生活支援ロボットの作業空間把握のための 時系列距離データに適したマップ表現と平面検出

山崎 公俊^{*1} 森 健光^{*2} 山本 貴史^{*2} 稲葉 雅幸^{*1}

Workspace Representation and Plane Detection Using Time-series Range Data for a Daily Assistive Robot

Kimitoshi Yamazaki^{*1}, Takemitsu Mori^{*2}, Takashi Yamamoto^{*2} and Masayuki Inaba^{*1}

This paper describes methods of environmental mapping and plane detection for daily assistive robots. Under the assumption that 3D information of around environment is constructed by collecting range data gradually measured, we propose an effective map representation named TCCM (Time-series Composite Cuboid Map). The method copes with temporal sequence explicitly, and map updating is efficiently performed. In addition, two plane estimation methods are proposed. Through experiments in daily environment, we ensured the effectiveness of our methods.

Key Words: TCCM, Time-series Range Data, Cuboid, Center Core, Daily Assistive Robots

1. はじめに

日常環境において物品の置き置きなどを行う支援型ロボットには、人などの動的な存在がいる状況下で安全・確実に作業を行うことが求められる。そのためには、対象物だけでなく周囲の環境形状を把握する機能が重要である。本稿では、距離センサを搭載したロボットが自身のワークスペースに関する計測データを時々刻々と蓄積していく状況下で、環境形状を効率よく表現する方法について述べる。また、人工環境の基本的な構成要素である「平面」を、効率よく検出する方法について述べる。

等身大のロボットに卓上の物品を操作させる課題を考えたとき、まず環境の三次元形状を計測し、それを行動生成に利用していくアプローチは、確実性が高い。最近では距離センサの小型化・軽量化が進み [16] [17], このアプローチが現実的なものになってきている。しかしながら、ロボットの作業空間全範囲の距離データを一度に信頼性高く取得することは難しく、既存のセンサを用いる場合は、一回の計測で観測できる範囲に限られるため、センサ自体を動かして計測を行う必要がある。すなわち、このようなセンシングを元にして動作を行うためには、センサデータを時系列で扱える枠組みが必要である。また、表現の効率化の観点では、計測情報を適切に圧縮してデータ量を削減することが望ましい。さらに、そこから平面を見つける処理

があれば、生活環境で利用するツールとして有用である。

本稿では、環境形状を多数の四角柱の集まりとして表現し、それらを時系列で扱うことのできるマップ TCCM (Time-series Composite Cuboid Map) を提案する。これは、Triebel ら [11] によって提案された MLSM (Multi-Level Surface Map) から着想を得たものであり、高い形状表現力を持ちつつもマップの構成要素数を低く抑えられるといった特長がある。本研究ではこれを発展させ、時系列のデータ管理と高速なマップの更新を可能にする。

また、この表現方式を基にして平面検出を行う二つの方法について述べる。一つは、水平面の検出方法である。ロボットや持ち運びすべき物体は多くの場合水平面上に設置されることから、それらの面を簡便に検出する。もう一つは、任意の平面を検出する方法である。生活支援ロボットが観測すべき対象には、棚などの垂直面を持つ物体が含まれていたり、平板形状の把持物について、その傾きを確認することも必要になると思われる。そのような傾きのある平面に対しては水平面検出が適用できないが、本研究では芯線表現という新たな考え方を導入することで、距離データのマッピングにおける離散化の影響を低減しつつ、効率的な探索により平面要素を見つける。筆者らの実装によれば、水平面検出と任意平面検出に要する時間はそれぞれ 0.05 [ms], 4 [ms] 程度であり、高速な検出処理が実現された。

本論文の構成は以下のようである。次章では、関連研究と本研究の貢献について述べる。3 章では、提案するマップ表現形式 TCCM について説明する。4 章では、TCCM を利用した平面検出方式について述べる。5, 6 章で検証実験について述べ、7 章でまとめる。

原 2011 年 4 月 5 日

^{*1} 東京大学情報理工学系研究科

^{*2} トヨタ自動車パートナーロボット部

^{*1} The Univ. of Tokyo

^{*2} Toyota Motor Cooperation

■ 本論文は有用性で評価されました。

2. 関連研究

三次元空間のマップ表現形式としては、ボクセル表現がよく知られている。空間を均等な立方体（ボクセル）の並びと考え、計測点をそれらへ投票していき、投票値の高いボクセルのみを集めることで環境の形状を表現する。この方式は簡便に利用できるが、空間表現のためのメモリ消費が多い。この効率化を図り、Wurm ら [12] の Octomap のように、Octree 等を導入した効率化も検討されている [2] [7]。

DEM (Digital Elevation Map, 2.5 次元マップ) は、移動型ロボットの環境地図表現として利用されている [5]。水平面に設定した基準面を格子状のセルに区切り、それぞれの格子に高さ値を入れることで、環境の凹凸を少ないメモリ消費で表現可能である。ただし、一つのセルに二つ以上の値を入れることは想定されておらず、表現力は高くない。例えば、トンネル構造などが表現できない。DEM の拡張として、Nagatani ら [4] によって提案された Sphere-DEM がある。この表現は、倒壊家屋等の内部を探索するロボットでの利用を想定しており、垂直方向の凹凸についても表現可能である。これは基準面を球面としているためであるが、一方で、人工物は多くの平面によって構成されていることを考えると、生活支援タスクにおいては基準面は平面とするほうが扱いやすいと思われる。

他の関連研究として、計測データをそのまま point cloud として保持しておくアプローチがある [6]。この利用例として、既知形状がうまく当てはまる部分の点群を除去していき、操作対象物とその周囲環境の情報を得る研究がある [10]。この方針は、既知のテーブル上に既知の物体が存在するような条件下ではうまく機能するが、未知物体や動的物体が多く存在する環境下では、多量の計測点情報を保持しつづける必要がある。また、データの時系列管理が難しい。

