Vol.2020-ARC-241 No.12
2020/7/31

Graph-based SLAMにおける計算負荷制御による推定精度および計算時間への影響の調査

福重　敢太[1,a)]　和　遠[1]　近藤　正章[1]

概要：本研究では graph-based simultaneous localization and mapping (SLAM) を扱う．SLAM とはロボットが未知の環境下で環境の地図生成と自己位置推定を同時に行う手法であり，その中でもロボットの姿勢と環境内のランドマークをノードとし，観測や制御の関係をエッジとしたグラフを構築し，その最適化を行うことで問題を解決する手法が graph-based SLAM である．Graph-based SLAM はある程度蓄積したデータに対し推定を行う完全 SLAM の一手法であり，精度が良い一方で，計算時間が大きいという課題がある．本稿では，graph-based SLAM の高速化に向け，推定に用いる点群数などの計算条件を変えた際の推定精度および計算時間への影響を調査する．シミュレーションにより作成したデータセットを用いた評価により，推定精度および計算時間を計測し解析する．評価の結果，観測データ量の削減は主に計算時間の観点から処理の高効率化に有効であることがわかった．

1. はじめに

モバイルロボットなどにより運送や救助，自動清掃を効率的に行う場合には環境の地図が必要であり，また同時にロボットが自身の位置を推定することが必要である．特に，災害時や屋内などでは周囲の環境の地図がない場合や GPS などの外部の測位システムが使用できない場合も考えられ，そのような未知の環境下においても正確な動作を行うために，ロボット自身が環境の地図生成と自己位置推定を行うことが課題である．このように未知の環境下において環境の地図生成と自己位置推定を同時に行う手法として SLAM（simultaneous localization and mapping）[1] と呼ばれる手法がある．近年では，自動車の自動運転やスマートデバイス等の発展により SLAM を用いた自己位置推定が注目されており，性能の向上や処理の効率化が求められている．SLAM は，ロボットがカメラや LiDAR などのセンサを用いて周囲を観測し，得られた観測データから環境の地図と自己位置を推定することが主流であり，用いるセンサの種類，推定手法などから様々な手法が提案されている．

SLAM は大きく分けて，オンライン SLAM と完全 SLAM の 2 つに分類することができる．オンライン SLAM とは，ロボットの現在の位置とその位置から見えている環境の地図を逐次的に推定，更新する SLAM である．オンライン SLAM 手法として，カルマンフィルタを用いた自己位置推定 [2]，パーティクルフィルタを用いた自己位置推定 [3], [4], [5]，FastSLAM[6] などが提案されており，逐次的に自己位置推定を行うためリアルタイム性を持つ手法であるが，一方で累積誤差の影響が大きく，またループの検出や閉じ込みが難しいなどの課題がある．これに対して，完全 SLAM とはセンサデータを蓄積した後に，それらすべてのデータを用いてロボットの姿勢の履歴，すなわち軌跡と環境の地図のすべてを推定する SLAM である．完全 SLAM では，蓄積したデータを用いてロボットの軌跡を求めることから，オンライン SLAM と比較して精度が良いという利点があるが，推定に用いるデータが大きくなってしまうため，推定時間が大きいという課題がある．

完全 SLAM の主な手法として，graph-based SLAM[7], [8], [9] がある．Graph-based SLAM は，センサから得られた観測データを用いて構築されるグラフ表現を用いた SLAM の一手法である．観測データのうちロボットの姿勢と環境中のランドマークをノード，観測被観測や制御の関係をエッジとしたグラフを構築し，そのグラフを最適化することで環境の地図生成と自己位置推定を同時に達成する．ここでランドマークとは，環境中の特徴となる物体のことを指すが，カメラから得られる画像データ内のエッジや LiDAR から得られる点群データの特徴ある並びなどから得られる．Graph-based SLAM は観測されたすべてのランドマークや制御量を用いて，環境や軌跡のすべてを推

[1] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[a)] fukushige@hal.ipc.i.u-tokyo.ac.jp

定する完全 SLAM の一つであり，規模の大きな環境に対する実行では計算時間が極めて大きくなるという課題がある．特にバッテリ駆動などのロボットでは処理の高効率化が必要であり，得られるグラフが疎であることなどを利用するなど，効率よく計算することが求められる．

本稿では，graph-based SLAM の処理の高効率化に向け，軌跡や推定値に用いるランドマークの観測数や，浮動小数点数のビット長などが推定精度や計算時間へ与える影響について調査する．シミュレーション実験により推定精度や計算時間を計測し，推定に用いるランドマーク数との関係を考察する．

2. Graph-based SLAM

1 章で述べたように，graph-based SLAM はグラフ表現を用いた SLAM であり，得られたセンサデータからグラフを構築し，構築したグラフを最適化する最適化問題として定式化される．グラフ最適化問題の定式化やその解法，および推定のアルゴリズムとして様々な提案がなされているが，本稿では文献 [7] および [8] に基づく．

Graph-based SLAM の処理は大きく 2 つに分けることができる．一方は front-end と呼ばれるセンサデータを処理しグラフを構築するタスクであり，もう一方は back-end と呼ばれる構築したグラフを最適化し，軌跡や地図などの系の状態を推定するタスクである．front-end では，得られた観測データからランドマークを抽出し，そのランドマークに ID を割り振るといった処理を行う．この ID はランドマークを判別するために必要であり，graph-based SLAM では異なる時刻間で同一のランドマークの観測を行うことにより推定を行うため，重要な情報となる．本稿では，得られた観測データに関して適切にランドマークが抽出され，ID が割り振られたものとして，それらからグラフを構築し最適化を行う back-end について取り扱う．

文献 [7] および [8] に基づくアルゴリズムでは，まずロボットの姿勢のみをノードとしたグラフを構築して，観測や制御といった関係をもとにエッジを作成し，それを拘束条件としたもとで最適化を行うことでロボットの姿勢を推定する．その後，ロボットの姿勢の推定値をもとにランドマークの位置を推定する．以下では graph-based SLAM の定式化を紹介する．

2.1 推定問題の定式化

完全 SLAM はロボットの初期姿勢，観測値および制御値からロボットの軌跡と環境の地図を同時に推定する手法であり，確率密度関数

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{m} \mid \boldsymbol{x}_0, \boldsymbol{u}_{1:T}, \boldsymbol{z}_{1:T}) \tag{1}$$

