ポテンシャル関数を用いるT-RRTによる経路探索

○株丹 亮 (九州工業大学)　西田 健 (九州工業大学)

1. はじめに

現在までに多くの経路探索の手法が提案されており，それらは対象空間における所与のノードを基準する手法と連続空間を対象とする手法に大別できる．前者には，ダイクストラ法やA*[1]があり，後者には，ポテンシャル法[2]，PRM（probabilistic road map）[3]，RRT（rapidly-exploring random tree）[4]などがある．自律走行ロボットや垂直多関節ロボットの経路探索を考える場合，稼働領域に障害物や他者が存在する状況や動的に環境が変化する状況では規定のノードを与えることが困難であるため，後者の手法の適用が有利である．

後者に属する手法の中でも初期に提案されたポテンシャル法は，目標からの引力と障害物からの斥力を表現するポテンシャル関数を生成し，その関数について最急降下勾配法を適用することで経路を生成する．引力と斥力の拮抗点ではデッドロックが発生するという問題点があるが，周囲の状況や対象物の種類に応じた柔軟な評価関数を構成できるという利点を有する[5]．PRMとRRTは，ランダムサンプリングの利用により，演算の高速化を目指した手法である．特に，RRTは前処理を必要とせず，局所解の回避能力が高いことから，高次元空間における探索に適用可能である．また，様々な条件下での経路計画に適用できるように拡張された複数の修正手法が提案されている[6, 7, 8, 11]．中でも，状況に応じて設計可能な評価関数を経路探索の手順に導入したT-RRT（Transition-based RRT）[9]は高い汎用性を有する．

一方，上述の経路探索手法はいずれも万能ではなく，適用する対象や環境に応じて適切に選択し調整する必要がある．例えば，最短距離を最速で移動することが重要な場合もあれば，ロボットの運動能力に応じた安全性が重要な場合もある．ロボットに搭載されるセンサの信頼性も考慮する必要があるかもしれない．さらに，自身だけではなく，障害物や他者の動作に注意する必要がある状況も考えられる．そこで本論文では，できるだけ少ないパラメータ調整で種々の状況に対応可能な経路探索手法を志向したT-RRTの構成方法を提案する．現在までのT-RRTに関する先行研究では，実環境における高低差や対象物からの距離などの簡便なパラメータが評価関数に利用されており，その具体的な設計手法が議論されていない[9, 10, 11]．そこで本研究では，T-RRTの評価関数部にポテンシャル法を導入し，さらに生成された経路の洗練化手法までを体系化して評価する．さらに，生成された軌道の安全性を評価する指標を用い，二次元平面における数値シミュレーションと産業用ロボットの軌道生成シミュレーションにより，従来手法と性能を比較する．

2. T-RRT

2.1 アルゴリズム

T-RRTの処理をAlgorithm 1に示す．まず1行目で探索木τを初期化するために，開始位置$\boldsymbol{x}^s$，目標位置$\boldsymbol{x}^g$および探索エリアを初期条件として与える．2行目では，探索開始前に評価関数としてCostFn($\boldsymbol{x}$)を与える．3行目から探索ループに入り，4行目では，探索エリア内で障害物に含まれないランダムな点$\boldsymbol{x}^{sample}$を設定する．5行目では，探索木τの中で$\boldsymbol{x}^{sample}$に最も近い点$\boldsymbol{x}^{near}$を探索する．また，$\boldsymbol{x}^{near}$から$\boldsymbol{x}^{sample}$に対して距離をεだけ伸ばした点を$\boldsymbol{x}^{new}$とする．ただし$|\boldsymbol{x}^{sample}-\boldsymbol{x}^{near}|<\varepsilon$となる場合には，$\boldsymbol{x}^{sample}$を$\boldsymbol{x}^{new}$とする．6行目では，$\boldsymbol{x}^{near}$から$\boldsymbol{x}^{new}$までの線分が障害物と干渉していなければ$\boldsymbol{x}^{new}$として探索木$\tau$に登録し，$\boldsymbol{x}^{near}$と$\boldsymbol{x}^{new}$の接続情報を保存する．干渉していれば，$\boldsymbol{x}^{new}$を破棄する．7行目では，TransitionTest関数により，新たなノード$\boldsymbol{x}^{new}$の登録の判定を行う．10行目では，ゴール座標に到達しているかを確認する．CheckGoal関数は，$\boldsymbol{x}^{new}$と目標位置$\boldsymbol{x}^g$の偏差がしきい値を下回れば$True$を返す関数である．$True$が返された場合には，$\boldsymbol{x}^{new}$と$\boldsymbol{x}^g$を差分の長さで完全に接続し処理を終了する．