マップのデータ量削減については、Yguel ら [15] のように情報量損失を抑えることを目的とした研究が存在する。しかし、オンライン性の観点では、手順がやや複雑で処理時間の評価もなされていない。

本研究の焦点は、生活環境下で物体操作を行うロボットのための空間形状把握である。特に、アームが届く範囲の把握を目的とする。上述の従来研究と比較した本研究の貢献は以下である。

- (1) マップの表現形式において、Triebel ら [11] による MLSM の利点を引き継ぎ、少ない要素数で垂直方向の凹凸形状にも対応する高い表現力を実現
- (2) 計測データの三次元情報を離散化する場合、時系列が曖昧になりがちだが、それを明示的に記録する形式と、効率的なマップ更新法の提案
- (3) (1) の拡張として芯線方式を採用し、マップの解像度を擬似的に向上するとともに、計測結果から平面要素を効率よく見つける方法の提案

3. 時連結多段マップ TCCM

本章では、TCCM の構成と生成方式について述べる。なお、本稿で四角柱と称するものは MLSM におけるサーフェスに対応する。

Fig. 1 Map representation

3.1 MLSM と TCCM の要素表現

TCCM は MLSM を元に行っているため、まず MLSM について説明する。MLSM では、二次元の基準平面を格子状のセルマップとし、それぞれのセルには複数の高さ情報を入力できるようにする。これにより、垂直方向の解像度が無限の多段マップを構成する。

MLSM の基本構成を Fig. 1 (a) に示す。計測点 $\mathbf{x}_i = (x_i, y_i, z_i)$ があるセルに投票されると、その z 座標が μ として設定される。さらに、計測誤差を見積もった変数 σ が情報として付加され、一つのサーフェスが構成される。もし、近い z 座標を持つ計測点があるセルに存在した場合、二つのサーフェスは結合され、奥行き値 d が定義される。一方で我々は、MLSM のサーフェスをより簡略化し、一つの計測点から一つの四角柱 (cuboid) を構成する。この四角柱は、Fig. 1 (b) に示すように高さ値 u と固定長の奥行き値 d の二変数で表される。

四角柱生成の手順は以下のとおりである。まず、ある計測点 $\mathbf{x}_i = (x_i, y_i, z_i)$ が所属すべきセルを x_i, y_i の値から特定する。そして、 z_i 値を高さと情報 u とし、厚み d を指定することで、そのセルに一つの四角柱を追加する。

なお、MLSM におけるサーフェスの結合条件は、サーフェス同士の距離が閾値以下となることであったが、本方式では、四角柱が直接重なり合わない限り接続は行わないこととし、四角柱同士の関係をより厳密に扱う。具体的には、四角柱の内部もしくは上部に新たな点 $\mathbf{x}_j$ が計測された場合、以下に従い四角柱の更新を行う。

$$(u, d) = \begin{cases} (z_i, d + z_i - z_j), & \text{if } z_i \geq z_j \\ (z_j, d + z_j - z_i), & \text{if } z_i < z_j \text{ and } z_i \geq (z_j - d) \end{cases} \quad (1)$$

$z_i < (z_j - d)$ の場合は、 $\mathbf{x}_j$ に関して新たに四角柱を一つ生成する。なお、ここではすべての点に対して同じ d を付与すると想定したが、これをセンサの計測間隔などから適応的に決めることも考えられる。

3.2 TCCM のコンセプト

ある時刻 t においてセンサから得られた一セットの計測点群から、一つのマップ m_t を構築する。以後、このマップを部分マップと呼ぶ。一方で、部分マップを時系列に沿って必要数 (n 個) だけ合わせたマップを構築する。以後、このマップを統合マップと呼び、 $M_t (= \sum_{i=t-n+1}^t m_i)$ と記す。Fig. 2 に、三セットの計測点群から統合マップを構成する場合の概念図を示す。

Fig. 2 The concept of TCCM

TCCMにおける四角柱は、C++の構造体表記に従えば以下のように定義できる。

```
struct TCCM {
    int pnum, cls;
    float u, d;
    list < int > msnlist;
    float p[3], P[3][3], s[3], S[3][3];
};
```

ここで、*pnum* には四角柱に投票されている計測点数を記録する。*cls* は四角柱が所属するクラス番号である（4章参照）。*u*, *d* は前述したように四角柱の上面高さと厚みを表す。また、四つの浮動小数点配列は芯線情報を保持するものであり、詳細は次章で述べる。

msnlist は、部分マップでは利用せず、統合マップの四角柱が保持するものである。その四角柱がどの部分マップの四角柱から構成されたかを、部分マップのシリアル番号（1 ... *k*）として記録しておく。この構成により、統合マップのすべての四角柱について構成元となる部分マップを知ることができる。さらに、そこから部分マップの各四角柱を調べることによって、統合マップの四角柱のどの領域が、どの部分マップのどの四角柱に寄っているかまで特定することができる。すなわち、計測領域の厳密な時刻管理が可能になる。

3.3 統合マップ M_t の生成方法

統合マップを生成するには二つの方法がある。（1）現在時刻 *t* の部分マップが生成されるごとに、*n* 個の部分マップを集め、すべて足し合わせる（全加算）、（2）現在時刻 *t* の部分マップを統合マップへ加えたあと、時刻 *t - n* の部分マップに相当する四角柱群を統合マップから差し引く（入れ替え）。

3.3.1 部分マップの全加算

すべての部分マップを足し合わせて統合マップを生成する方法は、シンプルであり実装が容易である。一方で、時刻が変わる度に *n* 個の部分マップを加算する必要があるため、次に述べる入れ替え方式と比較し、処理の効率が悪い。