を最大とする $\boldsymbol{x}_{1:T}$ および $\boldsymbol{m}$ を求める問題として定式化される．ここで $\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{z}_t$ はそれぞれ時刻 t におけるロボットの姿勢，制御値および観測値を表し，$\boldsymbol{m}$ はランドマークの集合である地図を表す．式 (1) は次のように分解できる．

$$\begin{aligned} &p(\boldsymbol{x}_{1:T}, \boldsymbol{m} \mid \boldsymbol{x}_0, \boldsymbol{u}_{1:T}, \boldsymbol{z}_{1:T}) \\ &= p(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0, \boldsymbol{u}_{1:T}, \boldsymbol{z}_{1:T}) p(\boldsymbol{m} \mid \boldsymbol{x}_{0:T}, \boldsymbol{z}_{1:T}) \end{aligned} \tag{2}$$

これより完全 SLAM は $p(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0, \boldsymbol{u}_{1:T}, \boldsymbol{z}_{1:T})$ を最大とする $\boldsymbol{x}_{1:T}$ を求める問題と $p(\boldsymbol{m} \mid \boldsymbol{x}_{0:T}, \boldsymbol{z}_{1:T})$ を最大とする $\boldsymbol{m}$ を求める問題に分解することができる．

Graph-based SLAM では，前者を解くことでロボットの軌跡 $\boldsymbol{x}_{1:T}$ を推定し，その結果をもとに後者の問題を解くことで環境の地図 $\boldsymbol{m}$ を推定する．以下では，それぞれの問題に関して定式化を行う．

2.1.1 ロボットの軌跡の算出

ロボットの軌跡の算出問題は以下のように定式化される．

$$\boldsymbol{x}^*_{1:T} = \arg\max_{\boldsymbol{x}_{1:T}} p(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0, \boldsymbol{u}_{1:T}, \boldsymbol{z}_{0:T}) \tag{3}$$

簡単のため，$\boldsymbol{x}_0$ も推定対象に含めることとする．このとき，与えられる初期姿勢を $\hat{\boldsymbol{x}}_0$ として, 式 (3) は以下のように書き換えられる．

$$\boldsymbol{x}^*_{0:T} = \arg\max_{\boldsymbol{x}_{0:T}} p(\boldsymbol{x}_{0:T} \mid \hat{\boldsymbol{x}}_0, \boldsymbol{u}_{1:T}, \boldsymbol{z}_{0:T}) \tag{4}$$

この問題と同値である最小二乗問題を解くことで，ロボットの軌跡を推定する．以下では最小二乗問題の導出を行う．

まず，ロボットの姿勢をノードとしたグラフを考える．初期姿勢 $\hat{\boldsymbol{x}}_0$ と制御値 $\boldsymbol{u}_{1:T}$ から状態方程式を介して姿勢の初期推定値 $\hat{\boldsymbol{x}}_{1:T}$ を求めることができる．これより，連続した時刻間でのロボットの姿勢にエッジがあるようなグラフを構築することができる．この制御値により連続した時刻間でのロボットの姿勢間に存在するエッジを移動エッジと呼ぶ．

一方で，ロボットの姿勢 $\hat{\boldsymbol{x}}_{0:T}$ 間には，ランドマークの観測により互いに拘束を持つ組が存在する．そのような同一のランドマークを観測した時刻のロボットでは，姿勢 $\hat{\boldsymbol{x}}_t$ と観測 $\boldsymbol{z}_t$ から求められるランドマークの位置は等しいはずである．その拘束条件からロボットの姿勢間にエッジを作成することができ，そのエッジを仮想移動エッジと呼ぶ．しかしながら，実際の観測値や制御値にはノイズが存在するため，同一のランドマークの観測をした場合にも，推定姿勢と観測値から計算されるランドマークの位置にはズレが生じる．このズレを仮想移動エッジの残差と呼び，残差が小さくなるように推定姿勢を更新する．

このとき，推定姿勢を更新することにより，制御値から求められる推定姿勢に対してズレが生じることとなる．このズレを移動エッジの残差と呼ぶ．Graph-based SLAM では，これらの仮想移動エッジの残差，移動エッジの残差を小さくするような推定姿勢を導出する．

仮想移動エッジ，移動エッジの残差をそれぞれ

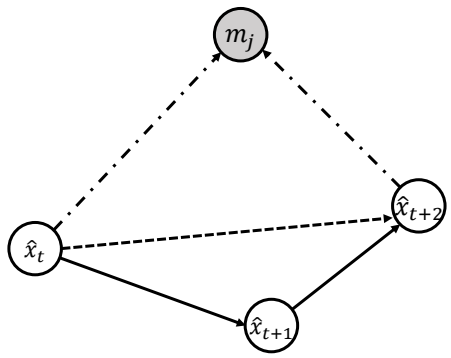

図 1 ロボットの姿勢，観測値および制御値から構築されるグラフ．実線は制御値および移動エッジを表し，点鎖線はランドマークの観測を，点線は仮想移動エッジを表す．

$\boldsymbol{e}_{j,t_1,t_2}, \boldsymbol{e}_{t_1,t_2}$ で表し，以下で定義する．

$$\begin{aligned}\boldsymbol{e}_{j,t_1,t_2} =& \\ &(\boldsymbol{x}_{t_2}\text{と }\boldsymbol{z}_{t_2}\text{から求められるランドマーク位置}) \\ &-(\boldsymbol{x}_{t_1}\text{と }\boldsymbol{z}_{t_2}\text{から求められるランドマーク位置}) \quad (5)\\ \boldsymbol{e}_{t_1,t_2} =& \\ &\boldsymbol{x}_{t_2} - (\boldsymbol{x}_{t_1}\text{と }\boldsymbol{u}_{t_1}\text{から求められる } t_2\text{での姿勢}) \quad (6)\end{aligned}$$

さらにこれらの残差の値 $\boldsymbol{e}$ の確率分布を以下で仮定する．

$$p(\boldsymbol{e}) = \mathcal{N}\left(\boldsymbol{e} \mid \boldsymbol{0}, \boldsymbol{\Omega}_{\mathrm{e}}^{-1}\right) = \eta \exp\left(-\frac{1}{2}\boldsymbol{e}^{\mathsf{T}}\boldsymbol{\Omega}_{\mathrm{e}}\boldsymbol{e}\right) \quad (7)$$

ここで，$\mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ は変数 $\boldsymbol{y}$ に対する平均ベクトル $\boldsymbol{\mu}$，分散共分散行列 $\boldsymbol{\Sigma}$ の正規分布を表し，η は正規化定数を表す．

