omega(t) \\ \nu(t) \cdot \tan\varphi(t)/L \end{bmatrix} \tag{1}$$

ここで、Lはホイールベース、(x, y)は後輪車軸の中点(図2の点P)、νはPの速度、φは操舵角、θは方位角、ωは前輪の角速度、aは加速度である。つまり、z(t) ¤ = f (z(t), u(t)) という式を z(t) = [x(t), y(t), ν(t), φ(t), θ(t)], u(t) = [a(t), ω(t)] と再定式化することができる。

前述の状態/制御変数には、車両の物理的または機械的制限を反映する範囲があり、それは次のように表すことができる:

$$\begin{bmatrix} a_{min} \\ \nu_{min} \\ -\omega_{max} \\ -\varphi_{max} \end{bmatrix} \leq \begin{bmatrix} a(t) \\ \nu(t) \\ \omega(t) \\ \varphi(t) \end{bmatrix} \leq \begin{bmatrix} a_{max} \\ \nu_{max} \\ \omega_{max} \\ \varphi_{max} \end{bmatrix} \tag{2}$$

2.3 衝突チェック

本節では、主に環境中の障害物との衝突を回避する方法について述べる。衝突検出プロセスを簡略化するために座標変換を導入することで、計算効率を大幅に向上させる。本作品における車両の形状は、矩形形状として近似される。2つのステップで衝突チェックを行う。まず、車は重なり合う円盤で合理的に近似することができ、長さl、幅ωの長方形は、次のように計算される半径rのn個の円盤で覆うことができる[10]:

$$r = \sqrt{\frac{l^2}{n^2} + \frac{\omega^2}{4}} \tag{3}$$

そして、円盤間の距離は次のように計算できる:

$$d = 2\sqrt{r^2 - \frac{\omega^2}{4}} \tag{4}$$

to make the path more feasible.

In this paper, we propose a method that combines MHA* and Hybrid A* called Multi-Heuristic Hybrid A* (MHHA*). The article is organized as follows. Section 2 introduces the problem formulation. Section 3 introduces the whole implementation of MHHA* and analyzes the effect of heuristic functions. Section 4 shows the simulation results of this work and makes a comparison with Hybrid A*. Conclusions are drawn in Section 5.

2 PROBLEM FORMULATION

In this section, we assume that all the scenario information can be obtained by the sensors without some errors creeping in. Then if the start position and the goal position are given, the autonomous parking algorithm could plan a path that could meet the non-holonomic constraints of the vehicle and does not collide with the environment, or report the non-existence of such a path. This work mainly focuses on the improvement of the classical path planner and we will introduce some necessary information for this problem.

2.1 Environment

The parking problem is considered as one of the most common driving behaviors. Common parking scenarios include parallel parking, reverse parking, and oblique parking. To be honest, different parking arrangements could be solved through this algorithm, in this work we choose parallel parking as the research target. Figure 1 could represent this scenario.

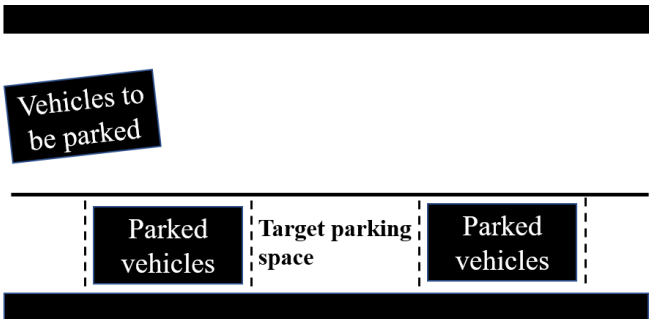

Figure 1: The parallel parking scenario.

2.2 Vehicle Kinematic Constraints

This subsection main focuses on the kinematic constraints of the vehicle and the vehicle kinematics model can be shown in Figure 2. As is known to us, driving in the unstructured scenario such as parking, the vehicle usually drives at a low velocity, so it is reasonable for us to simplify the vehicle model without thinking about the dynamics constraints. The kinematic model of vehicle can be formulated by $\dot{z}(t) = f(z(t), u(t))$ and we can use the single-track model to represent it [9]:

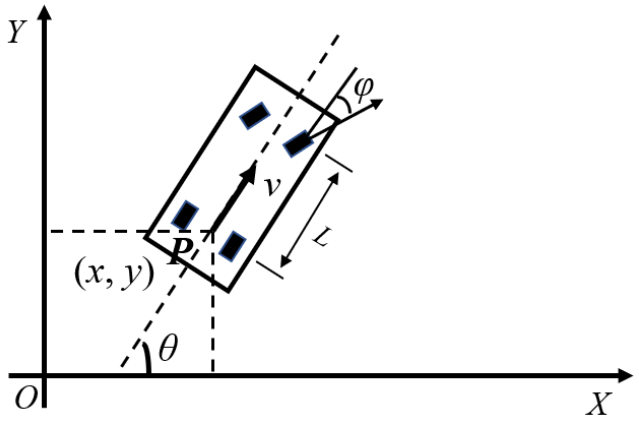

Figure 2: The motion model of vehicle.

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \nu(t) \\ \varphi(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} \nu(t) \cdot \cos\theta(t) \\ \nu(t) \cdot \sin\theta(t) \\ a(t) \\ \omega(t) \\ \nu(t) \cdot \tan\varphi(t)/L \end{bmatrix} \tag{1}$$

where L is the wheelbase, (x, y) is the rear-wheel axle midpoint (point P in Figure 2), ν is the velocity of P, φ is the steering angle, θ is the heading direction, ω is the angular velocity of the front wheel, a is the acceleration. So we can reformulate the equation $\dot{z(t)} = f(z(t), u(t))$ as $z(t) = [x(t), y(t), \nu(t), \varphi(t), \theta(t)]$, $u(t) = [a(t), \omega(t)]$.