以下で、加算操作について二つの部分マップを用いて説明する。まず双方の部分マップが含む四角柱群を統合マップ用のセルへコピーする。そして、四角柱 $C_A : (u_A, d_A)$ が四角柱 $C_B : (u_B, d_B)$ と重なることがあればそれらを融合する。重なり方と融合方法には、四つのパターンがある（Fig. 3）。

Fig. 3 Addition of cuboids

（1） $u_A \geq u_B$ かつ $d_A < u_B$ のとき、

$d_A = (u_A - u_B) + d_B$ として C_B を消去

（2） $u_A < u_B$ かつ $d_B < u_A$ のとき、

$d_B = (u_B - u_A) + d_A$ として C_A を消去

（3） $u_A > u_B$ かつ $d_A \geq d_B$ のとき、 C_B を消去

（4） $u_A < u_B$ かつ $d_A \geq d_B$ のとき、 C_A を消去

なお、片方の四角柱を消去する前に、それが持つ *pnum*, *msnlist* の中身をもう一方の四角柱へ追加しておく。

3.3.2 部分マップの入れ替え

統合マップを効率的に生成する方法として、減算を取り入れる。基本的には、最新の部分マップを統合マップに加算した後、最も古い部分マップに相当する領域を統合マップから取り除いていく。ただし、減算の場合は計算前後の整合性が取れている必要があるため、前処理として、古い時刻の情報より新しい時刻の情報を優先するように、過去の部分マップを変形しておく。

処理手順は以下のようである。

（1）前処理：部分マップの変形

統合マップの一つの四角柱は、複数の部分マップの四角柱の重ね合わせで構成される。この場合に、古い時刻の四角柱を統合マップから単純に差し引くと、それより新しい時刻での計測点が存在する部分も削り取ってしまうことがある。これに対処するため、時刻の異なる四角柱の共通部分をあらかじめ取り除く処理を行う。まず、最新の部分マップ m_t が入力される。次に、統合マップ M_t が持つ四角柱を順に読み出し、 m_t の四角柱との重なりを調べる。もしあった場合に、それが持つ *msnlist* から構成元の部分マップ番号を参照し、それらの部分マップと最新部分マップ m_t の四角柱と重なる領域については、構成元の部分マップの四角柱を変形して重複部分をなくす（Fig. 4 右下）。ただし、*pnum*, *msnlist* は元の値を維持する。

以上により、部分マップの四角柱が一切重なり合わずに統合マップが構築されるようにする。この処理は、計測データが実在する部分領域を削り取ることになるため、個々の部分マップに着目する限りは不整合が起きる。しかし、古い部分マップと新しい部分マップを足し合わせた状態では情報に欠落は起きておらず、ロボットが統合マップを衝突検知等に利用する段階では問題にならない。

（2）部分マップの加算

3.3.1 項と同様の手順で、統合マップ M_t に部分マップ m_t

Fig. 4 Subtraction of cuboids

を加算する。

(3) 統合マップの減算

統合マップから、最も古い部分マップ m_{t-n} との共通部分を減じる。(1)のマップ変形処理により、取得時刻の異なる部分マップが重なり合うことはなくなるため、最も古い部分マップの存在する分だけ統合マップの四角柱を消去すればよい。統合マップの四角柱を $C_M : (u_M, d_M)$ 、部分マップの四角柱 $C_m : (u_m, d_m)$ と表すと、減算は下記の三つのパターンにより行われる (Fig. 4)。

(1) $u_M = u_m$ のとき、

$u_M = u_M - d_m$, $d_M = d_M - (u_m - d_m)$ とする。

(2) $d_M = d_m$ のとき、 $d_M = d_M - (u_m - d_m)$ とする

(3) $u_M > u_m$ かつ $d_M < d_m$ のとき、上下二つに分割する。

上側の四角柱については $^1u_M = u_M$, $^1d_M = u_M - u_m$

とし、下側の四角柱については $^2u_M = u_m - d_m$,

$^2d_M = (u_m - d_m) - (u_M - d_M)$ とする

この方式は、全加算方式と比べて効率的である。なぜならば、全加算方式ではすべての部分マップのすべての四角柱を走査する必要があるのに対し、入れ替え方式では、変形の対象となる四角柱の走査領域を、最新の部分マップの四角柱が存在するセルのみに限定できるためである。このため、5.2 節で示すように、揺動型測域センサを用いた場合の統合マップ生成を高速に行うことができ、その処理時間が部分マップの数にあまり依存しないといった利点がある。

4. 平面検出

4.1 水平面検出

生活環境で物品の置き置きを行うロボットにとって、水平面の検出機能は極めて有用である。なぜなら、ロボットや持ち運びすべき物体は多くの場合、水平面上に設置されるためである。

TCCM は、MLSM の表現形式を一部引き継いでおり、高さ方向の解像度に制限がない。そのため、基準面と同じ傾きの平面を容易に見つけることができる。本研究では、基準面を水平に設定する。なお、文献 [11] では、同様に設定の元で 5 もしくは 8 近傍のサーフェスを参照し現在サーフェスの平面性を判定する方式が提案されている。本研究ではこれを拡張し、隣り合う四角柱の u 値を再帰的に参照していくクラスタリングアルゴリズムを用いる。この理由はセンサの計測誤差を考慮したためであり、根拠となる実データは 6.1 節に示す。

基本的手順は以下のとおりである。画像処理においてよく行

Fig. 5 Center core

われるのと同様に 8 近傍のセルを探索していき、その高さ値 u の差分が所定の閾値以下であれば同じクラスラベルを付与する。3.2 節で示した変数 cls はここで用いるものである。