以上の仮定の下で，式 (7) から次の関数を定義する．

$$f(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_0)\times \left\{\prod_{(j,t_1,t_2)\in\mathbf{I}_{\mathbf{e_z}}} p(\boldsymbol{e}_{j,t_1,t_2})\right\}\left\{\prod_{(t_1,t_2)\in\mathbf{I}_{\mathbf{e_x}}} p(\boldsymbol{e}_{t_1,t_2})\right\}^{\lambda} \quad (8)$$

ここで，$\mathbf{I}_{\mathbf{e_z}}, \mathbf{I}_{\mathbf{e_x}}$ はそれぞれ仮想移動エッジ，移動エッジのインデクスの集合を表し，$\lambda \geq 0$ は仮想移動エッジと移動エッジの重みを変えるためのパラメータである．この関数を最大とする推定姿勢 $\boldsymbol{x}^*_{0:T}$ は残差のモデルに対して尤もらしい軌跡となり，それを求めることが式 (4) で定義される問題を解くことと同値となる．すなわち，以下の目的関数を最小とする最小二乗問題とみなすことができる．

$$\begin{aligned}J(\boldsymbol{x}_{0:T}) &\stackrel{\text{def}}{=} -\log f(\boldsymbol{x}_{0:T}) \quad (9)\\ &\stackrel{c}{=} (\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0)^{\mathsf{T}}\Omega_0(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0) \\ &\quad + \sum_{(j,t_1,t_2)\in\mathbf{I}_{\mathbf{e_z}}} \boldsymbol{e}^{\mathsf{T}}_{j,t_1,t_2}\Omega_{j,t_1,t_2}\boldsymbol{e}_{j,t_1,t_2} \\ &\quad +\lambda \sum_{(t_1,t_2)\in\mathbf{I}_{\mathbf{e_x}}} \boldsymbol{e}^{\mathsf{T}}_{t_1,t_2}\Omega_{t_1,t_2}\boldsymbol{e}_{t_1,t_2} \quad (10)\end{aligned}$$

ここで $\stackrel{c}{=}$ は左辺と右辺が定数を除いて等しいことを表し，式 (10) の第 2 項を $J_{\mathbf{z}}(\boldsymbol{x}_{0:T})$，第 3 項を $J_{\boldsymbol{x}}(\boldsymbol{x}_{0:T})$ と定義する．式 (10) は，ベクトル $\boldsymbol{e}$ の内積 $\boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}$ を情報行列 $\boldsymbol{\Omega}_{\boldsymbol{e}}$ で重みづけしたマハラノビス距離の二乗 $\boldsymbol{e}^{\mathsf{T}}\boldsymbol{\Omega}_{\mathbf{e}}\boldsymbol{e}$ の和を最小化する最小二乗問題となり，最小二乗法を用いて解く．以下ではその解法を説明する．

まず，仮想移動エッジおよび移動エッジの残差を多項式で近似する．

$$\begin{aligned}\boldsymbol{e}_{j,t_1,t_2} \simeq \hat{\boldsymbol{e}}_{j,t_1,t_2} &+ \left.\frac{\partial \boldsymbol{e}_{j,t_1,t_2}}{\partial \boldsymbol{x}_{t_1}}\right|_{\boldsymbol{x}_{t_1}=\hat{\boldsymbol{x}}_{t_1}} \Delta\boldsymbol{x}_{t_1} \\ &+\hat{\boldsymbol{e}}_{j,t_1,t_2} + \left.\frac{\partial \boldsymbol{e}_{j,t_1,t_2}}{\partial \boldsymbol{x}_{t_2}}\right|_{\boldsymbol{x}_{t_2}=\hat{\boldsymbol{x}}_{t_2}} \Delta\boldsymbol{x}_{t_2} \quad (11)\\ \boldsymbol{e}_{t_1,t_2} \simeq \hat{\boldsymbol{x}}_{t_2} &- \boldsymbol{f}(\hat{\boldsymbol{x}}_{t_1}, \boldsymbol{u}_{t_2}) \\ &+\Delta\boldsymbol{x}_{t_2} - \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_{t_1}}\right|_{\boldsymbol{x}_{t_1}=\hat{\boldsymbol{x}}_{t_1}} \Delta\boldsymbol{x}_{t_1} \quad (12)\end{aligned}$$

ここで，関数 $\boldsymbol{f}$ は状態方程式を表し，$\boldsymbol{f}(\boldsymbol{x}_{t_1}, \boldsymbol{u}_{t_2})$ は姿勢 $\boldsymbol{x}_{t_1}$ と制御値 $\boldsymbol{u}_{t_2}$ から導かれる時刻 t_2 でのロボットの姿勢を表す．$\Delta\boldsymbol{x}_t$ を縦に並べた $3(T+1)$ 次元ベクトル

$$\Delta\boldsymbol{x}_{[0:T]} = \left(\Delta\boldsymbol{x}_0^{\mathsf{T}}, \Delta\boldsymbol{x}_1^{\mathsf{T}}, \cdots, \Delta\boldsymbol{x}_T^{\mathsf{T}}\right)^{\mathsf{T}} \quad (13)$$

を新たに変数として導入すると，$J_{\mathbf{z}}(\boldsymbol{x}_{0:T})$ および $J_{\boldsymbol{x}}(\boldsymbol{x}_{0:T})$ の各項はそれぞれ

$$\Delta\boldsymbol{x}^{\mathsf{T}}_{[0:T]}\Omega^*_{j,t_1,t_2}\Delta\boldsymbol{x}_{[0:T]} - 2\Delta\boldsymbol{x}^{\mathsf{T}}_{[0:T]}\boldsymbol{\xi}^*_{j,t_1,t_2} + C \quad (14)$$

$$\Delta\boldsymbol{x}^{\mathsf{T}}_{[0:T]}\Omega^*_{\boldsymbol{x}_{t_1},\boldsymbol{x}_{t_2}}\Delta\boldsymbol{x}_{[0:T]} - 2\Delta\boldsymbol{x}^{\mathsf{T}}_{[0:T]}\boldsymbol{\xi}^*_{\boldsymbol{x}_{t_1},\boldsymbol{x}_{t_2}} + C \quad (15)$$

