車輪型ロボットのためのステアリングセットによる経路探索法

Path Planning with Steering Sets for Car-Like Robots

○鈴木 夢見子 (奈良先端大, 産総研)
加賀美 聡 (産総研, 奈良先端大, CREST, JST)
James J.KUFFNER (産総研, CMU)

Yumiko Suzuki, NARA INSTITUTE of SCIENCE and TECHNOLOGY
Satoshi Kagami, AIST, NAIST, CREST & JST
James J.KUFFNER, AIST & CMU

■Abstract: This paper describes a new path planning method for car-like robots using precalculated steering sets including three trajectory types: straight, right and left. Our planner searches for a shorter path with this steering sets. Path planning using A* needs to check all neighboring cells of a current grid cell, so it takes time. This new planning method needs to only check a collision on the edge of the steering sets where shape is precalculated by a B-Spline curve. Also when A* is used to search a grid map, the path can have sharp changes of direction which are not smooth. A trajectory using such a path is not suitable for a car-like robots. Path planning with steering sets is used to construct trajectories with smooth changes in direction. We made a simulator of the path planner, and estimated suitable parameters to implement steering sets path planning on a real car-like robot. We also checked planning logs from the simulator, and compared it with an A* planner.

Key Words: path planning, steering set, car-like robot

1 緒論

Nilsson¹⁾ によって A*探索が発表されてから今日にいたるまで、その探索結果の最適性によって、探索問題の多くの場面で使われてきた。人工知能分野での探索アルゴリズムとして出発した A*探索のアイディアはグリッドベースの地図を用いてのロボットの経路探索問題の解決のためにも使われている。A*探索による経路探索では、到達ノードの隣接ノードのすべての状態をチェックする必要がある。このため、A*探索で得られる経路は最適性が保証されるが、計算コストは高くなる。

特に車輪型ロボットの経路探索に A*探索を用いた場合、右左折などの進路変更点でも生成される軌道が直線的になるため、滑らかな軌道を得ることができず、車輪型ロボットの経路には適していない。これを解決するために、A*探索から得た経路に対して 3 次多項式を用いて曲線経路を生成し、ハードウェア的に車輪を制御することで、通過経路が滑らかな曲線となるように軌道を制御する手法がある²⁾。しかし、この場合でも経路探索の時点でカーブを生成することはできていない。

探索による経路の滑らかさという点に着目した論文には Field D*がある³⁾。D*は Dynamic A*でありこのアルゴリズムも A*探索が持つグリッドベースによる探索の限界を継承しており、完全に滑らかな経路を生成することはできていない。

“steering behaviors”として、動作の特徴を 1 つの振る舞いとしてとらえた初めての論文は Reynolds⁴⁾ である。この “steering behaviors”の考え方は CG アニメーションの世界で実装され、表現上で滑らかな動きを実現している⁵⁾。この振る舞いを現実世界のロボットに実装することができれば、グリッドベースの探索の限界を超え、変化する環境にも適応できる滑らかな経路を生成し、車輪型ロボットの軌道として実装することができると考えられる。

そこで本手法では車輪の動作の特徴をステアリングセットとして事前に定義し、これを探索木の形状として使用し経路探索する

アルゴリズムを提案する。この探索手法を可視化したシミュレータを作成し、地図の静的なスタートから静的なゴールまでを経路探索させ、シミュレータの出力結果から A*探索とその挙動を比較し、実際の車輪型ロボットに実装できるかを検討した。

2 ステアリングセットによる経路探索

2.1 ステアリングセット

本手法では、車の動作要素である“直進”、“右折前進”、“左折前進”を “straight line”、“right curve”、“left curve”として、3 つの形状を 1 つのセットとし、この離散的なセットを探索木として用いた。また “right curve”、“left curve”の曲線形状はパラメトリック曲線表現の 1 つで接続性にすぐれた B-Spline 曲線によって決定した。

シミュレータへの初期パラメータとして、直進時の長さや右左折方向への展開角度を可変で与えカーブ形状を事前に算出した。

Fig. 1 Figure of Steering Sets

2.2 B-Spline 曲線

自由曲線の表現は、これまでに各種の数学的表現方法が提案され 3 次元 CAD や CG で使用されているが⁶⁾、本手法では、パラメトリック曲線の 1 つである B-Spline 曲線を用いてステアリングセットのカーブ形状を定義した。