4.2 芯線の導入

本節では、任意の傾きの平面を検出するための前段階として、四角柱へ付加する芯線情報について説明する。

芯線とは、四角柱内に存在する点群をだ円体とみなした場合の長軸を指す。これは、点群の共分散行列を固有値分解した結果から得ることができる。この表現を用いればセルマップの解像度を擬似的に向上することができ、四角柱の内部で点群の分布に偏りがある場合に有用である。

点群の平均・共分散はそれぞれ、

$$\mathbf{p}_N = \frac{\mathbf{s}_N}{N}, \mathbf{P}_N = \frac{\mathbf{S}_N + \mathbf{p}_N \mathbf{s}_N^T}{N} \quad (2)$$

として計算される。ここで、 T は転置である。 N は点の数であり、 $\mathbf{s}$, $\mathbf{S}$ はそれぞれ、

$$\mathbf{S}_N = (N-1)\mathbf{p}_{N-1} + \mathbf{x}_N \quad (3)$$

$$\mathbf{S}_N = \mathbf{S}_{N-1} + \mathbf{x}_N \mathbf{x}_N^T$$

である。 $\mathbf{x}$ は点の三次元座標である。この計算方法によれば、計測点が追加されるたびに平均・共分散の逐次更新が可能である。さらに、Fig. 5 に示すように、四角柱の変形が起こった場合にも以下のような演算によって分布を調整することができる。

●四角柱 C_1 (内部分布 $N(^1\mathbf{p}, ^1\mathbf{P})$) と C_2 (内部分布 $N(^2\mathbf{p}, ^2\mathbf{P})$) の加算：

$$\mathbf{p} = (^1\mathbf{s} + ^2\mathbf{s}) / (^1N + ^2N) \quad (4)$$

$$\mathbf{P} = (^1\mathbf{S} + ^2\mathbf{S}) / (^1N + ^2N) - ^1\mathbf{s} ^2\mathbf{s}^T$$

●四角柱 C_1 (内部分布 $N(^1\mathbf{p}, ^1\mathbf{P})$) から C_2 (内部分布 $N(^2\mathbf{p}, ^2\mathbf{P})$) の減算：

$$\mathbf{p} = (^1\mathbf{s} - ^2\mathbf{s}) / (^1N - ^2N) \quad (5)$$

$$\mathbf{P} = (^1\mathbf{S} - ^2\mathbf{S}) / (^1N - ^2N) - ^1\mathbf{s} ^2\mathbf{s}^T$$

一方で、上述した演算のほか、四角柱が上下に分割される場合へも対応する必要がある。分割が起きる状況は、3.3.2 項で述べた入れ替え方式における以下の二つである。

(1) 前処理において、最新の部分マップにより過去すべての部分マップを変形するとき (Fig. 6 上段)

Fig. 6 Two patterns of division of cuboid

(2) 減算処理において、最も古い部分マップを用いて統合マップの四角柱を変形するとき (Fig. 6 下段)

前者については、分割対象となった部分マップ (part map 1) の四角柱の分布を二つに分割する。その比率は厳密に決めてもよいが、統合マップ生成時にはどのみち合成されるため、適当でよい。すなわち、四角柱の形状については重ね合わせがなくなるように part map 2 の四角柱に従って変形を施すが、内部分布については便宜上二つに分けるだけで、その比率は問わない。一方で、後者の場合は、分割対象となった統合マップの四角柱について、その構成元となる部分マップの四角柱をすべて検索する。そして、分割する側の四角柱 (part map 2 に所属) を取り除いたあと、統合マップの四角柱を二つに分け、それぞれで分布を計算しなおす。

4.3 芯線を用いた平面検出

距離データとして得られた点群情報から平面を検出する方法は、従来から提案されている。その一つは、ランダムに選択した少数の三次元点群から平面パラメータを求め、そこへ当てはまる点群を調べる方法である [9] [13]。また、近傍の三次元点でペアを構成し、そこから同属領域を拡張していく region growing のアプローチもある [1] [8]。いずれも、三次元点群が入力となる。

一方で本研究の方針によれば、前節の処理により四角柱が芯線情報を保持している。この状態から平面検出を行うことで処理を効率化する。算出手順を以下に述べる。まず、 l 本の芯線をランダムに選び出し、そのうちの 2 本をペアとして以下のような計算を行う。

$$\begin{aligned} \mathbf{a} &= \mathbf{j}\mathbf{p} - \mathbf{i}\mathbf{p} \\ {}^i\mathbf{b} &= {}^i\mathbf{n} \times \mathbf{a} \\ {}^j\mathbf{b} &= {}^j\mathbf{n} \times \mathbf{a} \end{aligned} \quad (6)$$

ここで、 $\mathbf{n}$ は芯線方向のベクトルである。もし ${}^i\mathbf{n}, {}^j\mathbf{n}$ が同一平面に載るならば、 ${}^i\mathbf{b}, {}^j\mathbf{b}$ は同じか真逆のベクトルになる。Fig. 7 に概念図を示す。

このペアリングを l 本のすべての組み合わせにおいて行い、それらから得られる外積ベクトル $\mathbf{b}$ について以下を計算する。

Fig. 7 Plane estimation based on a set of centercore

$$\left\{ \cos^{-1} \frac{{}^i\mathbf{b} \cdot {}^j\mathbf{b}}{|{}^i\mathbf{b}||{}^j\mathbf{b}|} \text{ or } \cos^{-1} \frac{-{}^i\mathbf{b} \cdot {}^j\mathbf{b}}{|{}^i\mathbf{b}||{}^j\mathbf{b}|} \right\} < \text{threshold} \quad (7)$$