となる．ただし，C は定数項を表す．また，$\Omega^*_{j,t_1,t_2}, \Omega^*_{\boldsymbol{x}_{t_1},\boldsymbol{x}_{t_2}}$ はそれぞれ式 (13) で与えられる変数の次元に対応する情報行列であり，$\boldsymbol{\xi}^*_{j,t_1,t_2}, \boldsymbol{\xi}^*_{\boldsymbol{x}_{t_1},\boldsymbol{x}_{t_2}}$ はそれぞれ係数ベクトルである．これらは式 (11) および式 (12) におけるヤコビ行列，情報行列 $\Omega_{j,t_1,t_2}, \Omega_{\boldsymbol{x}_1,\boldsymbol{x}_2}$，ならびに残差から導出される．このとき，目的関数 $J(\boldsymbol{x}_{0:T})$ は以下のように表すことができる．

$$J(\boldsymbol{x}_{0:T}) \stackrel{c}{=} \Omega\Delta\boldsymbol{x}_{[0:T]} - 2\Delta\boldsymbol{x}^{\mathsf{T}}_{[0:T]}\boldsymbol{\xi} \quad (16)$$

ただし，Ω，$\boldsymbol{\xi}$ はそれぞれ目的関数の情報行列および係数ベクトルであり，以下で定義される．

$$\Omega = \Omega^*_0 + \sum_{(j,t_1,t_2)\in\mathbf{I}_{\mathbf{e_z}}} \Omega^*_{j,t_1,t_2} + \lambda \sum_{(t_1,t_2)\in\mathbf{I}_{\mathbf{e_x}}} \Omega^*_{\boldsymbol{x}_{t_1},\boldsymbol{x}_{t_2}} \quad (17)$$

$$\boldsymbol{\xi} = \sum_{(j,t_1,t_2)\in\mathbf{I}_{\mathbf{e_z}}} \boldsymbol{\xi}^*_{j,t_1,t_2} + \lambda \sum_{(t_1,t_2)\in\mathbf{I}_{\mathbf{e_x}}} \boldsymbol{\xi}^*_{\boldsymbol{x}_{t_1},\boldsymbol{x}_{t_2}} \quad (18)$$

このとき目的関数 $J(\boldsymbol{x}_{0:T})$ を最小とする $\Delta\boldsymbol{x}_{[0:T]}$ は

$$\Delta\boldsymbol{x}^*_{[0:T]} = \Omega^{-1}\boldsymbol{\xi} \quad (19)$$

で与えられる．これより，ロボットの姿勢は

$$\boldsymbol{x}^*_{[0:T]} = \hat{\boldsymbol{x}}_{[0:T]} + \Omega^{-1}\boldsymbol{\xi} \quad (20)$$

で与えられ，新たに $\hat{\boldsymbol{x}}_{[0:T]}$ を $\boldsymbol{x}^*_{[0:T]}$ で置き換え反復計算を行うことにより最終的な解を得る．

2.1.2 地図の算出

ランドマークの位置は，前項の問題を解いて最終的に得られた推定値 $\boldsymbol{x}^*_{0:T}$ を真とし，複数のセンサ値から計算されるランドマークの位置を重ね合わせることで推定できる．この問題は以下で定義される目的関数を最小とするランドマークの位置 $\boldsymbol{m}_j$ を求める最小二乗問題として定式化される．

$$\begin{aligned} J_{\boldsymbol{m}_j}(\boldsymbol{m}_j) &\stackrel{\text{def}}{=} -\log f(\boldsymbol{m}_j) \\ &= \sum_{t\in\boldsymbol{I_z}} \boldsymbol{e}_{j,t}^{\mathsf{T}}\Omega_{j,t}\boldsymbol{e}_{j,t} \end{aligned} \tag{21}$$

$$f(\boldsymbol{m}_j) = \prod_{t\in\mathbf{I_z}} p_{i,j}(\boldsymbol{e}_{j,t}) \tag{22}$$

$$p_{j,t}(\boldsymbol{e_{j,t}}) = \eta\exp\left(-\frac{1}{2}\boldsymbol{e}_{j,t}^{\mathsf{T}}\boldsymbol{\Omega}_{j,t}\boldsymbol{e}_{j,t}\right) \quad \left(t\in\mathbf{I}_{\boldsymbol{m}_j}\right) \tag{23}$$

ここで，$\boldsymbol{m}_j$ はランドマーク j の位置を表し，$\mathbf{I}_{\boldsymbol{m}_j}$ はランドマーク j の観測があった時刻の集合を表す．また，$\boldsymbol{e}_{j,t}$ はランドマーク位置の残差を表し，以下で定義される．

$$\begin{aligned} &\boldsymbol{e}_{j,t} = \boldsymbol{m}_j \\ &\quad -(\boldsymbol{x}^*_t, \boldsymbol{z}_{j,t}\text{から求められるランドマークの位置}) \end{aligned} \tag{24}$$

なお，$\boldsymbol{z}_{j,t}$ は時刻 t におけるランドマーク j に対する観測量を表す．各ランドマークに対し最小二乗法を用いて式 (21) を目的関数とした最小二乗問題を解き推定値を計算する．以下でその解法を紹介する．

$\boldsymbol{x}^*_t, \boldsymbol{z}_{j,t}$ から求められるランドマークの位置を $\boldsymbol{m}_{j,t}$ とすると，残差は以下で表される．

$$\boldsymbol{e}_{j,t} = \boldsymbol{m}_j - \boldsymbol{m}_{j,t} \tag{25}$$

また，情報行列 $\Omega_{j,t}$ はセンサのノイズに関する分散共分散行列 $Q_{j,t}$ から以下で与えられる．

$$\Omega_{j,t} = \left(\left.\frac{\partial\boldsymbol{m}_{j,t}}{\partial\boldsymbol{z}}\right|_{\boldsymbol{z}=\hat{\boldsymbol{z}}_{j,t}}\right) Q_{j,t} \left(\left.\frac{\partial\boldsymbol{m}_{j,t}}{\partial\boldsymbol{z}}\right|_{\boldsymbol{z}=\hat{\boldsymbol{z}}_{j,t}}\right)^{\mathsf{T}} \tag{26}$$

このとき，式 (21) で与えられる目的関数 $J_{\boldsymbol{m}_j}(\boldsymbol{m}_j)$ は

$$J_{\boldsymbol{m}_j}(\boldsymbol{m}_j) \stackrel{c}{=} \boldsymbol{m}_j^{\mathsf{T}} \sum_{t\in\boldsymbol{I_z}} \Omega_{j,t}\boldsymbol{m}_j - 2\boldsymbol{m}_j^{\mathsf{T}} \sum_{t\in\boldsymbol{I_z}} \Omega_{j,t}\boldsymbol{m}_{j,t} \tag{27}$$

