計測自動制御学会東北支部第302回研究集会(2016.6.24)
資料番号302-5

RGB-Dカメラを用いた移動ロボットの環境認識

An Environmental Recognition System for a Mobile Robot Using an RGB-D Camera

○林賢志*，釜谷博行*，工藤憲昌*
○Satoshi Hayashi*，Hiroyuki Kamaya*，Norimasa Kudoh*
*八戸工業高等専門学校
*National Institute of Technology, Hachinohe College

キーワード: RGB-Dカメラ(RGB-D Camera)，移動ロボット(Mobile Robot)，
環境認識(Environmental Recognition)，PCL(Point Cloud Library)

連絡先: 〒039-1192 八戸市田面木字上野平 16-1 八戸工業高等専門学校　産業システム工学専攻
林賢志，Tel.: 0178-27-7283，E-mail: h28ae05@hachinohe.kosen-ac.jp

1. はじめに

近年、科学技術の発達により様々な分野でロボットの活躍が期待されている。ロボットを自律化することにより、その用途は広範囲なものとなる。未知の環境下において効率のよい自律動作を実現するためにも、ロボット自身で周囲環境の情報を認識する必要がある。この問題に対し、近年では、3次元の情報を取得する技術が注目されている。3次元の情報の取得には、ステレオカメラやRGB-Dカメラが用いられる。安価なRGB-Dカメラには、Intel社のRealSenseやMicrosoft社のKinect等が挙げられる。

本研究では、容易に入手可能であるKinectを用いることで3次元での環境を認識するシステムの開発を行う。実験では、リアルタイムでの動作を実現するための高速化について検討し、3次元での地図生成を行う。また、ロボットの動作計画に必要となる通行可能領域の自動判定処理について検討する。

2. 開発環境

2-1. システム構成

本研究を行うにあたって、使用したシステムの構成をFig. 1に示す。本システムはパソコンとKinect，MobileRobots社のPioneer 3-DXによって構成される。

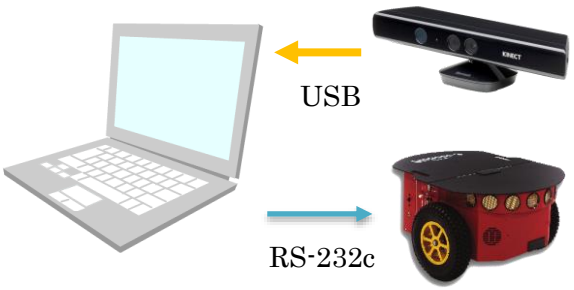

Fig. 1 System configuration

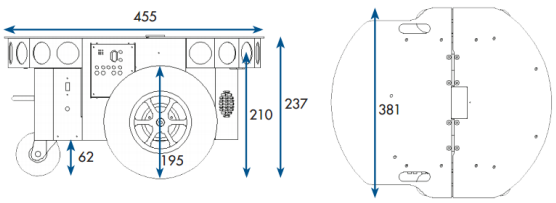

Fig. 2 Pioneer 3-DX mobile robot

研究に用いたPCの仕様はWindows8.1(x64), Intel CORE i7-4790 (3.6GHz, 4コア8スレッド), メモリ8GBである。また、Pioneer 3-DXの全体図、寸法をFig.2に示す[1]。

2-2. RGB-Dカメラ

RGB-Dカメラには、Microsoft社のKinect v1[2]を使用した。本研究では以下の設定で用いる。

- カラー画像解像度：RGB_640x480(30fps)
- 深度画像解像度：640x480 (30fps)
- 深度測定範囲：(Near mode)400～3500mm

2-3. ソフトウェア

ソフトウェアの開発にはMicrosoft Visual Studio 2013(C++)を使用した。3次元点群処理のライブラリとして、Point Cloud Library(PCL)v1.7.2[3]を使用した。Kinectからのデータの取得は、Kinect for Windows SDK v1.8.0[4]を使用した。

3. 前処理

Kinectにより取得した点群データ(Point Cloud Data: PCD) にはノイズ等により外れ値が含まれている。ノイズの原因は測定面の反射率の影響、物体の端面（エッジ）を測定した時の計測のミスによって発生すると考えられる。これは、法線の推定での誤差を生じさせてしまい、結果的に地図生成でのレジストレーション（位置合わせ）の失敗を招いてしまう可能性がある。