この結果が是となった場合、平均 $\hat{\mathbf{b}}$ を求め、 $\hat{\mathbf{b}}^k \mathbf{p} + d = 0$ として d を求めることで、平面の四つのパラメータを決定する。 ${}^k\mathbf{p}$ は、本研究においては最も長い芯線に関する平均値とした。

次に、マップを構成するすべての芯線を順番に読み出し、この平面に属するものを探す。そこでは、以下の二種類の閾値により、当てはまりの判断を行う。

- 角度の閾値：その芯線の向き $\mathbf{n}$ が $\hat{\mathbf{b}}$ に対してほぼ垂直になっている
- 距離の閾値：芯線の中心 $\mathbf{p}$ を平面方程式に代入したときの絶対値が十分小さい

ただし、このようにして得られた芯線群は、複数の平面領域を含んでいる可能性がある。そこで、その芯線群をクラスタリングにかける。同じクラスへの所属とするための基準は、二つの芯線同士の距離が閾値以下となることである。この処理は、基本的には水平面検出と同じく再帰探索により行うが、新たに選んだ芯線の所属するセルが現在の芯線の所属するセルに近いかどうかを調べる前処理を入れることで、より効率化できる。

最後に、平面の当てはまりを判断するため、クラスタ内のすべての芯線情報を用いて共分散行列を計算し、その固有値を降順に並べた $\lambda_1, \lambda_2, \lambda_3$ について以下を計算する。

$$val = \frac{\lambda_1 \lambda_2}{\lambda_3} \quad (8)$$

この値が所定の閾値より大きいほど検出結果の信頼性は高いと考える。ただし、一方向のみに細長い領域で高い値が出ることを防ぐため、 λ_3 が 1.0 より小さい場合は $\lambda_3 = 1.0$ とする。

5. TCCM 生成の検証実験

5.1 設定

水平走査が可能な測域センサ (UBG-04LX-F01 [16]) をチルト軸による上下揺動機構と組み合わせた揺動センサシステムを構成し、等身双腕型ロボットの頭部に搭載した。ロボット直立時のセンサ揺動中心は高さ 1,550 [mm] とした。ロボットの身体構成は、二輪駆動型の車体に、一自由度の腰、三自由度の首、各七自由度の双腕アームである。

Fig. 8 (A)-1,2, (B)-1,2 に、対象環境の例を二つ示す。

(A) 物品が載せられたテーブル

天板の高さが 700 [mm] のダイニングテーブルを設置し、いくつかの物体をその上に置く。ロボットはテーブルの前面に立ち、揺動中心を斜め下向きとして、揺動振幅 30 [deg]

Fig. 8 Experimental conditions

Fig. 9 Processing time for map generation

で計測を行う。

(B) 物品が入れた引き出し

幅 900 [mm], 奥行き 300 [mm], 高さ 1,200 [mm] の五段タンスを設置し, その三段目にいくつかの物体を入れておく。ロボットはそのタンスの前面に立ち, 引き出し操作を行い, その中身を観測する。腰軸, 首軸を動かして揺動中心が引き出し内部に向くようにし, 揺動振幅 30 [deg] で計測を行う。

なお, 以降に述べる処理の実行環境は, 2.4 [GHz CPU], 仮想 Linux OS 上で, オフラインで行った。

5.2 マップ構築の処理時間評価

Fig. 8 (A)-1 に示す環境で取得した 5,000 回分の計測データをを用いて, マップ構築実験を行った。Fig. 9 は, その処理時間をスキャン数で割った値をプロットしたものである。3.3 節で述べ

Fig. 10 Workspace mapping with time-series data

た二種類のマップ構築方式について, セルサイズを 10 [mm] から 40 [mm] の間で変えて実行した。横軸は一つの統合マップに含まれる部分マップの数であり, 縦軸は処理時間を示している。

本実験では, ロボットのワークスペース内 (半径約 1 [m]) において幅 50 [mm] 以上の物体を検知することを目的に, 一回の揺動にかかる時間を 1,575 [ms] に設定した。測域センサのデータ取得間隔は 28 [ms], 点数は 569 点であり, この条件下で一回の揺動時に取得できる計測数は 35 から 40 であった。

Fig. 9 が示すように, 計測数 40 の場合は, 方式, セルサイズによらず計測間隔 (28 [ms]) 以上の速度でマップの更新は可能だが, 3.3.2 項で述べた入れ替え方式のほうが高速である。具体的には, セルサイズ 20 [mm] の場合に平均 1.35 [ms] で統合マップを更新することが可能であった。また, 統合マップ生成に利用する計測数をさらに増やしても処理時間はほとんど変わらなかった。一般に, セルサイズを小さくするほど密な環境形状を得るのに多くの計測データを必要とするため, 入れ替え方式が持つこの性質は極めて有用である。

なお, Fig. 8 (A)-3 の例では, 569×40 の計測点が 1,300 程度の四角柱へ変換されており, 要素数は 17 分の 1 になった。この例のほかにも, 物体の置き場所を変える, 人が近づいて手を差し出す, 計測中にロボットがアームをテーブル上に伸ばす, などの状況で実験を行い, マップ生成が上述の実験と同程度の処理速度で実現可能なことを確認した。

5.3 動的環境における TCCM の効果

Fig. 10 中央の画像は, ボットを持った人間がテーブル上にあるコップへ注ぎ動作をしている様子である。Fig. 10 の両端の画像は, その人間がテーブルの横に直立した状態から中央図の姿勢になるまでの計測データから, TCCM を構築した結果を示している。左図では, 現在スキャン n から過去 30 (時間にすると 1 秒弱) のスキャンが用いられており, 右図ではすべての計測データが利用されている。動的環境においては, 過去に得られた計測データを適切に消去していかないと, 右図点線領域が示すように, 現在時刻において観測がない領域にデータが多く残る。これは, マップを利用した物体の切り出しやロボットの行動計画を立てる際, 妨げになる。提案方式は, マップの減算を導入することでそれを回避することができている。