と表される．情報行列 Ω_j および係数ベクトル $\boldsymbol{\xi}_j$ を新たに

$$\Omega_j = \sum_{t\in\boldsymbol{I_z}} \Omega_{j,t} \tag{28}$$

$$\boldsymbol{\xi}_j = \sum_{t\in\boldsymbol{I_z}} \Omega_{j,t}\boldsymbol{m}_{j,t} \tag{29}$$

と定義すると，目的関数 $J_{\boldsymbol{m}_j}(\boldsymbol{m}_j)$ を最小とする $\boldsymbol{m}^*_j$ は

$$\boldsymbol{m}^*_j = \Omega_j^{-1}\boldsymbol{\xi}_j \tag{30}$$

で与えられ，これがランドマーク j の推定位置となる．

2.2 推定アルゴリズムと計算量

本節では，前節までに紹介した graph-based SLAM の問題を解く際の具体的な流れについてまとめ，それぞれのステップにおける計算量を述べる．

1). エッジの作成：
現在の軌跡の推定値 $\boldsymbol{x}^*_{0:T}$ とセンサ値から仮想移動エッジ，および移動エッジを作成し，その残差を計算する．このステップにおける計算量は $\mathcal{O}(T + \mathrm{Z} + \sum_j \mathrm{Z}_j^2)$ である．ただし，Z は全時刻における観測数の和を表し，Z_j は全時刻におけるランドマーク j に対する観測数の和を表す．このとき，$\mathrm{Z} = \sum_j \mathrm{Z}_j$ を満たす．

2). 情報行列 $\boldsymbol{\Omega}$ および係数ベクトル $\boldsymbol{\xi}$ の作成：
1) で作成したエッジの値から目的関数の情報行列 $\boldsymbol{\Omega}$ および係数ベクトル $\boldsymbol{\xi}$ を計算する．このステップにおける計算量は $\mathcal{O}(T + V)$ である．ただし，V は仮想移動エッジの数を表す．

3). 更新値の計算：
2) で作成した情報行列 $\boldsymbol{\Omega}$ および係数ベクトル $\boldsymbol{\xi}$ から軌跡の更新値 $\Delta\boldsymbol{x}^*_{[0:T]}$ を計算する．このステップにおける計算量は逆行列の計算方法に依存するが，計算方法を固定すれば，行列の大きさ $3(T+1)\times 3(T+1)$ により決定する．

4). 反復終了判定：
3) で計算した更新値から反復終了条件を満たすかどうかを判定する．反復終了条件を満たした場合は推定値を更新し，4) へ進む．反復終了条件を満たさない場合は推定値を更新し，1) へ進む．

5). ランドマーク位置の計算．
各ランドマークに関して，ランドマーク位置の残差を計算し，情報行列 Ω_j と係数ベクトルを $\boldsymbol{\xi}_j$ 計算し，推定値を計算する．このステップにおける計算量は $\mathcal{O}(\mathrm{Z})$ である．

3. Graph-based SLAM の計算負荷制御

1 章で述べたように，graph-based SLAM は完全 SLAM の一手法であり，環境の規模が大きくなるにつれ計算時間も極めて大きくなるという課題がある．例えば，943 ノード，1837 エッジから成る Intel データセット [10] に対する実行では，既存のグラフ処理ライブラリ g^2o[11] を用いた場合の推定にかかる計算時間が 19.51 ms であることが報告されている [9]．また，特に近年では，カメラや LiDAR などのセンサの高性能化により得られるセンサデータ量も増大しており，規模の大きな環境でなくても観測データの膨大により計算時間が大きくなることが予想される．一方で，SLAM の応用が考えられるモバイルロボット等では，運送や救助に用いることからバッテリ駆動であることが予想され，そのようなロボットでは電力の観点から制約があり，SLAM の処理を高速化をはじめとして高効率化するこ

とが求められている．

これらの課題に対し，文献 [9] ではアルゴリズムの改善を行い，それを GPU と CPU を搭載した SoC にマッピングし実行することにより高速化を行った．その結果，先に述べた Intel データセットに対し，その実行時間は 8.9 ms に高速化されたと報告されている．一方で，推定に用いる観測データ量を削減することによる高効率化や計算に用いる浮動小数点数のビット長を最適化することによる高効率化，またそれらを graph-based SLAM 専用のアーキテクチャとして構築することなどは，従来あまり検討されてこなかった．本研究では、具体的な応用や外部の環境に依存して必要な性能を決定し，その精度を達成するために必要最低限な観測データ量，浮動小数点数のビット長を用いて graph-based SLAM を実行することにより，高速・低消費電力化の実現を目指している．

それに向け，まず推定に用いる観測データ量や浮動小数点数のビット長が推定性能や計算時間に与える影響を知る必要がある．そこで本稿では，推定に用いるランドマークの観測数 Z や浮動小数点数のビット長を変化させた際の推定精度や計算時間への影響について調査する．同時に，アルゴリズムの各処理の計算時間を計測することにより，graph-based SLAM 専用のハードウェアの構築による高速化を行うための，処理のボトルネックの解析も行う．

4. 計算負荷制御の評価

本章では，シミュレーションで作成したデータセットに対し，ランドマークの観測データ量や浮動小数点数のビット長を変更し実験を行い，その推定結果や計算時間を解析することでそれらの計算負荷制御が結果に及ぼす影響を実験から調査する．4.1 節では，シミュレーションにより作成するデータセットの詳細を述べ，4.2 節ではその他の実験条件について述べる．最後に 4.3 節では実験により得られた結果を述べ，その解析を行う．

4.1 データセットの作成

実験に用いるデータセットはシミュレーションにより作成した．平面上に $(0, 0)$ を中心とした 1 辺の長さが 32 である正方形内に等間隔なグリッド状にランドマークを 1024 個配置し，その平面上をセンサを搭載したロボットが $(0, 0)$ を中心とした半径 10 の円運動を 2 周するシミュレーションを行った．ロボット移動のタイムフレーム数は 120 とした．ロボットに搭載したセンサは視野角が正面から左右にそれぞれ $\pi/3$ であり，観測できる距離が 0.5 から 6.0 であるとした．また，センサは観測値として被観測物までの距離と角度を出力し，ロボットへの制御値としては速度と角速度を想定した．このとき，ロボットの動作やセンサによる観測にノイズを加えた．以下にその詳細を述べる．