一方、Kinectから取得したオリジナルのデータは640×480の画素数との対応付けにより合計307,200個の点群から構成される。点群の数が多いほど多くの計算コストが必要となり、処理時間が膨大になってしまう。

これらの問題に対処するため、以下のフィルタを用いる。

3-1. Pass Through filter

センサから遠く離れたデータは信頼性が低いものとなる。Fig.3の右側の円で囲まれた部分に不安定なデータ領域を示す。Pass Through filterを用いることにより、今回は3m以上離れたデータの除去を行う。この結果をFig.4に示す。

Fig.3 Original data

Fig.4 After Pass Through filter

3-2. Voxel Grid filter

点群の特徴を保存したまま点群データをダウンサンプリングにより軽量化する。これには、Voxel Grid filterを使用する。このフィルタは、空間を一辺の長さa[m]の立方体に分割し、それぞれの立方体の中で代表点(重心)を求めることで点群を削減する。Fig.5にフィルタ適用前のデータ、Fig.6に適用後のデータを示す。

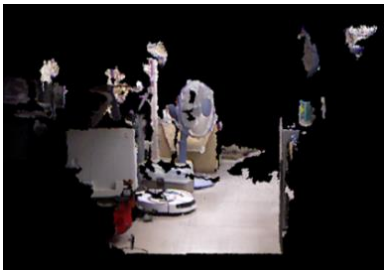

Fig.5 Before Voxel Grid filter

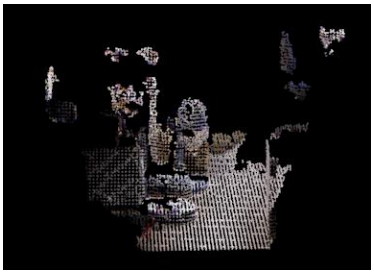

Fig.6 After Voxel Grid filter

3-3.Statistical Outlier Removal filter

ノイズ等の影響によって孤立した点の集合が発生する。また、Pass Through filterにより近くのデータのみのデータを取り出した時にも同様に発生することがある。後の処理の妨げとなる孤立した点を取り除くためにStatistical Outlier Removal filter用いる。

4. 3次元地図生成

ロボットが過去に移動した経路情報を用いて効率の良い動作を行うために地図が必要となる。3次元地図の生成は、ロボットを動作させながら連続したPCDを取得し、マッチングを繰り返していくことで行う。マッチング手順を以下に示す。

4-1. PCDから法線の推定

FPFH(Fast Point Feature Histogram)[5]による特徴量の検出には、各々の点群の法線データが必要となる。法線の推定には、OpenMPを用いて高速化したNormalEstimationOMPモジュールを使用する。結果をFig.7に示す。

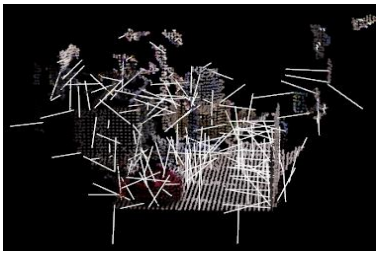

Fig.7 The normals that was estimated

4-2.FPFH(Fast Point Feature Histogram)による特徴量の検出

4-1で推定した法線データを用いて、注目点と近傍点群との組により局所特徴量を計算し、特徴Histogramを作成する。これには、OpenMPを用いて高速化したFPFHEstimationOMPモジュールを使用する。

4-3.SAC-IA(Sample Consensus Initial Alignment)によるマッチング

4-2で生成した2つのPCDの特徴Histogramを用い、PCD間のマッチングを行う。SAC-IAによる位置合わせはRANSAC(Random Sample Consensus)に基づいた手法で、反復計算により2つのPCD間の回転ベクトルと並進ベクトルを求める。これらのベクトルを用いて2つのPCDの位置を近づける。

4-4.ICP(Interactive Closest Point)によるマッチング

4.3で近づけた2つのPCDのマッチング精度をさらに向上させるためにICPマッチングを用いる。なお、ICPマッチングは2つのPCD間の点群の位置がほとんど合っている状態でなければうまくマッチングできない。このため、SAC-IAの適用後に行う。