6. 平面検出の検証実験

6.1 水平面検出

Fig. 11 は, Fig. 12 左の点線に示すような幅 1,200 [mm], 高さ 700 [mm] のテーブルの天板を計測し, その結果をロボット正面から表示したものである。縦横比が大きく異なることに

Fig. 11 A measurement result of the top board of a table

Fig. 12 Scanning situation

注意して見ると、平面がおおむね計測できているものの、最大 20 [mm] 程度のばらつきがあることが分かる。この要因は、揺動のキャリブレーションやセンサ自身の計測精度などが考えられる。これらの影響を完全になくすことは難しく、誤差を許容できる平面検出法が必要である。

例えばボクセル形式のような高さ方向にも離散化を行う表現では、高さ値が若干異なる計測データに対して異なるボクセルが割り当てられる場合があり、それらを考慮に入れた水平面検出を行う等の工夫が必要である。一方で、TCCM をベースとする方法では、単純な再帰的探索により実現できる。

Fig. 8 の (A)-3, 4 と (B)-3, 4 は、4.1 節に述べた方法により水平面を検出し、それに該当する四角柱を取り除いた実験例である。テーブルの天板や引き出しの底板部分が除かれ、物体が置かれている部分を分離できていることが分かる。なお本実験では、隣り合う四角柱同士の u 値の差分閾値は 10 [mm] とした。一枚の検出に要した処理時間は約 0.05 [ms] であった。この処理時間は、次節に示す芯線ベースの処理よりも 80 倍程度高速であり、適用対象が水平面に限られるとしても十分な利用価値があると思われる。

6.2 芯線情報を用いた平面検出法の評価

本節では、4.3 節で述べた平面検出法の性能評価をシミュレーションデータを用いて行う。Fig. 13 に示す四種類の仮想環境を用意し、揺動型センサによる仮想計測を行ってデータを取得した。それらのデータからマップ生成と平面検出を行い、処理時間、平面推定精度を評価した。なお、各計測点には正規分布 $N(0, \sigma^2)$ のノイズを加えた。さらに本シミュレーションでは、Fig. 11 のような実データを参考にし、各点に与えるノイズのほかに、抽出したい平面上の点群には波長 500 [mm]、振幅 σ [mm] となるサイン波で変形を加えた。テーブル天板のスキニング結果へのノイズ付加の例を Fig. 14 に示す。

比較手法として、Web 上で公開されている点群処理ライブラリ PCL (Point Cloud Library) [18] の平面検出関数を用いた。このプログラムは、ランダムサンプリングにより選んだ点群に平面パラメータを当てはめることで、もっともらしい平面を検

Fig. 13 Simulated scanning situation

Fig. 14 Simulated scan data with measurement error

出するものである。提案手法については、検出処理の試行回数を 50 回、10 回とした二種類を行い、それぞれの試行の中で式 (8) の値が最も大きかった平面を選択した。なお、ここで言う試行回数とは、4.3 節で示した角度と距離の閾値を越える組み合わせを 50 ないし 10 個見つけることであり、実際のサンプリング回数はそれよりも多い。各種パラメータについては経験的に、線分のランダム選択の本数を $l = 3$ 、角度と距離の閾値をそれぞれ 20 [mm]、10 [deg] とした。

Fig. 15 は、Fig. 13 (B), (D) に示すタンスの前面の検出結果をまとめたものである。横軸は、計測点に混入させた正規分布誤差の σ を示しており、縦軸については左から、推定位置の平均、推定位置の偏差、処理時間である。また、Table 1~3 に各手法・条件における検出結果の一部を示した。各セルには、平面検出を 20 回行ったときの平均・標準偏差の値を並べて示した。

テーブル天板位置の真値は 700 [mm]、タンス前板位置の真値は 600 [mm] であった。提案手法は、検出平面の位置の繰り返し精度としては点ベースの平面検出手法に劣るが、主要な平面を見つける目的には試行回数 10 回としても十分であった。また、Table 4 では PCL、提案手法 50 回試行、提案手法 10 回試行の三種類での処理時間を示しており、試行回数 10 において提案手法は PCL の 10 倍高速であった。すなわち、マップ生成にかかる時間 (5.2 節参照) と合わせてもおおむね 5 [ms] 程度で平面検出までが可能である。

6.3 芯線情報を用いた平面検出

Fig. 16 は、Fig. 12 右図のようにタンスを前にした状態で実

Fig. 15 Plane detection results on (B) and (D)

Table 1 Plane detection results (PCL [18])

	(A)table	(B)shelf	(C)table	(D)shelf
error[mm]	ave,dev[mm]	ave,dev[mm]	ave,dev[mm]	ave,dev[mm]
$\sigma = 0.0$	698.46, 0.00	599.37, 0.00	698.27, 0.00	599.73, 0.00
$\sigma = 2.5$	698.02, 0.15	599.37, 0.02	697.63, 0.15	609.31, 4.08
$\sigma = 5.0$	697.91, 0.11	599.23, 0.26	696.60, 0.14	601.13, 1.82

Table 2 Plane detection results (proposed method, 50 trials)

	(A)table	(B)shelf	(C)table	(D)shelf
error[mm]	ave,dev[mm]	ave,dev[mm]	ave,dev[mm]	ave,dev[mm]
$\sigma = 0.0$	699.90, 0.34	600.00, 0.01	699.86, 0.61	600.05, 0.68
$\sigma = 2.5$	699.14, 3.21	599.15, 1.46	699.08, 3.08	600.24, 2.65
$\sigma = 5.0$	699.49, 4.41	597.94, 3.06	700.05, 4.04	598.15, 3.16