- ロボットの移動に対するノイズ：
  ある時刻ごとにロボットの向きにノイズを加えた．ノイズを加える間隔は指数分布 $P(x) = 5e^{-5x}$ に従うとし，加えるノイズは平均 0，標準偏差 $\pi/60$ である正規分布に従うとした．
- ロボットの移動速度へのバイアス：
  ロボットへの制御値である速度および角速度に一定の定数を乗算することにより，ロボットの移動速度へバイアスを加えた．定数は平均 1，標準偏差 0.1 である正規分布に従う乱数とした．これにより，ロボットへの制御値と実際のロボットの動作による出力は異なる値となる．
- 観測値に対するノイズ：
  センサによる観測値にノイズを加えた．距離に対するノイズは，平均が被観測物までの距離，標準偏差が被観測物までの距離の 0.1 倍の値である正規分布に従うとし，角度に対するノイズは，平均が被観測物までの角度，標準偏差が $\pi/90$ である正規分布に従うとした．これはセンサによる観測が距離が大きくなるほど曖昧となることに対応する．
- 観測値に対するバイアス：
  センサによる観測値である距離に一定の定数を乗算し，角度に一定の定数を加えることにより，センサによる観測値へバイアスを加えた．距離に乗算する定数は，平均 1，標準偏差 0.1 である正規分布に従う乱数であり，角度に加える定数は平均 0，標準偏差 $\pi/90$ である乱数とした．

以上に述べたシミュレーションを，ノイズの値を変更して 10 回行い，10 種類のデータセットを作成した．このとき観測されたランドマーク数と観測数を**表 1** に示す．さらに，作成したデータセットを親データセットとし，親データセットに対し観測データ量を削減したデータセットを作成した．これを子データセットと呼ぶこととする．

観測データの削減は対象をランドマークおよび観測数のいずれかとした．削減割合を r とし，削減対象の数を n としたとき，ランドマークを対象とした場合には $\lfloor nr \rfloor$ 個のランドマークに対する観測をすべて，観測数を対象とした場合には $\lfloor nr \rfloor$ 個の観測データを削除した．ここで，$\lfloor \cdot \rfloor$ は床関数を示す．このとき，$\lfloor nr \rfloor$ 個の削減対象は一様分布に従う重複のない乱数により決定した．4.3 節では，それぞれの削減に対し乱数のシードを変えた 10 個のデータセットに対し実験を行い，それらの結果をもとに解析を行った．

4.2 実験条件

4.1 節で述べたデータセットを用いて評価を行う際の実験条件ついて説明する．観測データ量の削減割合は 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 としてそれぞれに対して実験を行った．削減割合が 0.0 であるデータセットは親データセットと同一であり，削減割合が 1.0 である

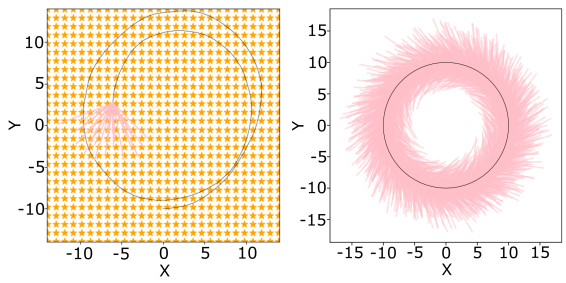

図 2 シミュレーションにおけるロボットの軌跡（左）と観測値および制御値から導かれる軌跡の初期推定値（右）．星はランドマークを示し，黒の実線はロボットの軌跡を，ピンクの実線はロボットからの観測を示す．

表 1 親データセットにおける観測されたランドマーク数と観測数．

Dataset	# of observed landmarks	# of observations
1	677	3764
2	613	3846
3	691	3824
4	661	3807
5	688	3702
6	601	3843
7	555	3760
8	661	3289
9	675	3556
10	679	3767

データセットは観測がなく，制御値のみで推定を行うデータセットとなる．仮想移動エッジと移動エッジの重みを決めるパラメータ λ は 1 とし，また浮動小数点数のビット長は 32 bit および 64 bit とした．Intel Core i9-9900K @ 3.60 GHz，64 GB のメモリを搭載したマシンで実験を行い，使用言語は Python とした．また，2.2 節で述べたアルゴリズムのステップ 4) における反復終了条件は

$$\frac{|J_{n-1}(\boldsymbol{x}_{0:T}) - J_n(\boldsymbol{x}_{0:T})|}{J_n(\boldsymbol{x}_{0:T})} < 10^{-4} \tag{31}$$

とした．ここで $J_n(\boldsymbol{x}_{0:T})$ は n 回目の反復における目的関数の値を示す．

4.3 実験結果および考察

1 つの親データセットおよびそれから作成した子データセットに対する実験結果を**図 3**〜**図 8**，および**表 2**〜**表 5** に示す．図 3 および**図 4** は，反復終了条件を満たした際のロボット位置の推定値と真値との平均誤差を示しており，**図 5** および**図 6** はアルゴリズムの 1 反復ごとの平均計算時間を，**図 7** および図 8 はアルゴリズムステップ 5) の平均計算時間を示している．図中の RR は削減割合を示す．また，表 2 および**表 3** は浮動小数点数のビット長を 64 bit とした際の各実験条件における各アルゴリズムステップでの計算時間の削減率を示している．削減率は削減割合が 0.0 である場合の平均計算時間を基準としており，アルゴリズムステップ 1)〜3) については 1 反復ごとの削減率を示し