以上の処理手順がRRTと異なるのは2行目と7行目である．2行目のCostFn($\boldsymbol{x}$)の詳細は後述する．7行目のTransitionTest関数の詳細をAlgorithm 2に示す．この関数では，1行目と2行目で親ノード$\boldsymbol{x}^{parent}$，子ノード$\boldsymbol{x}^{child}$に関する評価を算出する．3行目でそれらの評価値を比較し，$ChildCost > ParentCost$の場合は，

$$\Delta C = \frac{ChildCost - ParentCost}{distance} \tag{1}$$

を算出する．親ノードと子ノードの評価値の差と，ノード間距離に応じて定まるΔCを用いて，それらのノードの組み合わせの採択を次式の確率関数pによって判

Algorithm 1: T-RRT Algorithm

```
1  τ.init(x_start);
2  Define cost function : CostFn(x);
3  while GoalReached = False and
         iterations < MAX ITERATIONS do
4      randSample(x^sample);
5      x^near ⟵ NearestNode(x^sample);
6      x^new ⟵ GenNewNode(x^near, x^sample);
7      if x^new ≠ NULL and
       TransitionTest(x^new, x^near) = True then
8          τ ⟵ AddNode(x^new);
9          τ ⟵ AddEdge(x^near, x^new);
10         if CheckGoal(x^new) = True then
11             GoalReached ⟵ True;
```

Algorithm 2: TransitionTest($\boldsymbol{x}^{child}$, $\boldsymbol{x}^{parent}$)

1 $ChildCost \longleftarrow \mathrm{CostFn}(\boldsymbol{x}^{child})$;
2 $ParentCost \longleftarrow \mathrm{CostFn}(\boldsymbol{x}^{parent})$;
3 **if** $ChildCost \leq ParentCost$ **then**
4 　**return** $True$;
5 $\Delta C \longleftarrow \dfrac{ChildCost - ParentCost}{distance}$;
6 **if** $\Delta C > 0$ **then**
7 　$TransitionProbability \longleftarrow \exp\left(-\dfrac{\Delta C}{K \cdot T}\right)$;
8 **else**
9 　$TransitionProbability \longleftarrow 1.0$;
10 **if** $Rand(0,\ 1) \leq TransitionProbability$ **then**
11 　$T \longleftarrow T/\alpha$;
12 　$Failed \longleftarrow 0$;
13 　**return** $True$;
14 **else**
15 　**if** $Failed \geq Failed_{max}$ **then**
16 　　$T \longleftarrow T \cdot \alpha$;
17 　　$Failed \longleftarrow 0$;
18 　**else**
19 　　$Failed \longleftarrow Failed + 1$;
20 　**return** $False$;

定する．

$$p = \begin{cases} \exp\left(-\dfrac{\Delta C}{K \cdot T}\right) & \text{if} \quad \Delta C > 0 \\ 1.0 & \text{otherwise} \end{cases} \tag{2}$$

この関数の概形を図 1 に示す．ここで $K > 0$ は確率 p の調整用の定数であり，開始位置と目標位置の評価値の平均値で与える．T は TransitionTest 関数の実行の度に調整される変数であり，確率 p をループごとに遷移させる役割を持つ．$ChildCost > ParentCost$ の場合，確率 p ($\Delta C > 0$) を利用し判定を行うフェーズに入る（Algorithm 2，10 行目）．確率の範囲内であれば，注目する二つのノード間の経路が採択される．このとき同時に，T を α ($\alpha > 0$) で除することで T の値を減少させる．この操作により，次回のループで TransitionTest 関数が実行され同じフェーズに入った場合に，確率 p の値が低くなるので，注目する二つのノード間の経路が採択される確率が低くなる．二つのノード間の経路が採択されない場合が一定回数以上続いた場合，すなわち Algorithm 2 の変数 $Failed$ によるカウントがある一定値以上に増加した場合には，変数 $Failed$ を初期化し，T を α との乗算によって値を再び増加させる．