Table 3 Plane detection results (proposed method, 10 trials)

	(A)table	(B)shelf	(C)table	(D)shelf
error[mm]	ave,dev[mm]	ave,dev[mm]	ave,dev[mm]	ave,dev[mm]
$\sigma = 0.0$	699.90, 0.34	600.00, 0.01	699.86, 0.61	600.37, 2.78
$\sigma = 2.5$	698.23, 3.20	598.67, 2.25	700.51, 5.20	600.99, 2.86
$\sigma = 5.0$	700.49, 7.74	599.21, 4.86	699.17, 5.79	599.26, 5.51

Table 4 Processing time for plane detection

	(A)table	(B)shelf	(C)table	(D)shelf
(error[mm])	[ms]	[ms]	[ms]	[ms]
pcl($\sigma = 0.0$)	26.72	25.13	24.84	37.51
pcl($\sigma = 2.5$)	22.34	24.98	24.83	28.85
pcl($\sigma = 5.0$)	23.66	25.85	27.42	29.97
TCCM-based, 50(0.0)	28.59	14.44	25.70	15.44
TCCM-based, 50(2.5)	16.96	14.10	15.83	15.96
TCCM-based, 50(5.0)	17.78	12.75	16.16	14.12
TCCM-based, 10(0.0)	5.82	3.50	4.87	3.05
TCCM-based, 10(2.5)	3.47	2.82	3.18	3.58
TCCM-based, 10(5.0)	3.07	2.58	3.67	3.67

際に取得した計測データからマップを生成し、その芯線を表示したものである。ここでは、1,600 程度の四角柱が生成された。上が正面図、下が横面図である。y = 0, z = 1,000 の付近で縦長の線が見られるが、これは多くの四角柱が連結されて一つの長い芯線が得られた部分である。

なお、このタンスは引き出し部分に取っ手があるため、z 方向には一定の間隔ごとに突起が観測される。また、ロボットの車体に若干の傾きがあったため、床面やタンスの計測結果は垂直・水平ではない。芯線表現は、セルマップにより単純に離散化した場合と比べ、そのようなデータの質を的確に表現できて

Fig. 16 Measurement results of the top board of a table (top: front view, bottom: side view)

いる。

Fig. 17 は、この計測結果に対して平面検出処理を 10 試行し、そのうち二つの結果の横断面図を示したものである。それぞれで求めた平面に対して、当てはまりがよかった芯線の中心座標を表示している。両方ともにタンスの前面における計測データを捉えているが、左側には取っ手部分が含まれており、平面検出の結果としては誤差が大きい。

Fig. 17 左は、式 (8) において 0.2222 を示した。一方で、右の例は 0.3472 となり、他の推定結果と比べ最も高い値となった。一つの平面を発見するのに要した時間は、平均で 0.4 [ms] であった。各種条件は 6.2 節と同様に、線分のランダム選択の本数を $l = 3$ 、角度と距離の閾値をそれぞれ 20 [mm], 10 [deg] とした。

なお、この実験において、本方式を用いた場合のデータ量の推移をまとめると以下になる。まず、もとの計測点の数は $569 \times 40 = 22,760$ である。これが 1,600 程度の四角柱に変換された。そのなかで 3 点以上の計測点を含んでおり芯線が計算できる四角柱は 350 個程度であり、平面推定はこれらの中

Fig. 17 2 examples of plane estimation

Table 5 Plane detection results

yaw angle	pitch angle	processing time	maximum val
20 [deg]	0 [deg]	4.354 [ms]	0.2996
40 [deg]	0 [deg]	4.530 [ms]	0.3900
60 [deg]	0 [deg]	3.377 [ms]	0.4382
80 [deg]	0 [deg]	3.351 [ms]	0.1830
0 [deg]	20 [deg]	3.946 [ms]	0.3464
0 [deg]	40 [deg]	4.156 [ms]	0.2012
0 [deg]	60 [deg]	4.289 [ms]	0.6344
0 [deg]	80 [deg]	4.106 [ms]	0.7483

で行われる。すなわち、平面推定の前段階でデータの要素数は1/65にまで削減されており、ランダムサンプリングに基づく処理の高速な実現に寄与している。

Table 5 は、もとのタンス計測結果を y 軸まわり (pitch), z 軸まわり (yaw) にそれぞれ回転した後、そのデータから平面検出を行った結果を表にしたものである。ここでも、ランダムサンプリングに基づく平面検出処理を10回試行し、最も当てはまりのよい結果を選んだ。我々の経験では式 (8) の val の値が0.15を越えるようであれば当てはまりのよい結果が得られていると判断できるものであったが、本実験ではすべてのケースでそれ以上の値となった。この実験の狙いは、芯線方式が垂直な平面以外でも適用可能かを調べるためであった。一般に、対象平面がセルの並びに沿っておりその傾きが垂直であるほど、整って並んだ長い芯線が得やすいため、提案手法はよく機能すると想像できる。一方で、水平面などでは芯線が細かく途切れるため、平面推定が安定して行えないのではないかとこの疑問が生じる。しかしながら、Table 1 では、どの傾きの場合でも比較的安定した値が得られた。提案手法は、入力される芯線群が必ずしも縦長でなくとも、本研究で対象としたレンジデータに対しては平面検出法として機能することが示された。

Fig. 18 は、複数の平面がある環境下での検出実験の結果である。左列は、テーブル前に立たせたロボットに対して、白いプレートを持った人がそれを斜めに傾けながら差し出した様子とその平面検出結果である。右列では、テーブル上に約600 [mm] 四方の箱を置いている。それぞれのケースにおいて val の値が高くなる部分が二ヶ所観測されたが、中段と下段の結果は、それぞれの箇所でもっと高い val 値を持つ平面に対し、距離が20 [mm] 以下となる点のみを表示したものである。これらの結果より、提案手法は環境を構成する主要な平面を複数見つけることができ、様々な傾きに対応できることが分かった。