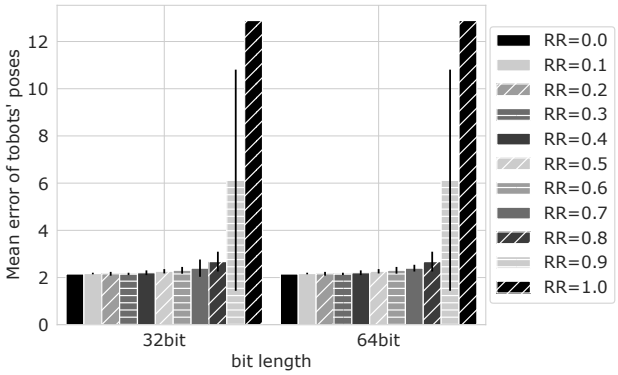

図 3 削減対象をランドマークとした際の，各削減割合および浮動小数点数のビット長に対するロボット位置の平均誤差．エラーバーは標準偏差を表す．

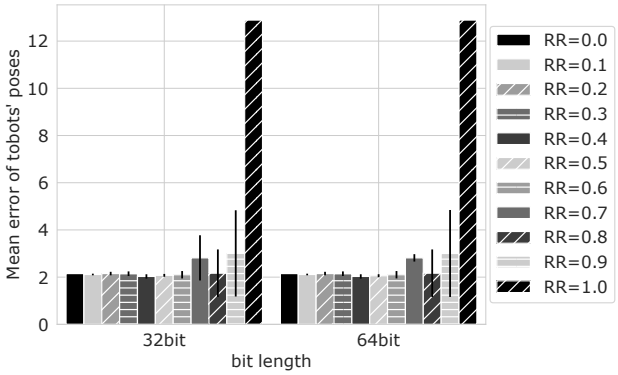

図 4 削減対象を観測数とした際の，各削減割合および浮動小数点数のビット長に対するロボット位置の平均誤差．エラーバーは標準偏差を表す．

ている．**表 4** および表 5 は各実験条件における反復終了条件を満たすまでの平均反復回数を示している．

図 3 および図 4 から，削減対象がいずれの場合も削減割合が大きい場合には，平均誤差が大きくなる傾向があり，標準偏差も大きくなることがわかる．一方で，削減割合が 0.8 以下では平均誤差はあまり変わらないこともわかる．また，図 5〜図 8，表 2 および表 3 から，削減割合が大きくなるにつれ，計算時間は小さくなることがわかる．しかしながら，削減割合が 1.0 である場合には計算時間が大きくなっており，これに関してはより詳細な検証が必要である．さらにアルゴリズムのステップ別では，ステップ 1), 2) および 5) では全体の計算時間と同様の結果が見られるが，ステップ 3) の計算時間は削減割合によらずおおよそ一定であることがわかる．これらの結果は，削減割合が 1.0 である場合を除き，2.2 節で述べた計算量に対し妥当な結果となっている．

また，表 4 および表 5 から，反復終了条件を満たすまでの反復回数はランドマークを削減する場合には概ね増加し，観測数を削減する場合には概ね減少することがわかる．これは，削減対象をランドマークとした場合の方が，ランドマークごとに観測を削減することから観測データの密度

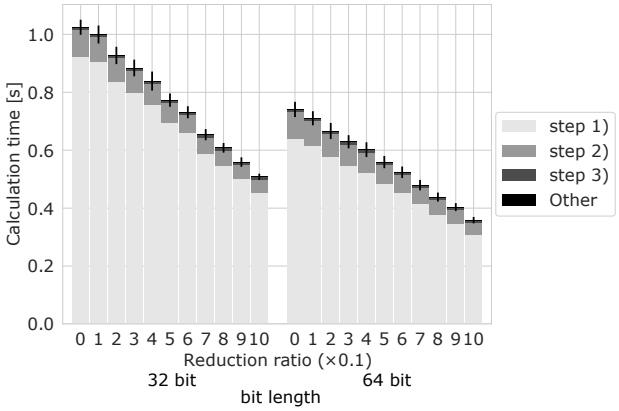

図 5 削減対象をランドマークとした際の，各削減割合および浮動小数点数のビット長に対する，1 反復ごとのロボットの推定における平均計算時間．Step 1)〜3) は各アルゴリズムのステップにかかる計算時間を表し，Other はその他の処理にかかる計算時間を表す．また，エラーバーは標準偏差を表す．

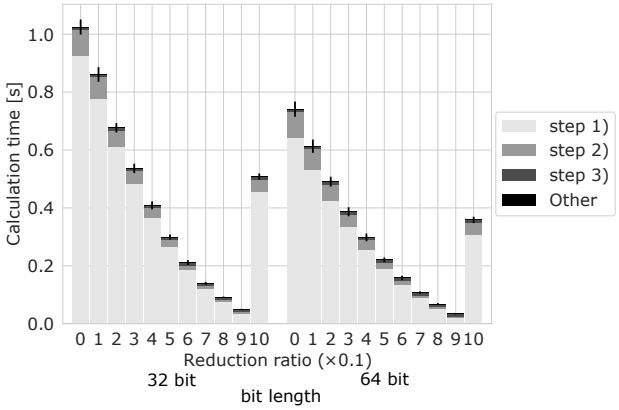

図 6 削減対象を観測数とした際の，各削減割合および浮動小数点数のビット長に対する，1 反復ごとのロボットの推定における平均計算時間．Step 1)〜3) は各アルゴリズムのステップにかかる計算時間を表し，Other はその他の処理にかかる計算時間を表す．また，エラーバーは標準偏差を表す．

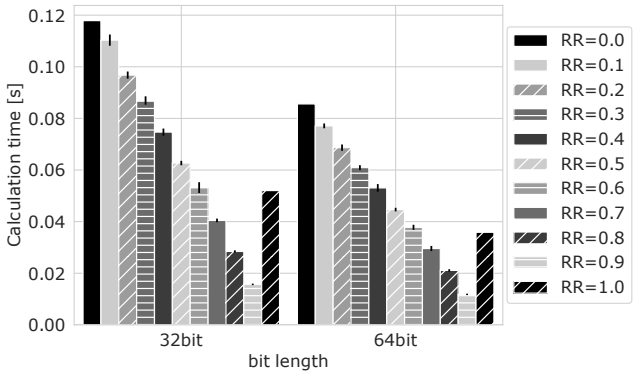

図 7 削減対象をランドマークとした際の，各削減割合および浮動小数点数のビット長に対する，アルゴリズムステップ 5) における平均計算時間．エラーバーは標準偏差を表す．

の偏りが大きく，収束に必要な反復回数が大きくなっていると考えられる．このことから，観測データ量を削減する

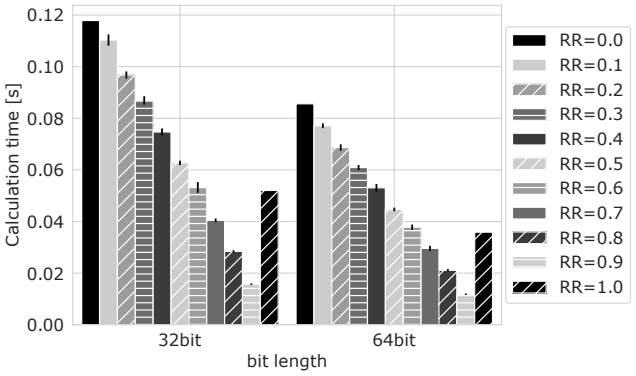

図 8 削減対象を観測数とした際の，各削減割合および浮動小数点数のビット長に対する，アルゴリズムステップ 5) における平均計算時間 ．エラーバーは標準偏差を表す．