5. 実験

5-1. データ軽量化による処理の高速化

まず、最初にPCDを取得し、つぎに0.25m直進後に再度PCDを取得する。これら2つのPCDを用いて、マッチング処理の評価を行う。

フィルタ処理前のオリジナルの点群データ数は307,200である。前処理においてVoxel Grid filterを適用しない場合、点群データ数は約250,000となり、その後マッチングを行ったところ、全体の処理時間は約70分であった。このため、点群データ数を大幅に減らす必要が

ある。そこで、Voxel Grid filterを適用し、そのパラメータa(立方体の一辺の長さ)の値を変えて実験を行った。実験結果をTable 1に示す。FitnessScoreとはマッチング後の2つのPCD間のユークリッド距離の誤差に関係した評価値で、この値が小さいほど精度が良いことを表す。また、移動量の推定には、マッチング時に得られた並進ベクトルを用いる。実験結果より、フィルタのパラメータaの値が大きくなるにつれて、点群データ数が減り、処理時間も急激に小さくなるが、FitnessScoreと移動量の推定精度は悪くなることがわかった。つまり、処理時間と推定精度にはトレードオフの関係がある。これ以降の実験では、移動量推定の誤差が小さく、処理時間もある程度小さいa=0.025を用いる。

Table 1 Value and result of the filter

フィルタの一辺の長さa[m]	データ数	処理時間[s]	Fitness Score	移動量[m]
0.010	約50,000	81.72	0.0696	0.240
0.025	約15,000	6.37	0.0876	0.238
0.050	約5,000	0.78	0.0940	0.211

5-2. 3次元地図生成

実際に研究室内で0.25mずつ6ステップ前進移動させ、その後、旋回移動を6ステップさせ、合計12ステップの移動を行った。このとき、それぞれの地点で得られたPCDを用いてマッチングを行い、3次元地図の生成を行った。Fig.8に生成したマップデータにおいて、上部に視点を置いて表示させたものを示す。この図において、ロボットは下方から中央付近まで直進し、その後、右旋回動作を行った。

Fig.9に推定された移動量から導出した経路を示す。ここでは、ロボットの移動開始時の向きをRx方向、横方向をRy方向とする。

実験結果から、直進移動ではほとんど正確にマッチングが行われ安定した地図が生成されている。しかし、旋回時には結果にばらつきが見られ良好な結果を得ることができなかった。

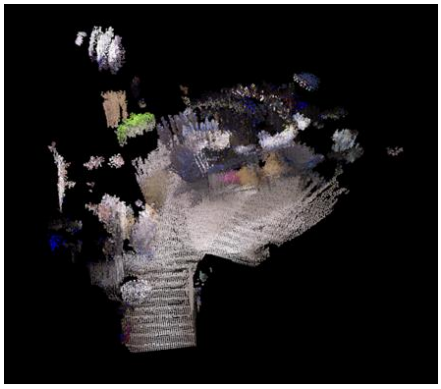

Fig.8 The 3D map which was generated

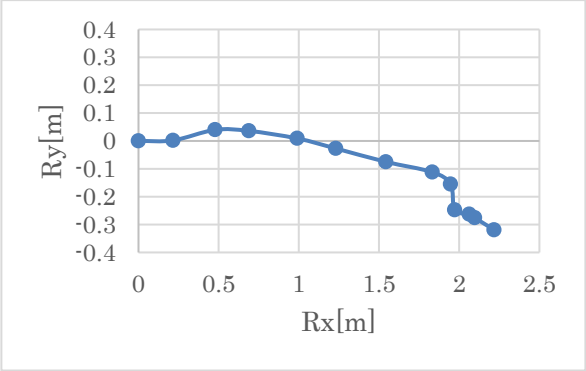

Fig.9 Quantity of estimated movement

5-3. 通行可能領域の自動判定処理

ロボットの動作計画において、ロボットが通行可能であるかの判定が重要になる。このため、障害物のない空間の横幅の最小値を正確に計測することが必要である。横幅の計測は、以下の手順で行う。

まず、取得したPCDを横方向にセクタ分割する（Fig.10）。今回は20分割とした。つぎに、分割されたセクタi毎に、センサ中心から左右方向にそれぞれ最近傍点までの距離r_{li}, r_{ri}を求める（Fig.11）。