7. おわりに

本稿では、時々刻々と取得されていくセンサデータを対象にしたマップ表現法について述べた。環境形状を多数の四角柱の

Fig. 18 Results of plane detection

集まりとして表現し、それらを時系列で扱うことのできる枠組みを提案した。また、表現の拡張として芯線の利用を提案し、それを応用した平面検出方式についても述べた。

今後の課題として、提案方式により生成したマップを用いて、ロボットの生活支援行動を実現し評価することがある。また、本研究で利用したものと異なる計測範囲を持つようなセンサについても、提案手法を適用可能性を評価していく。

参考文献

- [1] J. Deschaud: "A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing," 3DPVT, 2010.
- [2] N. Fairfield et al.: "Real-time SLAM with octree evidence grids for exploration in underwater tunnels," Journal of Field Robotics, pp.3-21, 2007.
- [3] M. Hevert, C. Caillas, E. Krotkov, I.S. Kweon and T. Kanade: "Terrain Mapping for a Roving Planetary Explorer," Proc. of IEEE Int'l Conf. on Robotics and Automation, vol.2, pp.997-1002, 1989.
- [4] K. Nagatani et al.: "Three-dimensional Localization and Mapping for Mobile Robot in Disaster Environment," Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, pp.3112-3117, 2003.
- [5] F. Nashashibi et al.: "3-D Autonomous Navigation in a Natural Environment," Proc. of IEEE Int'l Conf. on Robotics and Automation vol.1, pp.322-439, 1994.
- [6] A. Nuchter, K. Lingemann, J. Hertzberg and H. Surmann: "6D SLAM - 3D mapping outdoor environment: Research articles," Journal of Field Robotics, vol.24, no.8-9, pp.699-722, 2007.
- [7] P. Payeur, P. Hebert, D. Laurendeau and C.M. Gosselin: "Probabilistic Octree Modeling of a 3-D Dynamic Environment," Proc. of IEEE Int'l Conf. on Robotics and Automation, pp.1289-1296, 1997.
- [8] J. Poppinga et al.: "Fast Plane Detection and Polygonalization in Noisy 3D Range Images," Int'l Conf. on Intelligent Robots

- and Systems, 2008.
- [9] R.B. Rusu et al.: Perception for Mobile Manipulation and Grasping using Active Stereo. Humanoids, 2009.
- [10] R.B. Rusu, N. Blodow, Z.C. Marton and M. Beetz: "Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Domestic Environment," Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, pp.1-6, 2009.
- [11] R. Triebel, P. Pfaff and W. Burgard: "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing," Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, pp.2276-2282, 2006.
- [12] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss and W. Burgard: "Octomap: A Probabilistic, Flexible and Compact 3D Map Representation for Robotic Systems," Proceedings of the ICRA 2010, 2010.
- [13] M. Yang and W. Forstner: Plane Detection in Point Cloud Data. Univ. of Bonn, 2010.
- [14] C. Ye and J. Borenstein: "A new terrain mapping method for mobile robots obstacle negotiation," Proc. of the UGV Technology Conf. at the 2002 SPIE AeroSense Symposium, pp.21-25, 2002.
- [15] M. Yguel, C. Keat, C. Braillon, C. Laugier and O. Aycard: "Dense Mapping for Range Sensors: Efficient Algorithm and Sparse Representation," Proc. of Robotics: Science and Systems, pp.129-137, 2007.
- [16] SOKUIKI sensor, <http://www.hokuyo-aut.jp/02sensor>
- [17] MESA Imaging, SR4000, <http://www.mesa-imaging.ch>
- [18] Point Cloud Library, <http://www.pointclouds.org>

山崎公俊 (Kimitoshi Yamazaki)

2002 年筑波大学第三学群工学システム学類卒業, 2007 年筑波大学大学院システム情報工学研究科博士課程修了。博士 (工学)。2006 年日本学術振興会特別研究員。2007 年より東京大学特任助教。2010 年より科学技術振興機構さきかけ研究者 (兼任)。知能ロボットの視覚と動作計画に興味を持つ。情報処理学会, 日本機械学会, IEEE の会員。 (日本ロボット学会正会員)

森 健光 (Takemitsu Mori)

2000 年早稲田大学理工学部卒業。2002 年東京工業大学大学院総合理工学研究科修士課程修了。同年松下電器産業 (株) (現, パナソニック) 入社。工場管理システムの開発に従事。2004 年トヨタ自動車 (株) 入社。愛知万博ロボットの隊列制御, モビリティロボットの開発を経て, 現在介護支援ロボットの開発を行っている。 (日本ロボット学会正会員)

山本貴史 (Takashi Yamamoto)

1990 年京都大学工学部卒業 (機械系)。1992 年京都大学大学院工学研究科修士課程修了 (数理工学)。同年トヨタ自動車 (株) 入社。自動車用ユニットおよび愛知万博ロボットの制御システム開発を担当。2006 年 TOYOTA MOTOR EUROPE NV/SA 出向。2009 年よりトヨタ自動車 (株) パートナーロボット部 先行開発室にて人共存ロボットの研究開発に従事。計測自動制御学会の会員。

稲葉雅幸 (Masayuki Inaba)

1981 年東京大学工学部機械工学科卒業, 1986 年東京大学大学院工学系研究科情報工学専門課程博士課程修了。工学博士。1986 年東京大学講師。1989 年同助教授, 2000 年同教授。現在大学院情報理工学系研究科創造情報学専攻所属。知能ロボット, ヒューマノイドの研究教育に従事。 (日本ロボット学会正会員)