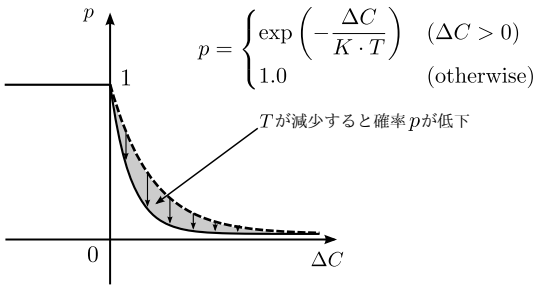

図 1　確率関数 p の概形

2.2　T-RRT の問題点と提案手法

本研究では，斥力ポテンシャルと引力ポテンシャルの合成によって構成されるポテンシャル場 [1] を評価関数 $\mathrm{CostFn}(\boldsymbol{x})$ として利用する．斥力ポテンシャルは障害物に接近するほど高い値を示すため，障害物との距離を考慮する経路探索を可能にする．さらに引力ポテンシャルによって，開始座標から目標座標に向かう評価値の下降を定義することができる．これにより，目標座標に向かう経路が生成されると同時に，評価の比較において $ChildCost > ParentCost$ となる場合の確率的な判定手順を減らすことできる．

T-RRT では，評価関数に基づいた軌道生成が行われると同時に，評価値が低い場所から高い場所への遷移確率を TransitionTest 関数によって発生させることで，局所解回避性能を向上させる．一方で，ノード選択に判定条件を付加したことにより，RRT よりも実行時間が大幅に増加するという問題が発生する [10]．さらに，高速演算のためにランダムサンプリングを用いて経路を生成するため，連続的な勾配計算に基づくポテンシャル法よりも経路の総距離が増加する傾向がある．これに対して，T-RRT の評価関数にポテンシャル場を利用すると，目的座標への引き込みの強さ調整可能になるため，無駄なノード判定の増加と経路の総距離の増加傾向を抑制できる．

2.3　ポテンシャル場

ここでは簡単のため，2 次元平面における経路探索を例に挙げて，ポテンシャル場の構成法を示す．以下に，ポテンシャル場の定義に用いるポテンシャル関数を示す．

$$P_g(\boldsymbol{x}_k) = K_g\left\{(x_k - x^g)^2 + (y_k - y^g)^2\right\} \tag{3}$$

$$P_o(\boldsymbol{x}_k) = K_o \exp\{-r_1(x_k - x^o)^2 - r_2(y_k - y^o)^2\} \tag{4}$$

ここで $\boldsymbol{x}_k \equiv [x_k \quad y_k]^{\mathrm{T}}$ は現在の位置，$\boldsymbol{x}^g \equiv [x^g \quad y^g]^{\mathrm{T}}$ は目標位置，$\boldsymbol{x}^o \equiv [x^o \quad y^o]^{\mathrm{T}}$ は障害物の位置である．また，式（3）の $P_g(\boldsymbol{x}_k)$ は引力ポテンシャル，式（4）の $P_o(\boldsymbol{x}_k)$ は斥力ポテンシャルを表す．さらに，K_g は目標座標に引き込む強さ調整するパラメータ，K_o は斥力ポテンシャルの大きさ，r_1，r_2 は斥力ポテンシャルの広がりを表す．障害物が発生する斥力ポテンシャルの合成 $P(\boldsymbol{x}_k)$ は次式で与える．

$$P(\boldsymbol{x}_k) = P_g(\boldsymbol{x}_k) + \sum_{i \in N} P_{o,i}(\boldsymbol{x}_k) \tag{5}$$

