価値反復を用いた移動ロボットによる屋外ナビゲーション

Outdoor navigation with a mobile robot using value iteration

○学　登内 リオン（千葉工大）　正　林原 靖男　(千葉工大)
正　上田 隆一（千葉工大）

Leon TONOUCHI, Chiba Institute of Technology, s20c1078un@s.chibakoudai.jp
Ryuichi UEDA, Chiba Institute of Technology
Yasuo HAYASHIBARA, Chiba Institute of Technology

The authors used a real-time path planning algorithm with value iteration on an outdoor mobile robot and evaluated the computational complexity. Although the algorithm can theoretically obtain an optimal path, it is computationally expensive, so the authors investigated whether the algorithm can be applied to a real robot in a real outdoor environment with a free space of 3714.98[m^2]. For the evaluation, the resolution of the map used for value iteration was set to 0.15[m/pixel], a computer with an Intel Core i7-11800H was used as the computing machine, and a Raspberry Pi Cat was used as the aircraft, and it was confirmed that the calculation was completed in 84.6[s].

***Key Words*:** Autonomous mobile robots, Navigation, route planning generation

1　緒言

価値反復は，最適制御理論を用いて，有限マルコフ決定過程における最適方策を正確に求めることができる手法である [1]．自律移動ロボットが実環境で走行することを考えると, 変化する目的地や経路に対して即座に行動計画を必要となる．例えば, 運搬用の自律移動ロボットを考えると, 目的地の到達したから即座に次の目的地への行動の計画が必要である．さらに, 安定した行動を計画しなければ, 現場にいる同様のロボットや動く人に対応することが難しい．

価値反復を移動ロボットの行動計画に適用するとロボットの様々な状態に対して最適な行動を計算することができる [2]．しかし, 実際のコスト（距離や時間）と事前情報から予測されるコストからひとつの経路を求めることを主な目的とした A*[3] よりも計算量が大きくなってしまう．

一方, 上田らは価値反復を用いたナビゲーションは限られた環境の広さであれば利用できるのではないかと考え ROS パッケージを実装した [4]．このパッケージを用いると, ロボットが走行中, 常に価値反復を行うため, 計算量が多いが

- 決められた行動の中で最適な経路を導出
- ロボットの様々な状態（位置や向き）での最適な行動を選択

することができるでの, 環境のあらゆる地点で最適な行動を計算可能である. そして 100[m^2] の広さのシミュレータ環境で 2 秒の計算コストで行動計画が可能であることが確認されている．

そこで本稿では, 価値反復を用いた実時間経路計画アルゴリズムを屋外移動ロボットで用い, 屋外の実環境を走行させ計算量を評価する．このアルゴリズムを用いると理論上, 最適な経路を得ることができるが計算量が大きいため, 実機に適用できるか調査する．

2　実験

今回の実験では上田らが開発した ROS パッケージ（https://github.com/ryuichiueda/value_iteration）[4] を実装し実験を行った．本研究では千葉工業大学津田沼キャンパス内でロボット（Raspberry Pi Cat）を走行させ計算量を測定する．地図のスタート地点から設定したゴール地点まで 10 回走行させ, 価値反復の計算量を計測した．

計算機には CPU として Intel Core i7-11800H（8 コア 16 スレッド）, DRAM として DDR4-3200 64GB を用いた．自己位置推定には上田らが作成した ROS の emcl パッケージ（https://github.com/ryuichiueda/emcl）[5], 価値反復の際のコストの計算には占有格子地図を用いた．

Table 1 には, 地図の大きさ, 格子の解像度, ロボットが移動できる地図のセルの数, 面積を記載する．Table 2 には, 価値反復の離散状態数と行動の数を記載する．離散状態数は, 今回用いる地図のセルの数に, θ 軸上の区分数（今回の実験では 60）をかけたものである．Table 3 には,6 種類の行動と各行動の前進の速度と角速度を記載する．この値は使用するロボットに合わせ, 適切だと考えられる値に調整したものである．

Table 1 configurations of the map

map size	294.3[m] × 199.95[m]
cell resolution	0.15[m] × 0.15[m]
number of cells	2,615,346
number of free cells	165,076
the area of the free cells	3714.98[m^2]

Table 2 parameters for value iterations

number of states	156,920,760
number of free states	9,904,560
number of actions	6

Table 3 type of actions

type	forward versity[m/s]	angular versity[deg/s]
forward	0.3	0.0
backward	-0.2	0.0
right	0.0	-20.0
left	0.0	20.0
right forward	0.2	-20.0
left forward	0.2	20.0

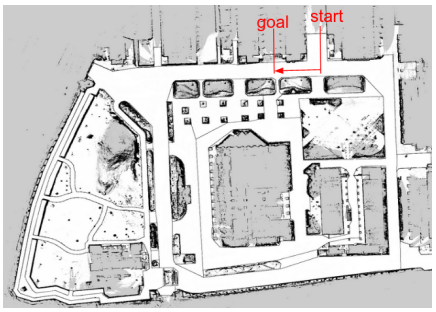

Fig.1 the map for value iteration

2.1 価値反復

今回使用する価値反復の ROS パッケージ [4] は,2 次元の離散化した占有格子地図を読み込み, ロボットの向きの 1 次元を加えた 3 次元の離散空間で経路計画を行っている. この離散空間を $\mathcal{S}$ で表し, 各区分 $s \in \mathcal{S}$ を離散状態と呼ぶ. 大域計画器は, 次の 2 つの写像を出力する.