表 2 浮動小数点数のビット長が 64 bit の場合における，削減対象をランドマークとした際の，各削減割合に対する各アルゴリズムステップでの計算時間の平均削減率および標準偏差．

Step	Reduction ratio			
	0.2	0.5	0.8	1.0
1)	0.10 ± 0.04	0.24 ± 0.03	0.41 ± 0.02	0.51 ± 0.01
2)	0.10 ± 0.02	0.26 ± 0.01	0.43 ± 0.01	0.55 ± 0.00
3)	-0.01 ± 0.02	-0.01 ± 0.02	-0.01 ± 0.02	-0.04 ± 0.03
5)	0.11 ± 0.02	0.28 ± 0.01	0.46 ± 0.00	0.58

表 3 浮動小数点数のビット長が 64 bit の場合における，削減対象を観測数とした際の，各削減割合に対する各アルゴリズムステップでの計算時間の平均削減率および標準偏差．

Step	Reduction ratio			
	0.2	0.5	0.8	1.0
1)	0.33 ± 0.02	0.70 ± 0.01	0.92 ± 0.00	0.51 ± 0.01
2)	0.36 ± 0.01	0.74 ± 0.00	0.90 ± 0.00	0.55 ± 0.00
3)	0.00 ± 0.02	-0.02 ± 0.02	-0.01 ± 0.07	-0.04 ± 0.03
5)	0.19 ± 0.01	0.47 ± 0.00	0.75 ± 0.00	0.58

にあたって，その削減方法により必要な反復回数は増減し得ることがわかり，削減割合も含めた最適な削減方法を適用することが必要である．

浮動小数点数のビット長を変更した影響は計算時間のみに見られ，32 bit の場合の計算時間は 64 bit の場合の計算時間よりも大きくなっている．これは実験に使用した言語が Python であることによる影響があると考えられ，今後 C 言語による実装などで再度検証する必要がある．平均誤差や反復回数に関しては，ビット長による影響はほとんど見られなかった．

以上の結果から，対象のアプリケーションに応じて必要な精度を満たしつつ観測データを削減することは，計算時間の観点から有効であると考えられる．

5. まとめと今後の展望

本稿では graph-based SLAM において観測データ量や浮動小数点数のビット長が推定結果や計算時間に与える影響について調査した．実験により観測データ量の削減は主に

表 4 削減対象をランドマークとした際の，各削減割合および浮動小数点数のビット長に対する，反復終了条件を満たすまでの平均反復回数および標準偏差．

Reduction ratio	32 bit	64 bit
0.0	27.0	27.0
0.1	27.3 ± 0.6	27.3 ± 0.6
0.2	27.6 ± 1.0	27.7 ± 1.2
0.3	29.1 ± 1.8	29 ± 1.7
0.4	30.1 ± 1.9	30.1 ± 1.9
0.5	32.5 ± 3.2	32.5 ± 3.2
0.6	33.5 ± 3.1	33.5 ± 3.1
0.7	35.8 ± 6.0	35.8 ± 6.0
0.8	43.8 ± 10.9	43.8 ± 10.9
0.9	36.3 ± 13.4	36.3 ± 13.4
1.0	8.0	8.0

表 5 削減対象を観測数とした際の，各削減割合および浮動小数点数のビット長に対する，反復終了条件を満たすまでの平均反復回数および標準偏差．

Reduction ratio	32 bit	64 bit
0.0	27.0	27.0
0.1	25.6 ± 0.9	25.6 ± 0.9
0.2	25.1 ± 1.7	25.1 ± 1.7
0.3	21.5 ± 2.5	21.5 ± 2.5
0.4	18.6 ± 1.8	18.6 ± 1.8
0.5	16.2 ± 1.5	17.4 ± 2.7
0.6	20.8 ± 6.6	20.8 ± 6.6
0.7	19.7 ± 7.1	19.7 ± 7.1
0.8	25.8 ± 4.9	25.7 ± 4.9
0.9	20.6 ± 5.3	20.5 ± 5.5
1.0	8.0	8.0

計算時間の観点から処理の高効率化に非常に有効であり，また Python 版の実装では浮動小数点数のビット長を小さくしても推定精度には影響がないことがわかった．今後の課題として，観測データ量の削減割合を決定する手法や観測データ量の削減方法の検討，C 言語による実装による評価などがあげられる．

謝辞 本研究の一部は，JST CREST 課題番号 JPMJCR18K1（研究課題名「エッジでの高効率なデータ解析を実現するグラフ計算基盤」）によるものである．

参考文献

[1] Durrant-Whyte, H. and Bailey, T.: Simultaneous localization and mapping: part I, *IEEE Robotics Automation Magazine*, Vol. 13, No. 2, pp. 99–110 (2006).
[2] Castellanos, J. A., Montiel, J. M. M., Neira, J. and Tardos, J. D.: The SPmap: a probabilistic framework for simultaneous localization and map building, *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 5, pp. 948–952 (1999).
[3] Fox, D., Burgard, W., Dellaert, F. and Thrun, S.: Monte Carlo Localization: Efficient Position Estimation for Mobile Robots, pp. 343–349 (1999).
[4] Dellaert, F., Fox, D., Burgard, W. and Thrun, S.: Monte Carlo localization for mobile robots, Vol. 2, pp. 1322–1328 (1999).
[5] Grisetti, G., Stachniss, C. and Burgard, W.: Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters, *IEEE Transactions on Robotics*, Vol. 23, No. 1, pp. 34–46 (2007).
[6] Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B.: FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, pp. 593–598 (2002).
[7] Grisetti, G., Kümmerle, R., Stachniss, C. and Burgard, W.: A Tutorial on Graph-Based SLAM, *IEEE Intelligent Transportation Systems Magazine*, Vol. 2, pp. 31–43 (2010).
[8] 上田隆一：詳解 確率ロボティクス Python による基礎アルゴリズムの実装，講談社 (2019).
[9] Dine, A., Elouardi, A., Vincke, B. and Bouaziz, S.: Graph-Based Simultaneous Localization and Mapping: Computational Complexity Reduction on a Multicore Heterogeneous Architecture, *IEEE Robotics Automation Magazine*, Vol. 23, No. 4, pp. 160–173 (2016).
[10] Howard, A. and Roy, N.: The Robotics Data Set Repository (Radish) (2003). `http://radish.sourceforge.net/`.
[11] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W.: G2o: A general framework for graph optimization, *Proc. 2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3–613 (2011).