The state/control variables mentioned earlier have a range, which reflects the physical or mechanical limitations of the vehicle, it can be represented by:

$$\begin{bmatrix} a_{min} \\ \nu_{min} \\ -\omega_{max} \\ -\varphi_{max} \end{bmatrix} \leq \begin{bmatrix} a(t) \\ \nu(t) \\ \omega(t) \\ \varphi(t) \end{bmatrix} \leq \begin{bmatrix} a_{max} \\ \nu_{max} \\ \omega_{max} \\ \varphi_{max} \end{bmatrix} \tag{2}$$

2.3 Collision Check

This section mainly focused on how to avoid collisions with obstacles in the environment. We introduce coordinate transformation to simplify the collision detection process, which will improve computational efficiency greatly. The geometry of the vehicle in this work is approximated as a rectangle shape. We will do a collision check with two steps. Firstly, the car could be reasonably approximated by overlapping circular disks, a rectangle with length l and width ω can be covered by n circular disks of radius r calculated as [10]:

$$r = \sqrt{\frac{l^2}{n^2} + \frac{\omega^2}{4}} \tag{3}$$

And the distance between circular disks can be calculated as:

$$d = 2\sqrt{r^2 - \frac{\omega^2}{4}} \tag{4}$$

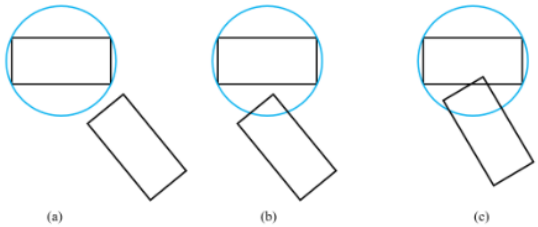

図3: 粗い衝突チェックの例。

本研究では、図3に示すように、矩形形状をカバーするために1枚の円形円盤を使用する。このようにして粗いチェックを行うことができる。矩形の中心から障害物上の点までの距離が円盤の半径より大きい場合、車両が障害物に衝突しないように確認することができ、図3(a)のように示すことができる。しかし、距離が円盤の半径より小さい場合、図3(b)、(c)に示すように、車両は衝突する可能性がある。そこで、次のステップが必要である。

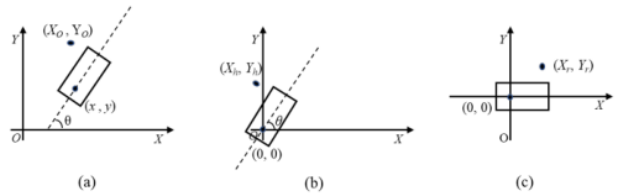

図4:座標変換による完全な衝突チェック。(a):原点座標系。(b):水平変換。(c):回転変換。

図3(b)のような欠陥を克服するために、その点が矩形の内側にあるかどうかを判断する必要がある。最初のステップで衝突をチェックするだけで、たとえ解があったとしても、スペースの無駄のために解が得られないことがある。この問題に対処するために座標変換を用いるが、計算効率を大幅に向上させることができる。図4(a)からわかるように、これは地球座標系、(x, y)は地球座標系における後輪車軸の中点の座標、(X_O, Y_O)は地球座標系における障害物上の点の座標、θは方位角である。座標変換は、水平変換と回転変換の2つの部分から構成される。図4(b)では、地球座標系を水平変換して得られる座標系を確立している。この座標系は、後輪車軸の中点を座標原点とし、(X_h,Y_h)は水平変換後の障害物上の点の座標であり、次のように計算できる:

$$\begin{bmatrix} X_h \\ Y_h \end{bmatrix} = \begin{bmatrix} X_O \\ Y_O \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \tag{5}$$

次に、図4(c)に示すように、後輪車軸の中点を座標原点とし、その水平軸が車両に平行な座標系を確立する。

したがって、この座標系に基づいて障害物上の点の座標を求めることができれば、その点が矩形の内側にあるかどうかを判断しやすくなる。(X_r, Y_r)は図4(c)に示す座標系における障害物上の点の座標であり、次のように計算できる:

$$\begin{bmatrix} X_r \\ Y_r \end{bmatrix} = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} \begin{bmatrix} X_h \\ Y_h \end{bmatrix} \tag{6}$$

そこで、地球座標系での座標から水平変換と回転変換を行うことで、図4(c)のような座標系での障害物上の点の座標を求めることができるので、次のようにまとめることができる。

$$x_h = Rx_o + t \tag{7}$$

ここで、x_hは変換座標系の座標、Rは回転行列、x_oは地球座標系の座標、tは並進ベクトルを表す。

3 ALGORITHM

本節では、主にMultiHeuristic A* (MHA*)アルゴリズムの特徴を取り入れながら、Hybrid A*をベースとしたMultiHeuristic Hybrid A* Algorithm(MHHA*)を紹介することに焦点を当てる。ハイブリッドA*の拡大方法とMHA*による高速探索効率により、運動学的実現可能性を保証することができる。

3.1 Heuristic

このサブセクションでは、主にヒューリスティック関数の効果を分析し、マルチヒューリスティックハイブリッドA*アルゴリズムで使用されるヒューリスティック関数を決定することに焦点を当てる。ヒューリスティック関数に基づくパスプランナーの性能は、ヒューリスティック関数の性能に大きく依存する[7]。ヒューリスティック関数は、無駄な領域の探索を減らし、探索プロセスを高速化するために使用されるが、許容されるヒューリスティックのみが最適な結果を導くことができる。ヒューリスティック関数は、現在の状態からゴール状態までのコストを決して過大評価しない場合、現在の状態とヒューリスティック関数によって計算されたゴール状態との間の距離が、現在の状態とゴール状態との間の実際の距離よりも常に小さいことを意味する[11]。ヒューリスティック関数は、h(x_{goal}) = 0 かつ 0 0 h(s) ≦ h(s) + c(s, s) を満たす場合、図5のように整合的である: ここで、x_sは開始状態、x_gはゴール0状態、Sは現在状態、Sは現在状態の子ノード、g(s)は現在状態の子ノード、g(s)はS 0とx_sの間のコスト、h(s)はSとx_gの間のコスト、c(s, s) 0 0はSとSの間のコストを表すので、g(s)はg(s) = g(s) + c(s, s) で表される0 0となりうる。総コスト f (s) = g(s) + h(s).ヒューリスティック関数が現在の状態とゴール状態の間のコストを過小評価する場合、コストが実際のコストより小さくなることを意味し、最適解を求めることができる。

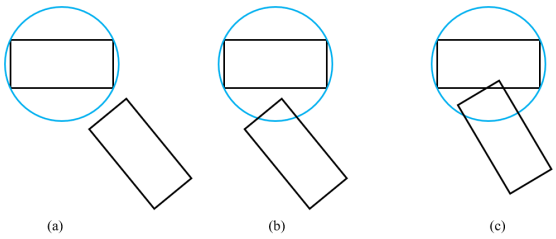

Figure 3: Examples for coarse collision check.