B-Spline 曲線は制御点と呼ばれる 4 つ以上の位置ベクトルから、各区間で 3 次多項式で、全区間で 2 階の導関数まで連続な関数 $S_i(x)$ である。制御点が $(x_i, f(x_i))$ ($i = 0, \dots, n$) のとき、以下の式で与えられる。

$$S_i(x) = M_{i-1} \frac{(x_i - x)^3}{6h_i} + M_i \frac{(x - x_i)^3}{6h_i} + \left(f(x_{i-1}) - M_{i-1} \frac{h_i^2}{6} \right) \frac{(x_i - x)}{h_i} + \left(f(x_i) - M_i \frac{h_i^2}{6} \right) \frac{(x - x_i)}{h_i} \quad (1)$$

ただし

$$h_i = x_i - x_{i-1} \\ f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h_i} \\ M_i = \frac{f'(x_{i+1}) - f'(x_i)}{x_{i+1} - x_{i-1}} \\ h(0) = 0, M_0 = M_n = 0$$

式 (1) をシミュレータに実装し、初期パラメータとして直進で 1 回に進む距離を元に 5 点の制御点を任意に選択しカーブ形状を決定した。

2.3 探索と経路生成

A*探索における現在ノードからゴールノードまでのヒューリスティックコストにはマンハッタン距離やユークリッド距離が使われることが多い。本手法におけるヒューリスティックコスト計算についても、ユークリッド距離を用いた。ステアリングセットの形状を用いて、固定スタートから固定ゴールにたどり着くまでステアリングセットの探索木を伸ばしていく。各ノードは位置情報およびノードコストと親ノードへのポイントから構成されている。ノードが展開できる場合、このノード情報を経路候補としてヒープに保持し、また次展開ノード候補として Open リストに追加する。各ノードのコストはスタートから現在ノードまでの経過コスト、現在グリッドの通過コスト、各ステアリングセットのウェイトコストから生成される。Open リストから最小コストのノードを次のステアリングセット展開ノードとして取り出し、上記処理を繰り返す。ゴール位置に到達した時点でステアリングセットの伸展をとめ、ゴールノードからスタートノードに到達するまでヒープをたどることで経路を取得する。

上記の処理の流れを以下に示す。

```
function Planning-Grown-Tree
L1 Initialize
L2 GrownTree
L3 GetTrajectory
```

```
function Initialize
```

```
L1 Precalculate Curvefigure
L2 Get ArmLength,WideAngle
L3 Get Start and Goal Position
L4 Do Obstacle Grown
```

```
function GrownTree
```

```
L1 Loop reached goal
L2 Get Node in Heap
L3 Check Collision on Map
L4 if (FREE) Calculate Cost
L5
L6 Check Unexplored
L7 CheckCollision on Edge
L8 Put New Node in Heap
L9 Put New Node in Open List
```

```
function GetTrajectory
```

```
L1 Get Goal Node
L2 Loop reached start
L3 Get Predecessor Node
```

2.4 障害物衝突チェック

ステアリングセットを伸ばし、それを経路として選択するためにはステアリングセットが通過する地図上に障害物がないことを確認する必要がある。このためにステアリングセットが通過する軌道上の障害物チェックを行う。

障害物チェックは、地図のスケールであるグリッド単位で行い、事前に作成したステアリングセットの直進時の長さの整数値を分割数とした。各ノードからのステアリングカーブ上の障害物チェックは事前に作成したステアリングカーブの座標情報を座標変換によって現在ノード座標に合わせ該当する座標の地図情報をチェックした。

Fig. 2 Experimental Map Image

Fig. 3 Experimental Map Image after Obstacle Grown

2.5 進入不可領域と通過方向フラグの設定

今回シミュレータで使用した地図は、室内の座標位置をグリッド単位で保持しており、1 グリッドのスケールは 1 辺が実際の 10cm 四方に相当する。地図の構造として 1 グリッド単位に位置、地図コスト、通過方向チェックフラグの情報を持たせた。

本手法は、車輪型ロボットに実装することを想定しているため、地図情報を読み込んだ際、実装予定の車輪型ロボットサイズの半径長で室内の壁、障害物の周辺を進入不可領域とした。これを前処理として地図情報の地図コストに設定した後、経路探索を行った。

また、A*探索のように隣接グリッドをすべて探索しないために、1つのグリッド上を複数回通過する場合がある。この通過方向を15度単位で24方向に分割し、通過方向が同じ場合はその探索枝は伸展させないこととした。