ここで N は障害物の数である．

2.4　経路の洗練

探索経路の洗練処理の手順を以下に示し，概要を図 2 に示す．

(1) 処理前の経路の総距離を計算する．
(2) 経由点データの要素から隣接しない 2 点をランダムに抽出する．経由点の要素数が 2 の場合は処理を終了する．

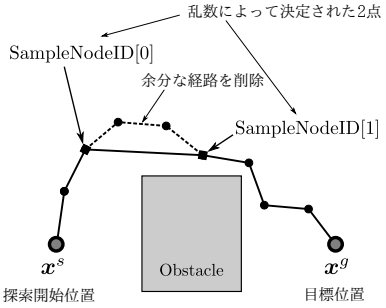

図 2　経路の洗練処理

(3) 選択された 2 点を結ぶ線分と障害物との干渉を確認する．干渉していなければ，2 点間に存在する経由点を削除する．
(4) 削除前の総距離と削除後の総距離を比較し，手順 (4) の処理によって経路が短縮されたか評価する．
(5) 経路の総距離に変化がない場合は処理を終了する．減少した場合は (2) に戻る．

3.　シミュレーション

3.1　経路の安全性の評価指標

経路の安全性は以下の指標により評価できる [9]．

$$W = \sum_{i \in i_+} \{c(\boldsymbol{x}_i) - c(\boldsymbol{x}_{i-1})\}\, d_i + \epsilon \sum_{i \in i} d_i \qquad (6)$$

ここで，i は経路のノード番号，i_+ は隣接したノードの評価値の差分が $c(\boldsymbol{x}_i) - c(\boldsymbol{x}_{i-1}) > 0$ の場合の i の集合，d_i はノード $\boldsymbol{x}_i$，$\boldsymbol{x}_{i-1}$ 間の距離，ϵ は第一項と第二項の重みを調整する定数である．評価関数の値が小さいノードから大きいノードへ遷移する場合に，式 (6) の右辺の第一項により W の値が増加する．一方，評価値が減少するように遷移した経路と総距離が短い経路について W の値は低くなる．本研究では評価関数にポテンシャル場を用いるので，W の値が低い経路は安全性が高いと判断できる．

3.2　二次元平面における経路探索

RRT と T-RRT，およびそれらに経路の洗練処理を追加した四通りの手法により経路探索のシミュレーションを行った．各手法の実行には乱数発生を伴うため，それぞれ 10 回ずつ探索を実行し評価を行った．それらの探索経路の例を図 3 と図 4 に示す．探索経路の評価は，経路の総距離，実行時間，経路の評価値の最大値 C_{max}，平均値 C_{ave}，総和 C_{sum}，W および分散 σ について評価した．評価結果を表 1 に示す．比較のために，理論的な最適経路とポテンシャル法による探索経路の評価結果も同時に示した．表 1 より，RRT よりも T-RRT の C_{max}，C_{ave}，C_{sum} の値が小さいことがわかる．また，T-RRT は W の値を RRT より小さくできていることもわかる．さらに，経路の洗練処理により，実行時間以外の評価項目について性能の向上が確認された．経路の洗練を行う T-RRT により生成された経路が，ポテンシャル法による探索経路の評価値に最も近いという結果が得られた．一方で，洗練処理を付加した T-RRT は，探索時間が最長であった．

3.3　垂直多関節ロボットの手先軌道生成

探索領域を三次元に拡張して垂直多関節ロボットの手先の軌道計画に提案手法を適用した．すなわち，図 5 に示すようにロボットの前方に柱の障害物が存在する状況を仮定し，動作開始位置から目標位置に到達するまでの手先の軌道を生成する課題のシミュレーションを行った．この課題について，前述の 2 次元平面におけるシミュレーションと同様の評価と比較を行った．この評価結果を表 2 に示す．これより，C_{max}，C_{ave}，C_{sum}，W，σ の値について，T-RRT は RRT よりも性能が良いことがわかった．RRT に経路の洗練処理を付加した場合，経路の総距離は改善するが，他の評価項目については性能が低下する傾向が見られた．一方，T-RRT に経路の洗練処理を付加した場合には，経路の総距離，C_{sum}，W に関して改善が見られた．ポテンシャル法に関しては，次元数増加に伴う勾配計算のコスト上昇により，実行時間が最も長くなった．W の値については，ポテンシャル法の 0.01032 が最小であった．これはポテンシャル法に用いた最急降下法が常にコストが低い値を探索する手法であることに起因する．