In this work, we use one circular disk to cover the rectangle shape, as it can be shown in Figure 3. We can do a coarse check this way. When the distance between the center of the rectangle and the point on the obstacle is greater than the radius of the circular disk, we can make sure that the vehicle won't collide with the obstacle, it can be shown as Figure 3 (a). But when the distance is less than the radius of the disk, the vehicle has the possibility of collision, as it can be shown in Figure 3 (b), (c). So we need the next step.

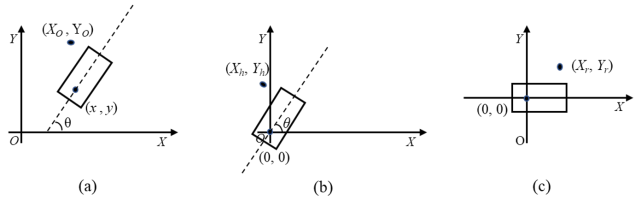

Figure 4: Complete collision check through coordinate transformation.(a):origin coordinate system.(b):horizontal transformation.(c):rotation transformation.

We need to determine whether the point is inside the rectangle in order to overcome the defect shown in Figure 3 (b). If we only check collision through the first step, even if there is a solution, we may not get a solution because of the waste of space. We use coordinate transformation to cope with this problem, it can improve computational efficiency greatly. As we can see in Figure 4 (a), this is the earth coordinate system, (x, y) is the coordinate of the rear-wheel axle mid-point in the earth coordinate system, (X_O, Y_O) is the coordinate of the point on the obstacle in the earth coordinate system, θ is the orientation angle. The coordinate transformation consists of two parts, horizontal transformation and rotation transformation. In Figure 4 (b), we establish a coordinate system which is obtained by horizontal transformation of the earth coordinate system, this coordinate system uses the rear-wheel axle mid-point as the coordinate origin, (X_h,Y_h) is the coordinate of the point on obstacle after horizontal transformation, we can calculate it as:

$$\begin{bmatrix} X_h \\ Y_h \end{bmatrix} = \begin{bmatrix} X_O \\ Y_O \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \tag{5}$$

Then we establish a coordinate system that uses the rear-wheel axle mid-point as the coordinate origin and its horizon axis is parallel to the vehicle, as it can be shown in Figure 4 (c). So if we can get the coordinate of the point on the obstacle based on this coordinate system, it will be easy for us to judge whether the point is inside the rectangle.(X_r, Y_r) is the coordinate of the point on the obstacle in the coordinate system shown in Figure 4 (c), we can calculate it as:

$$\begin{bmatrix} X_r \\ Y_r \end{bmatrix} = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} \begin{bmatrix} X_h \\ Y_h \end{bmatrix} \tag{6}$$

So we can get the coordinate of the point on the obstacle in coordinate system shown in Figure 4 (c) through horizontal transformation and rotation transformation from the coordinate in the earth coordinate system, we can summarize it as:

$$x_h = Rx_o + t \tag{7}$$

where x_h represents the coordinate in the transformed coordinate system, R represents a rotation matrix, x_o represents the coordinate in the earth coordinate system, t represents the translation vector.

3 ALGORITHM

This section mainly focused on introducing the Multi-Heuristic Hybrid A* Algorithm(MHHA*), which is based on Hybrid A* while incorporating the character of the Multi-Heuristic A* (MHA*) Algorithm. We can assure the kinematic feasibility through the expanding manner of Hybrid A* and fast search efficiency through MHA*.

3.1 Heuristic

This subsection mainly focuses on analyzing the effect of the heuristic function and deciding the heuristic function used in the Multi Heuristic Hybrid A* Algorithm. The performance of path planners based on heuristic function depends heavily on the performance of heuristic function [7]. Heuristic function is used to reduce the exploration of useless areas and speed up the search process, but only an admissible heuristic can lead to the optimal results. A heuristic function is admissible if it never overestimates the cost from the current state to the goal state, it means that the distance between the current state and the goal state calculated by the heuristic function always less than the actual distance between between the current state and the goal state [11]. A heuristic function is consistent if it satisfies $h(x_{goal}) = 0$ and $h(s) \leq h(s^{'}) + c(s, s^{'})$, it can be shown as Figure 5:
where x_s represents the start state, x_g represents the goal state, S represents the current state, $S^{'}$ represents the child node of the current state, $g(s)$ represents the cost between S and x_s, $h(s)$ represents the cost between S and x_g, $c(s, s^{'})$ represents the cost between S and $S^{'}$, so $g(s^{'})$ could be represented by $g(s^{'}) = g(s) + c(s, s^{'})$ and the total cost $f(s) = g(s) + h(s)$. When the heuristic function underestimates the cost between the current state and the goal state, it means the cost is less than the actual cost, so the optimal solution can be found. When $h(s) = 0$, then the algorithm

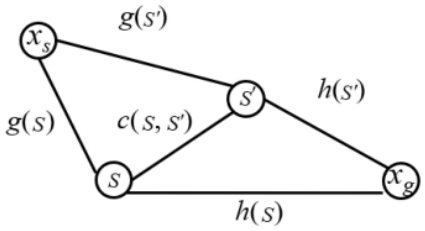

図5:ヒューリスティック関数の一貫した特性。

をダイクストラに縮退させる。ヒューリスティック関数が現在の状態とゴール状態の間のコストを過大評価すると、探索速度は増加するが、解が最適でない可能性があるため、最適性と速度のトレードオフが行われることになる。

一つは障害物を無視しながら車両の非ホロノミック制約を考慮するものであり、もう一つは車両の非ホロノミック制約を無視しながら障害物を考慮するものである。これらのヒューリスティックの詳細については、[6]を参照してください。