3 シミュレータの実装

探索結果を可視化するシミュレータを作成し、提案手法をこのシミュレータ上に実装した。ここではシミュレータの環境およびその出力結果を示す。

3.1 環境

本手法はC++を用いて実装し、結果の視覚化にはOpenGLを使用した。ハードウェアについては、CPUはIntel Pentium M processor 1.70GHz, RAMは1.00GB, OSはLinux(Debian) 2.6の環境にて実行した。ソフトウェア構成は、本手法を探索クラスとして、その他の機能を表示クラス、ナビゲーションクラス、ヒープ管理クラスとして実装した。出力は、伸展したステアリングセットの軌跡と取得したゴールまでの経路を画面出力し、探索処理反復数、ノード数、探索時間、経路長、進入不可領域の設定時間、ステアリングカーブ数をログ出力可能にした。

シミュレータには直進時の長さ、スタート時の方向、直進ラインとカーブセット間の広角、ゴール半径、直進ラインの距離とカーブセットの重み、の5つのパラメータを与えることができる。これらのパラメータのうち今回はゴール半径を実装予定ロボットの半径と同じサイズにし固定とした。地図は実験予定の部屋の地図を使った(図2)。

3.2 出力結果

直進ラインの長さ10の場合で直進ラインとカーブセットの重みを変化させて、探索結果を出力した。

直進ラインの重み1に対してカーブセットの重みを3,5,10,15,30,50,300と変化した。その結果、処理時間はどの場合でも1秒以下だった。カーブセットのコストを重くすることで、直進ラインを選択しやすくなった場合、重みをかけない場合と比べて経路長が短くなったが、重みの効果に比例関係は見られなかった。

スタート時の直進ラインの方向をいくつかの方向に向けた場合では、今回使用した地図ではスタート時の直進ラインの方向をサブゴール方向に向けると繰り返しの処理回数が減り、経路長も短くなっていた。

3.3 A*探索との比較

図2でA*探索⁷⁾についても結果を出力した。A*探索によって生成された経路では、進路変更の際の不連続点が生じており、今回の地図の場合では、49箇所発生していた。本手法では、事前に

Fig. 4 Curve Weight 1

Fig. 5 Curve Weight 3

Fig. 6 Curve Weight 5

Fig. 7 Curve Weight 10

Fig. 8 Curve Weight 15

Fig. 9 Curve Weight 30

Fig. 10 Curve Weight 50

Fig. 11 Curve Weight 300

定義したカーブを使用するため、進路変更の際の A*探索のような不連続点は生じていない。

Fig. 12 Results of A* Planner

4 結論

ステアリングセットをつかった車輪ロボットに適した経路探索法を提案した。シミュレータを実装し、パラメータを変化させ得られた経路を出力した。A*探索の場合には、進路変更時の不連続点が生じるために、A*探索による経路探索では滑らかな曲線を描くことができないが、本手法を用いた経路探索では、事前に定義したステアリングセットのカーブを用いて進路変更の経路を得ることが可能である。このため A*探索の場合に生じるような不連続点を生じることなく、滑らかな経路を生成することができる。実際のロボットに実装した場合においても、A*探索による経路探索と比較して、滑らかな経路を得ることができると予想される。

今後の課題として、ゴールへの最短経路を得るために、現在固定のステアリングセットを使用している点を、地図状況に適したステアリングセットを探索木として選択できるようアルゴリズムを拡張する予定である。また実際の車輪型ロボットに実装し、挙動を検証する。

参考文献

- (1) N. Nilsson, Principles of Artificial Intelligence. Tioga Publishing Company, 1980.
- (2) Simon Thompson and Satoshi Kagami. Smooth Trajectory Planning with Obstacle Avoidance for Car-Like Mobile Robots, The23rd Annual Conference of the Robotics Society of Japan, 2005.
- (3) D.Ferguson and A.Stentz. The Field D* Algorithm for Improved Path Planning and Replanning in Uniform and Non-Uniform Cost Environments, CMU-RI-TR-05-19, Robotics Institute, CMU, June, 2005.
- (4) Craig. W. Reynolds. Steering Behavior For Autonomouse Characters. in: Game Developers Conference 1999, 1999.
- (5) Jared Go, Thuc Vu and James J. Kuffner. Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2004, 2004.
- (6) Gerald E.Farin. Nurbs: From Projective Geometry to Practical Use. A K Peters Ltd, 1999.
- (7) James J.Kuffner. Efficient Optimal Search of Euclidean-Cost Grids and Lattices. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IEEE, September, 2004.