3 次元空間における経路探索では，T-RRT に洗練処理を付加した手法が，ポテンシャル法と比べて経路の総距離と実行時間を抑えながら，RRT と比較してコストに関する評価項目を小さくすることが示された．

4.　おわりに

本研究では，T-RRT の評価関数としてポテンシャル場を用いる手法を提案した．さらに，探索経路の洗練化手法を併用することで，安全性に着目した経路生成が可能になることを示した．さらに，探索経路の総距離や，式（6）で示した安全性の評価について，ポテンシャル法や RRT と比較した．その結果，T-RRT は実

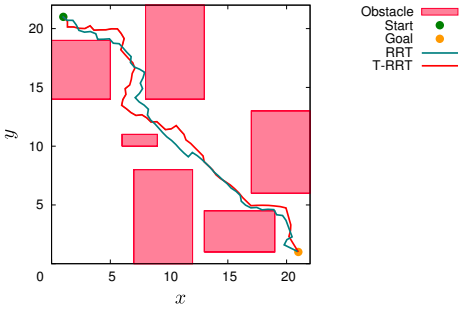

図 3　RRT と T-RRT によって得られた経路の比較

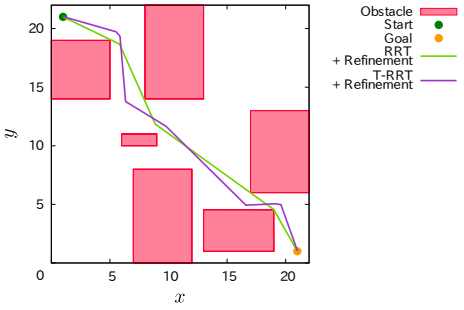

図 4 RRT と T-RRT に経路の洗練処理を加えた経路の比較

表 1　RRT と T-RRT の比較

	Length	C_{max}	C_{ave}	C_{sum}	W	σ	Time[s]
Minimum total distance path	25.17	1.352	0.3944	20.12	2.364	0.4430	—
RRT	34.45	1.413	0.4332	29.95	2.990	0.4472	4.655×10^{-3}
RRT + Refinement	**29.77**	1.268	0.3542	21.35	2.170	0.3840	8.350×10^{-3}
T-RRT	36.06	0.3470	0.1298	9.405	1.020	0.1119	5.604×10^{-2}
T-RRT + Refinement	**32.26**	0.3467	0.1137	7.397	**0.5716**	0.09838	$\mathbf{9.742 \times 10^{-2}}$
Potential field methods	32.86	0.4205	0.09889	6.527	**0.4392**	0.1064	2.678×10^{-2}

表 2　3 次元空間での比較

	Length	C_{max}	C_{ave}	C_{sum}	W	σ	Time[s]
Minimum total distance path	7.661	194.0	70.73	5446	14.99	68.60	—
RRT	9.909	93.87	38.19	3757	6.362	29.99	8.530×10^{-3}
RRT + Refinement	**8.070**	163.1	58.69	4729	11.78	56.01	1.218×10^{-2}
T-RRT	10.46	58.64	26.81	2814	2.189	19.89	1.612×10^{-2}
T-RRT + Refinement	**8.406**	58.85	29.50	2492	**1.490**	21.85	$\mathbf{2.507 \times 10^{-2}}$
Potential field methods	10.32	58.02	24.23	2520	**0.01032**	17.23	$\mathbf{5.054 \times 10^{-2}}$