3.2 マルチヒューリスティックA*アルゴリズム

この小節では、主にMHA*のレビューを行うことに焦点を当てる。MHA*は多くのヒューリスティックスを使用しており、そのうちの1つは許容されるが、他のものは許容されない。許容されるヒューリスティックも一貫しており、解の最適性と完全性を保証することができるが、複雑なシナリオで探索するときに局所最小に陥りやすい欠陥がある。異なる許容できないヒューリスティックは異なる検索場所で有用である可能性があり、これは異なる許容できないヒューリスティックが異なる指針を提供する可能性があることを意味する。MHA*は、許容されるヒューリスティックと許容されないヒューリスティックを組み合わせることで、完全で境界のある準最適解を得ることができる。MHA*は、これらの許容できないヒューリスティックの1つがうつ病領域周辺の探索を導くことができれば、高速な収束を保証することができる。MHA*は、準最適性を保証する方法で、異なる許容できないヒューリスティックで複数の検索を実行する。MHA*には、独立多ヒューリスティックA*(IMHA*)と共有多ヒューリスティックA*(SMHA*)の2つのバリエーションがある。MHA*の詳細については、論文[7]を参照されたい。

3.3 マルチヒューリスティックハイブリッドA*アルゴリズム

このサブセクションでは、主にMultiHeuristic Hybrid A* Algorithmの紹介に焦点を当て、このアルゴリズムはハイブリッドA*の特性とMHA*の特性を取り入れ、解が運動学的に実行可能、完全、かつ最適でないことを保証することができる。

MHHA*はAlg 1に示されている。MHHA*は、開始状態とゴール状態を結ぶ経路を返すか、または、そこで報告される。

h(s) = 0のとき、アルゴリズムは開始状態とゴール状態を結ぶ無衝突解を持たない。最初のステップは、初期化処理を完了することである(1-10行目)。そして、このアルゴリズムは、OP EN_0が空集合であるかどうかをチェックする。空集合であれば、アルゴリズムは解を返さない(40行目)。そして、アルゴリズムはノードを拡張する。MHHA*はω因子を用いて、許容ヒューリスティック関数で探索されるアンカー探索よりも、許容できない探索を優先する。MHHA*は、アンカー探索の最小鍵からωファクター以内の解を探索する限り、許容できない探索をラウンドロビン方式で実行する。もし、許容できないヒューリスティックの最小鍵がアンカーヒューリスティックの最小鍵のω倍という境界を満たさない場合、この許容できない探索は中断され、代わりにアンカー探索が実行される。これは、どのノードが拡張されるかを決定するプロセスである。どのノードを拡張するかを決定した後、14-24行目の処理は26-36行目と同じである。前のものを例にとると、まず、アルゴリズムは、拡張される現在のノードがゴールノードであるかどうかをチェックする。もしそうであれば、アルゴリズムは、親ノードをバックトラックして全経路を返す。もしそうでなければ、このアルゴリズムは解析解を得ようとするが、車両が前進と後退の両方向に移動できるため、Reeds-Sheep曲線を選択する。Reeds-Sheep曲線によって返されたパスが衝突のないものである場合、アルゴリズムはReeds-Sheep曲線と組み合わされたパスを返し、親ノードをバックトラックする。Reeds-Sheep曲線が障害物に衝突した場合、アルゴリズムはExpandnode(s)関数を呼び出して現在のノードの拡張を終了する。
Expandnode(s)関数はAlg 2に示されている。このアルゴリズムはA*の展開に似ているが、許容されないヒューリスティックだけでなく、許容されないヒューリスティックもノードを拡張するために許容されるヒューリスティックを使うわけではない。アルゴリズムに登場したkey(s, i)関数はAlg 3に示されており、1つのノードの総コストを計算するために使用される。

4 シミュレーション結果

本節では、MHHA*のシミュレーション結果を示し、MHHA*とハイブリッドA*を比較する。

4.1 シミュレーションのセットアップ

シミュレーション結果はすべてMATLABで行い、i7-9750 CPU 2.60GHz、16GB RAMを搭載したノートパソコンでMicrosoft Windows 10で実行した。車両は4.7×2mの矩形でモデル化し、車のホイールベースLは2.7mとした。そして、この作業では、ステアリング角は、$\phi \in [-0.6, 0.6]$ rad（約 $\pm 34^*$）の間に制限される。本研究では、図6に示すように、平行駐車を研究シナリオとして選択し、水平方向はX∈[-21, 21]m、別の方向はY∈[-1, 11]m、駐車場は深さ3.0m、長さ7.2mに制限している。
開始状態を静的（$\nu = 0$）にし、初期姿勢を右とする。本研究では、2つのIn MHHA*アルゴリズムグリッドマップを実装し、X方向とY方向の両方でグリッドのサイズは0.3mである。

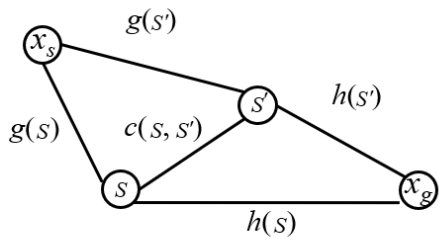

Figure 5: The consistent characteristic of the heuristic function.

degenerated into Dijkstra. When the heuristic function overestimates the cost between the current state and the goal state, the search speed will increase, but the solution may not be optimal, it means a trade-off is made between optimality and speed.

Two typical admissible heuristics are used, one takes into account the non-holonomic constraints of the vehicle while neglecting the obstacle and another considers the obstacle while neglecting the non-holonomic of the vehicle, we choose the maximum of both. For more detailed information about these heuristics, please refer to [6].

3.2 Multi-Heuristic A* Algorithm

This subsection mainly focuses on having a review of MHA*. MHA* uses many heuristics, one of those heuristics is admissible, while others are inadmissible. The admissible heuristic is also consistent, it can assure the optimality and completeness of the solution but it has the defect that is easy to fall into local minimum when searching in the complex scenarios. Different inadmissible heuristics may be useful in different search places, which means different inadmissible heuristics could provide different guiding powers. MHA* combines the admissible heuristic and the inadmissible heuristics so it could get a complete and bounded sub-optimal solution. MHA* could assure fast convergence if one of these inadmissible heuristics can guide the search around the depression regions. MHA* runs multiple searches with different inadmissible heuristics in a manner that will assure sub-optimality. MHA* has two variants, including Independent Multi-Heuristic A*(IMHA*) and Shared Multi-Heuristic A*(SMHA*). For more details about MHA*, it is recommended to see the full paper [7].

3.3 Multi-Heuristic Hybrid A* Algorithm

This subsection mainly focuses on introducing the Multi-Heuristic Hybrid A* Algorithm, this algorithm incorporates the characteristic of Hybrid A* and the characteristic of MHA*, so it can assure the solution is kinematically feasible, complete, and sub-optimal.