- 最適状態価値関数 V^*: $\mathcal{S} \to \mathbb{R}$
- 最適方策 Π^*: $\mathcal{S} \to \mathcal{A}$

$\mathcal{A}$ は選択肢として用意された行動セット（Table3）のことである. ある離散状態 $s \in \mathcal{S}$ の値 $V^*(s)$ は,s から最適方策 Π^* にしたがって行動を取り続けた際のコストと移動時間の和の期待値である. コストとは衝突の危険性などを時間の損失に換算し, 数値化したものであり, 占有格子地図から計算される, または人から与えられる.

2.2 並列処理

価値反復のノード（vi_node）は, 他のノードからゴールを指定されると即座に価値反復を開始する. 開始すると, 自己位置推定のノードから出力されるロボットの推定姿勢のトピックを受けて, その時点で得られる方策に基づいて行動（Table3）を選択し, 選んだ行動に対応するロボットの速度を出力する. 価値反復のノードには Linux の POSIX スレッドによる並列処理が導入されている. 並列化されているのは, 状態遷移確率を求める処理と価値反復を行う処理である.

状態遷移確率を求める処理は, ロボットが動作する前に一度だけ行うもので, スレッド数を多くして CPU を使い切り短い時間で終わらせる実装になっている. θ 軸の離散区間ごとにスレッドを用意する実装になっていて, 今回の実験では区間の幅が 6[deg] で, 区間数が 60 個なので 60 個のスレッドが同時に走る.

価値反復に用いるスレッドは設定で変更できるように実装されている. 価値反復はロボットが動作しているときに実行されるため, 演算に使用するスレッド数を多くしてしまうと, センサ処理の遅延などの悪影響が生じてしまう.

2.3 計算量の評価

評価する方法としては, 上田らの研究 [4] と同様の方法を用いる. しかし, スレッド数の増加によってナビゲーションにかかる時間が短縮すること, そして, 物理コア数以上のスレッド数を用いた場合, 短縮の効果が頭打ちであるという結果がでているので, 本研究では使用する計算機の物理コア数が 16 コアであるため, スレッド数は 16 に固定して実験を行う.

2.4 実験結果

ロボットは価値反復の計算終了前に行動を開始するため, 次の A と B の実験の時間の差から実質的な計算時間を評価する. Table 4 に A と B のそれぞれの場合の実験結果を示す. A と B の計測の内容は以下のとおりである.

- A: 価値反復中からロボットを動作させて計測
- B: 価値反復が終了したあとにロボットを動作させて計測

Table4 には,A の実験と B の実験それぞれでの計測した時間の平均と標準偏差を記載した. A の実験の平均時間と B の実験の平均時間の差より, 今回の実験条件では実質的な価値反復の計算時間は 84.6[s] であった.

Table 4 computational complexity

	average of total time[s]	standard deviation[s]
A	123.3	6.2
B	207.8	2.2

今回の実験では価値反復に用いる地図と自己位置推定に用いる地図の大きさを同じにしていた. そのため自己位置推定に悪影響を及ぼさない大きさの地図を使って計算量の評価を行っていた. しかし, 現在の地図の大きさだと価値反復の計算に時間がかかり過ぎてしまう. そのため, 価値反復に用いる地図と自己位置推定に用いる地図を別々にすることで, 価値反復に用いる地図を今以上に小さくすることができる. 地図を小さくすることで, 離散状態数が減少し価値反復の計算にかかる時間が短くなると考えられる.

3 結言

本稿では, 価値反復を用いた実時間経路計画アルゴリズムを屋外移動ロボットで用い, 屋外の実環境を走行させることで計算量を評価した. 結果としては,3714.98[m^2] のフリースペースがある環境において, 地図の解像度を 0.15[m/pixel],Intel Core i7-11800H を有する計算機を用いると, 84.6[s] も計算に時間がかかってしまうという結果が得られた. 結果から, 屋外などの広い環境を想定した場合, 今回の実験の条件では価値反復の計算に時間がかかり過ぎてしまうため, 屋外での走行を想定した移動ロボットへの適用は適切ではないと言える.

しかし, 価値反復に使用する地図の解像度をできる限り下げることで, 離散状態数が小さくなるので, 価値反復の計算に用いる時間を短くすることができると考えられる.

今後は, 価値反復に用いる地図の解像度をどこまで下げることで, 屋外を想定した移動ロボットへの適用が可能になるのかということを調査する.

参考文献

[1] Bellman, R., “*Dynamic Programming*,” Princeton Uni-versity Press, Princeton, NJ, 1957.

[2] 上田隆一，池邉龍宏，林原靖男，“移動ロボットのナビゲーションのための brute-force な価値反復を用いた大域計画・局所計画アルゴリズム”, 第 27 回ロボティクスシンポジア講演論文集, 2022.

[3] Hart, P. E., Nilsson, N. J. and Raphael, B. “A Formal Basis for the Heuristic Determination of Minimal Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100-107, 1968.

[4] 上田隆一，池邉龍宏，林原靖男，“brute-force な価値反復を用いた実時間経路計画 ROS パッケージ”, 第 39 回日本ロボット学会学術講演会予稿集, 2021.

[5] Ueda, R., *et al.*, “Real-Time Decision Making with State-Value Function under Uncertainty of State Estimation,” in *Proc. of* ICRA, 2005.