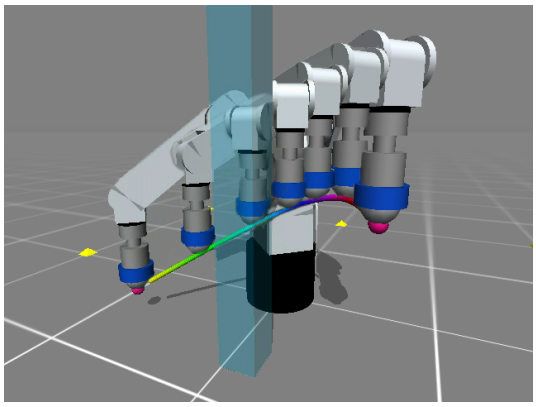

図 5　T-RRT によって生成した軌道

行時間が RRT よりも増加する傾向があるが，経路の安全性は向上することが確認された．また，経路の洗練処理は RRT，T-RRT に対して総距離に関する評価値を小さくする効果があることが確認された．さらに，経路の洗練処理を T-RRT に加えても，安全性に関する評価項目を維持もしくは向上させつつ，経路の総距離を低減できることが確認された．また，T-RRT に経路の洗練処理を付加する手法は，二次元空間の探索問題においてはポテンシャル法よりも実行時間が長くかかったが，三次元空間ではポテンシャル法よりも高速な探索が可能であることが確認された．

探索経路の安全性の評価について，ポテンシャル法は優れた性質を有するが，探索次元の増加に伴う実行時間の増加や，デッドロックの発生リスクの増加が問題となるが，T-RRT とポテンシャル法を組み合わせた提案手法は，それらの問題を回避することができることが見出された．

今後の課題として，ロボットアームの軌道生成を関節空間で行う場合など，より高次元空間での経路探索を高効率に行うための手法の拡張が挙げられる．

参　考　文　献

[1] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, “Principles of Robot Motion: Theory, Algorithms, and Implementations,” Cambridge: MIT Press, 2005.

[2] Rimon, E.; Koditschek, D.E., “Exact robot navigation using artificial potential functions,” Robotics and Automation, IEEE Transactions on , vol. 8, no. 5, pp. 501–518, 1992.

[3] L. E. Kavraki, P. Svestka, J. -C. Latombe, M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” IEEE Trans. on Robotics and Automation, Vol. 12, No. 4, pp. 566–580, 1996.

[4] S. M. LaValle, J. J. Kuffner, “Rapidly-exploring random trees: Progress and prospects,” in Algorithmic and Computational Robotics: New Directions, B. R. Donald, K. M. Lynch, and D. Rus, Eds. Wellesley, MA: A. K. Peters, pp. 293–308, 2001.

[5] 佐藤, “極小点のないポテンシャル場を用いたロボットの動作計画”, 日本ロボット学会誌, Vol. 11, No. 5, pp. 702–709, 1993.

[6] Kuffner, J.J.; LaValle, S.M., “RRT-connect: An efficient approach to single-query path planning,” Robotics and Automation, 2000. Proceedings. ICRA ’00. IEEE International Conference on, vol.2, pp. 995–1001, 2000.

[7] D. Berenson, S. S. Srinivasa, D. Ferguson, J. J. Kuffner, “Manipulation planning on constraint manifolds,” IEEE Int. Conf. on Robotics and Automation, pp. 625–632, 2009.

[8] 坂原, 升谷, 宮崎, “時空間 RRT による複数移動障害物を考慮したリアルタイム軌道生成”, 計測自動制御学会論文集, Vol. 43, No. 4, pp. 227–284, 2007.

[9] L. Jaillet, J. Cortés, T. Siméon, “Transition-based RRT for path planning in continuous cost spaces,” Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., pp. 2145–2150, 2008.

[10] L. Jaillet, J. Cortés, T. Siméon, “Sampling-Based Path Planning on Configuration-Space Costmaps,” IEEE Trans. on, Robotics, Vol.26, No.4, pp. 635–646, 2010.

[11] D. Devaurs, T. Simeon, J. Cortes, “A multi-tree extension of the transition-based RRT: Application to ordering-and-pathfinding problems in continuous cost spaces,” Proc. of IROS, pp. 2991–2996, 2014.