MHHA* is presented in Alg 1. It will return the path connecting the start state and the goal state or report there's no collision-free solution connecting the start state and the goal state. The first step is to complete the initialization process(line1-10). Then this algorithm checks whether the $OPEN_0$ is an empty set, if it is an empty set, the algorithm will return no solution(line 40). Then the algorithm will expand the node. MHHA* uses ω factor to prioritize the inadmissible searches over the anchor search which is searched by the admissible heuristic functions. MHHA* executes the inadmissible searches in a round-robin manner as long as they explore solutions within ω factor of the minimum key of the anchor search. If the minimum key of an inadmissible heuristic cannot satisfy the bound of ω times of the minimum key of an anchor heuristic, this inadmissible search will be suspended and the anchor search will run instead. This is a process that determines which node will be expanded. After determining which node to be expanded, the process in lines 14-24 is the same as lines 26-36. Take the previous one as an example, first, the algorithm will check whether the current node to be expanded is the goal node, if it is, the algorithm will return the whole path through backtracking the parent node. If not, the algorithm tries to get an analytic solution, this algorithm chooses Reeds-Sheep curves because it allows the vehicle to move in both forward and backward directions. If the path returned by Reeds-Sheep curves is collision-free, then the algorithm will return the path which is combined with the Reeds-Sheep curves and backtracking the parent node. If the Reeds-Sheep curve collides with the obstacle, then the algorithm will call $Expandnode(s)$ function to finish the expansion of the current node.

The $Expandnode(s)$ function is presented in Alg 2. This algorithm is similar to the expansion of A*, in addition to it not only using an admissible heuristic to expand the node as well as inadmissible heuristics. The $key(s, i)$ function that appeared in the algorithm is presented in Alg 3, which is used to calculate the total cost of one node.

4 SIMULATION RESULTS

In this section, we provide simulation results of MHHA* and compare the MHHA* with Hybrid A*.

4.1 Simulation Setup

All the simulation results are conducted in MATLAB and executed on a laptop with i7-9750 CPU 2.60GHz with 16GB of RAM under Microsoft Windows 10. We model the vehicle as a rectangle of size 4.7×2m and the wheelbase L of the car is 2.7m. And in this work, the steering angle is limited between $\varphi \in [-0.6, 0.6]$ rad (approximately $\pm 34°$). This work choose parallel parking as research scenario, the horizontal direction is limited between $X \in [-21, 21]$ m, another direction is limited between $Y \in [-1, 11]$ m and the parking spot is 3.0 m deep and 7.2 m long, as is shown in Figure 6.

In MHHA* algorithm grid map is implemented and in both X and Y direction the size of the grid is 0.3 m. And this algorithm penalizes the reverse movement, numbers of switchbacks, and the jerk of the vehicle. For admissible heuris-

アルゴリズム1 マルチヒューリスティックハイブリッドA*。

関数 MHHA*(s_{start}, s_{goal})

$g(s_{goal}) \leftarrow \infty$;
$g(s_{start}) \leftarrow 0$;
$bp(s_{start}) \leftarrow null$;
$bp(s_{goal}) \leftarrow null$;
$N \leftarrow 0$;
for $i = 0$ to n **do**
  $OPEN_i \leftarrow \emptyset$;
  $CLOSEi \leftarrow \emptyset$;
  insert s^{start} in $OPEN_i$ with $key(s_{start}, i)$ as priority;
end for
while $OPEN_0$ not empty **do**
  for $i = 1$ to n **do**
    if $OPEN_i.Minkey() \leq \omega OPEN_0.Minkey()$ **then**
      $N \leftarrow N + 1$;
      if $g(s_{goal}) \leq OPEN_i.Minkey()$ **then**
        return path pointed by bp(s_{goal});
      end if
      $s \leftarrow OPEN_i.Top()$;
      if $mod(N, setvalue) == 0$ **then**
        RSpath $\leftarrow$ RSCurve(s, s_{goal});
        if RSpath is Collision free **then**
          return RSpath with path pointed by bp(s)
        end if
      end if
      Expandnode(s);
    else
      $N \leftarrow N + 1$;
      if $g(s_{goal}) \leq OPEN_0.Minkey()$ **then**
        return path pointed by bp(s_{goal});
      end if
      $s \leftarrow OPEN_0.Top()$;
      if $mod(N, setvalue) == 0$ **then**
        RSpath $\leftarrow$ RSCurve(s, s_{goal});
        if RSpath is Collision free **then**
          return RSpath with path pointed by bp(s)
        end if
      end if
      Expandnode(s);
    end if
  end for
end while
return NULL

一方は障害物を無視しつつ、もう一方は障害物を無視しつつ、車両の非ホロノミック制約を考慮し、もう一方は車両の非ホロノミック制約を無視しつつ、両方の最大値を選択する。便宜上、この実装では、MHHA*は1つの許容できないヒューリスティックのみを選択し、許容できるヒューリスティックを膨張させることで許容できないヒューリスティックを得る。

4.2 Results

本研究のシミュレーションでは、端の位置を（x_f, y_f, ϕ_f, ν_f ）=（-1.35, 1.5, 0, 0）に固定し、

アルゴリズム2 親ノードの拡張

機能：ノードの展開

Remove s from $OPEN_i, \forall i = 0 \dots n$;
Add s to $CLOSE_i, \forall i = 0 \dots n$;
for each $s^{'}$ in Succ(s) **do**
  if $s^{'}$ in $CLOSE_0$ **then**
    Continue;
  else
    $g(s^{'}) \leftarrow g(s) + c(s, s^{'})$;
    $bp(s^{'}) \leftarrow s$;
    insert/update $s^{'}$ in $OPEN_0$ with $key(s^{'}, 0)$ as priority;
    if $s^{'}$ not in any $CLOSE_i$, $\forall i = 1 \dots n$ **then**
      for $i = 1$ to n **do**
        insert/update $s^{'}$ in $OPEN_i$ with key($s^{'}, i$);
      end for
    end if
  end if
end for

アルゴリズム3 1つのノードの総コストを計算する

Function: key(s, i)

return $g(s) + h_i(s)$

そして、このアルゴリズムは、逆の動き、スイッチバックの数、車両のジャークにペナルティを与える。許容されるヒューリスティックシナリオについては、MHHA*の性能を確認するために、図7に示すような前方進入駐車と、図8に示すような後方進入駐車のシナリオを選択した。障害物は赤い点で、解析解は青い線で表されている。図7の開始位置は（x_s, y_s, ϕ_s, ν_s ）=（-9, 8.0, 0, 0）とし、図8の開始位置は（x_s, y_s, ϕ_s, ν_s ）=（12, 8.0, 0, 0）とする。MHHA*の探索過程は非常に高速であり、このアルゴリズムは準最適性を保証しながら短時間で無衝突解を得ることができる。

4.3 考察と比較

この小節では、MHHA*とハイブリッドA*を比較する。両シナリオにおけるハイブリッドA*のシミュレーション結果を図9と図10に示す。また、MHHA*とハイブリッドA*の主要な性能指標の比較を表1、表2に示す。拡張ノード数、反復回数、拡張時間など、多くの点でMHHA*がハイブリッドA*より優れていることがわかる。経路長の面でのみ、MHHA*はハイブリッドA*より悪い。MHHA*は最適性と計算効率のトレードオフを行うため、最適性が保証されない。MHHA*で計算された経路は最適ではないが、実現可能であるため、OBCAの初期解として使用することができる。そして、MHHA*とハイブリッドA*が計算する経路はホモトピーであり、両解は同じ解に収束するので、計算効率は最適性よりも重要であり、MHHA*は最適でない。

Algorithm 1 Multi-Heuristic Hybrid A*

Function: MHHA*(s_{start} , s_{goal})

$g(s_{goal}) \leftarrow \infty$;
$g(s_{start}) \leftarrow 0$;
$bp(s_{start}) \leftarrow null$;
$bp(s_{goal}) \leftarrow null$;
$N \leftarrow 0$;
for $i = 0$ to n **do**
  $OPEN_i \leftarrow \emptyset$;
  $CLOSEi \leftarrow \emptyset$;
  insert s^{start} in $OPEN_i$ with $key(s_{start}, i)$ as priority;
end for
while $OPEN_0$ not empty **do**
  for $i = 1$ to n **do**
    if $OPEN_i.Minkey() \leq \omega OPEN_0.Minkey()$ **then**
      $N \leftarrow N + 1$;
      if $g(s_{goal}) \leq OPEN_i.Minkey()$ **then**
        return path pointed by bp(s_{goal});
      end if
      $s \leftarrow OPEN_i.Top()$;
      if $mod(N, setvalue) == 0$ **then**
        RSpath $\leftarrow$ RSCurve(s, s_{goal});
        if RSpath is Collision free **then**
          return RSpath with path pointed by bp(s)
        end if
      end if
      Expandnode(s);
    else
      $N \leftarrow N + 1$;
      if $g(s_{goal}) \leq OPEN_0.Minkey()$ **then**
        return path pointed by bp(s_{goal});
      end if
      $s \leftarrow OPEN_0.Top()$;
      if $mod(N, setvalue) == 0$ **then**
        RSpath $\leftarrow$ RSCurve(s, s_{goal});
        if RSpath is Collision free **then**
          return RSpath with path pointed by bp(s)
        end if
      end if
      Expandnode(s);
    end if
  end for
end while
return NULL

Algorithm 2 The Extension of Parent Node

Function: Expandnode(s)

Remove s from $OPEN_i, \forall i = 0 \dots n$;
Add s to $CLOSE_i, \forall i = 0 \dots n$;
for each $s^{'}$ in Succ(s) **do**
  if $s^{'}$ in $CLOSE_0$ **then**
    Continue;
  else
    $g(s^{'}) \leftarrow g(s) + c(s, s^{'})$;
    $bp(s^{'}) \leftarrow s$;
    insert/update $s^{'}$ in $OPEN_0$ with $key(s^{'}, 0)$ as priority;
    if $s^{'}$ not in any $CLOSE_i$, $\forall i = 1 \dots n$ **then**
      for $i = 1$ to n **do**
        insert/update $s^{'}$ in $OPEN_i$ with key($s^{'}, i$);
      end for
    end if
  end if
end for

Algorithm 3 Calculate The Total Cost of One Node

Function: key(s, i)

return $g(s) + h_i(s)$

tic, we choose the maximum of both, one takes into account the non-holonomic constraints of the vehicle while neglecting the obstacle and another considers the obstacle while neglecting the non-holonomic of the vehicle. For convenience, in this implementation, MHHA* only chooses one inadmissible heuristic and the inadmissible heuristic is obtained by inflating the admissible heuristic.

4.2 Results

In the simulation of this work,the end position is fixed at $(x_f, y_f, \varphi_f, \nu_f) = (-1.35, 1.5, 0, 0)$, the start status is static ($\nu = 0$) and the initial orientation is right. In this work two typical scenarios are selected to check the performance of MHHA*: one is forward entry parking, as shown as Figure 7, the other is backward entry parking, as shown as Figure 8. The obstacle is represented by the red points and the analytic solution is represented by the blue line. The black vertical line is not the obstacle, it is just used to represent the scope of the parking space.The start position in Figure 7 is set in $(x_s, y_s, \varphi_s, \nu_s) = (-9, 8.0, 0, 0)$ and the start position in Figure 8 is set in $(x_s, y_s, \varphi_s, \nu_s) = (12, 8.0, 0, 0)$. The search process of MHHA* is very fast and this algorithm can get a collision-free solution in short time while assuring the sub-optimality.

4.3 Discussion and Comparison

This subsection will compare the MHHA* with Hybrid A*. The simulation result of Hybrid A* in both scenarios can be shown as Figure 9 and Figure 10. And the comparison of the key performance indicators between MHHA* and Hybrid A* is shown in Table 1 and Table 2. We can see MHHA* is better than Hybrid A* in many aspects, such as the number of extended nodes, the number of iterations, and the extension time. Only in the path length aspect the MHHA* is worse than Hybrid A*, because MHHA* makes a trade-off between optimality and computational efficiency, it only ensures sub-optimality. Although the path calculated by MHHA* is not optimal, it is feasible, so it could be used as the initial solution for OBCA. And the paths calculated by MHHA* and Hybrid A* are homotopy that is both solutions will converge to the same solution, so the computational efficiency is more important than optimality, while MHHA* is sub-optimal. So in the H-OBCA algorithm, we can choose MHHA* to generate

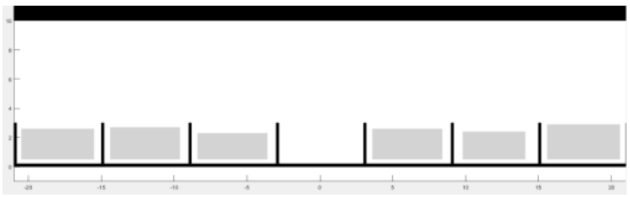

図6:実験駐車場のシナリオ。

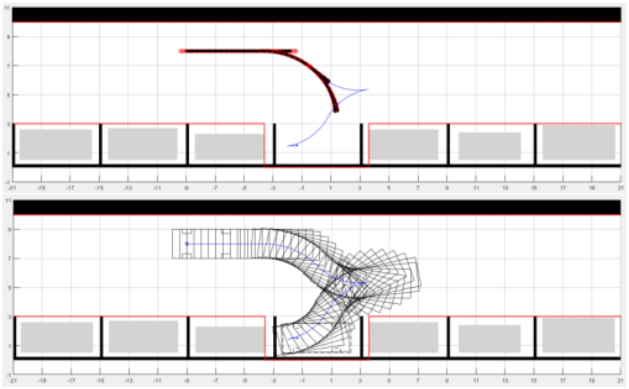

図7:MHHA*を使用した場合の前方駐車。

つまり、H-OBCAアルゴリズムでは、計算効率を向上させるために、初期解を生成するMHHA*を選択することができる。

表1:フォワードパーキングの性能比較

Performance	MHHA*	Hybrid A*
Number of Extended Nodes	273	1460
Number of Iterations	79	564
Extension Time (s)	1.0222	6.9123
Path lengths (m)	21.097	18.659

表2:バックワードパーキングの性能比較

Performance	MHHA*	Hybrid A*
拡張ノード数	253	6361
Number of Iterations	62	2486
Extension Time (s)	0.9753	43.49
Path lengths (m)	18.16321	16.691

5 CONCLUSION

本研究では、自律駐車のための探索ベースの運動計画アルゴリズムを提案した: マルチヒューリスティックハイブリッドA*(MHHA*)は、最適性の保証と高速な計算効率を提供することができる。また、座標変換に基づく新しい衝突チェック法を利用し、計算効率を向上させる。並行駐車を研究対象として選択し、車両モデルとしてシングルトラックモデルを選択する。シミュレーション実験では、2つの典型的なシナリオが選択され、1つは前方駐車、もう1つは後方駐車である。MHHA*は最適ではないが、OBCAで初期解として使用すると、ホモトピーのためハイブリッドA*と同じ解に収束する。

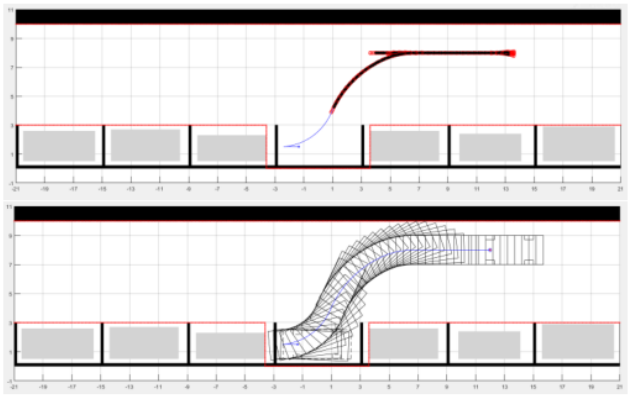

図8:MHHA*を使用した場合の後方駐車。

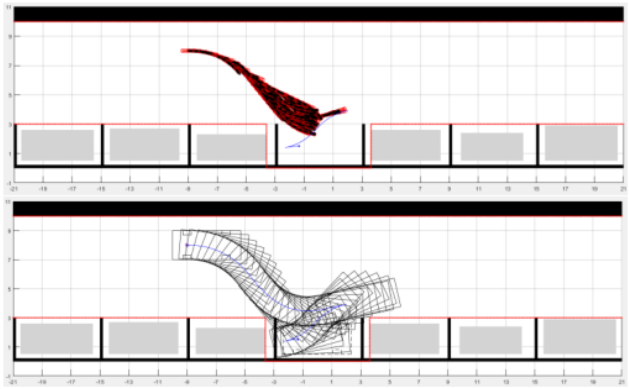

図9:ハイブリッドA*を使用した場合の前方駐車。

さらに、MHHA*は、両シナリオにおいて、拡張ノード数、反復回数、拡張時間において、ハイブリッドA*よりも優れた性能を持つ。今後の研究の方向性としては、H-OBCAにおけるMHHA*の性能を確認することである。

REFERENCES

[1] J. Guo, U. Kurup, and M. Shah, "Is it safe to drive? an overview of factors, metrics, and datasets for driveability assessment in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3135–3151, 2019.

[2] 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, 2015.

[3] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2537–2542.

[4] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.

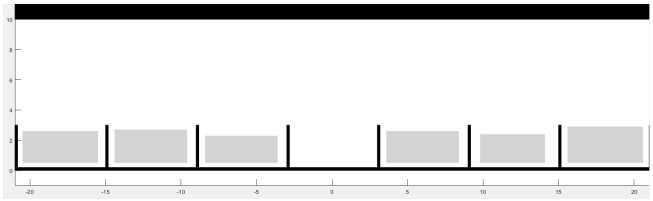

Figure 6: The experiment parking scenario.

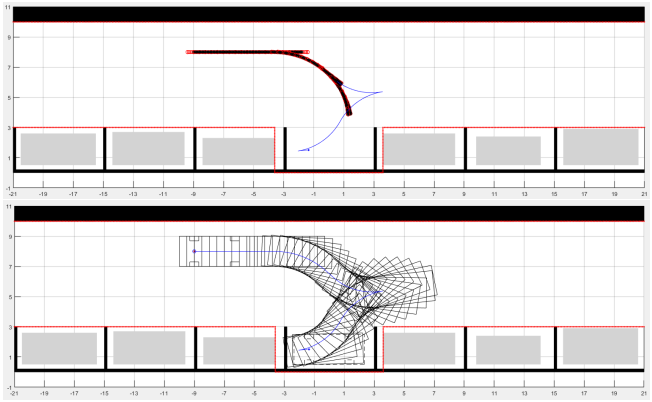

Figure 7: The forward parking of using MHHA*.

the initial solution to improve the computational efficiency.

Table 1: The performance comparison of forward parking

Performance	MHHA*	Hybrid A*
Number of Extended Nodes	273	1460
Number of Iterations	79	564
Extension Time (s)	1.0222	6.9123
Path lengths (m)	21.097	18.659

Table 2: The performance comparison of backward parking

Performance	MHHA*	Hybrid A*
Number of Extended Nodes	253	6361
Number of Iterations	62	2486
Extension Time (s)	0.9753	43.49
Path lengths (m)	18.16321	16.691

5 CONCLUSION

This work proposed a search-based motion planning algorithm for autonomous parking: Multi-Heuristic Hybrid A* (MHHA*), it could provide the guarantee of sub-optimality and fast computational efficiency. And a novel collision check method based on coordinate transformation is utilized to improve computational efficiency. Parallel parking is chosen as the research target, and the single-track model is selected as the vehicle model. In simulation experiments, two typical scenarios are selected, one is forward parking and the other is backward parking. Although MHHA* is not optimal, when used in OBCA as the initial solution, it will converge to the solution as same as Hybrid A* because of homotopy.

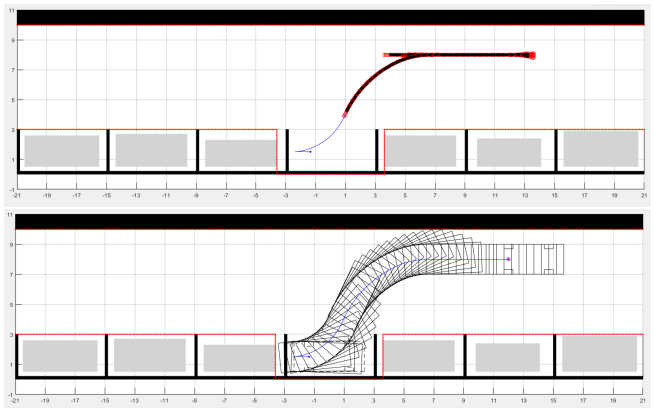

Figure 8: The backward parking of using MHHA*.

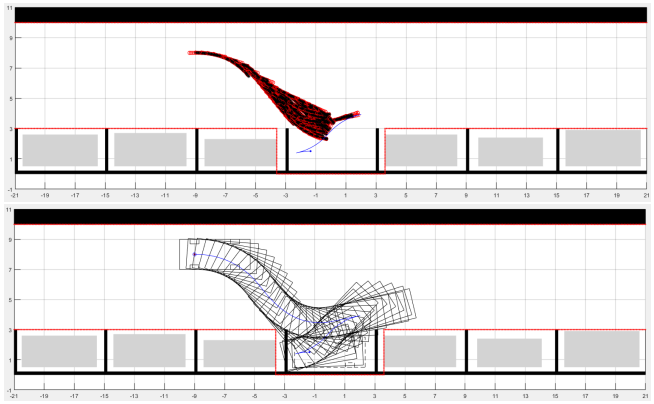

Figure 9: The forward parking of using Hybrid A*.

Moreover, the MHHA* has good performance than Hybrid A* in the number of extended nodes, the number of iterations, and the extension time in both scenarios. Our future research direction is checking the performance of MHHA* in H-OBCA.

REFERENCES

[1] J. Guo, U. Kurup, and M. Shah, "Is it safe to drive? an overview of factors, metrics, and datasets for driveability assessment in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3135–3151, 2019.

[2] 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, 2015.

[3] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2537–2542.

[4] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.

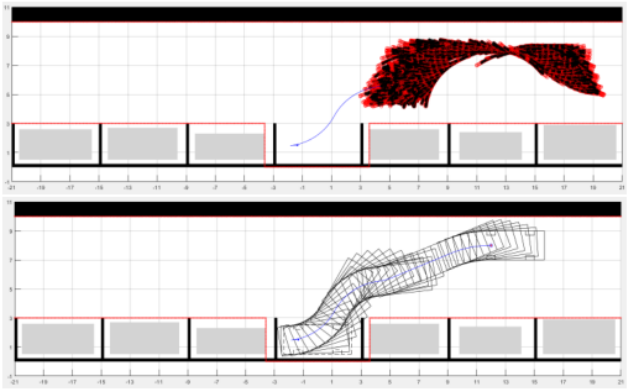

図10:ハイブリッドA*を使用した場合の後方駐車

[5] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[6] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[7] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[8] B. Adabala and Z. Ajanovic, "A multi-heuristic search-based motion planning for autonomous parking," in *30th International Conference on Automated Planning and Scheduling: Planning and Robotics Workshop*, 2020.

[9] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, and X. Peng, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[10] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *2010 IEEE intelligent vehicles symposium*. IEEE, 2010, pp. 518–522.

[11] K. Kurzer, "Path planning in unstructured environments: A real-time hybrid a* implementation for fast and deterministic path generation for the kth research concept vehicle," 2016.

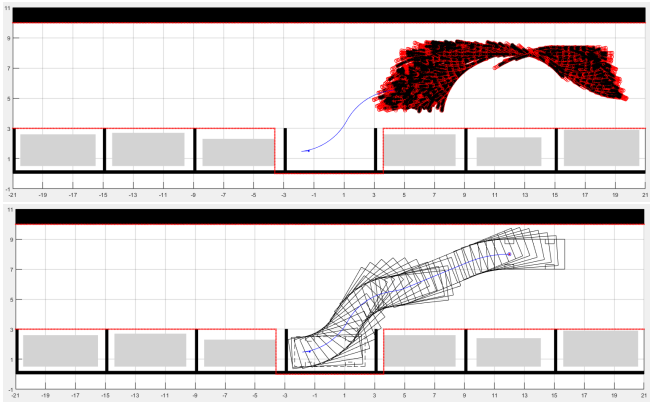

Figure 10: The backward parking of using Hybrid A*

[5] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[6] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[7] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[8] B. Adabala and Z. Ajanovic, "A multi-heuristic search-based motion planning for autonomous parking," in *30th International Conference on Automated Planning and Scheduling: Planning and Robotics Workshop*, 2020.

[9] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, and X. Peng, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[10] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *2010 IEEE intelligent vehicles symposium*. IEEE, 2010, pp. 518–522.

[11] K. Kurzer, "Path planning in unstructured environments: A real-time hybrid a* implementation for fast and deterministic path generation for the kth research concept vehicle," 2016.