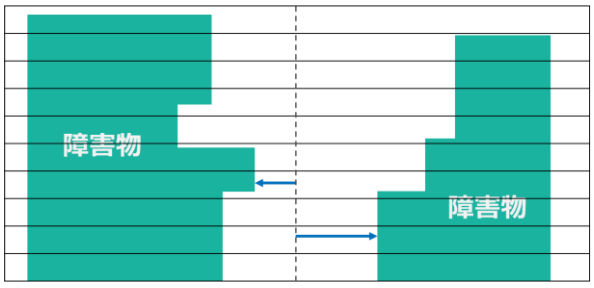

Fig.10 Measurement method

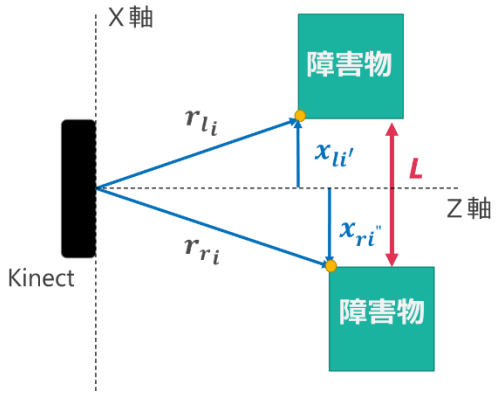

Fig. 11 Measurement method

$$r_{l_i}=\sqrt{x_{li}{}^2+z_{li}{}^2}$$

$$r_{r_i}=\sqrt{x_{ri}{}^2+z_{ri}{}^2}$$

ただし、添字のlはleft、rはrightを示す。その後、左右それぞれにおいて、すべてのセクタの中で距離が最小となるセクタ番号を求める。

$$i'=argmin(r_{li})$$

$$i''=argmin(r_{ri})$$

最後に、障害物のない空間の最小距離Lを次式で求める。

$$L=|x_{li'}|+|x_{ri''}|$$

通行可能領域の判定に用いる横幅は、ロボットの横幅から少し余裕をもった値とし、今回は45cmとした。

障害物として単純な形状のもの、複雑な形状のものをロボット前方において、通行可能領域の判定実験を行う。

まず、前方に2つの段ボール箱を置いた場合について実験を行う（Fig. 12）。ここでは、段ボール間の最短距離を10cm～60cmまで10cm刻みで変えた場合について測定を行う。実験結果

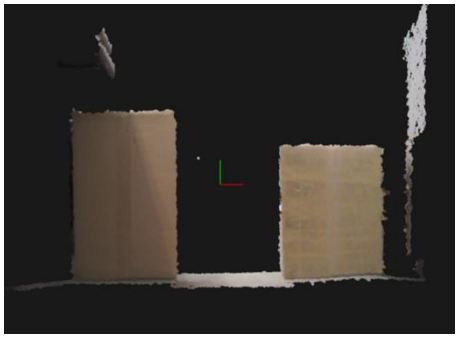

Fig. 12 Data to measure

Table 2 Measurement result

実測値 [cm]	計測距離 [cm]	移動の可否
10	11.6	✕
20	19.6	✕
30	30.0	✕
40	40.4	✕
50	49.7	○
60	62.0	○

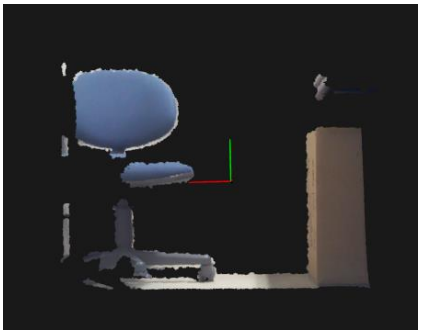

Fig. 13 Data to measure

Table 3 Measurement result

実測値 [cm]	計測距離 [cm]	移動の可否
10	11.6	✕
20	22.1	✕
30	30.7	✕
40	43.5	✕
50	48.1	○
60	60.5	○

をTable 2に示す。結果として、段ボール間の距離はほぼ正確に計測できていることがわかった。

つぎに、前方に椅子と段ボール箱を置いて実験を行う（Fig. 13）。左側の椅子の脚部分から右側の段ボールまでが最短距離となる。この値を変えて測定を行った。実験結果をTable 3に示す。2つの段ボール箱を用いた場合に比べて計測値のばらつきが多少大きくなったが、物体間の最小距離をほぼ正確に取得できているこ

とがわかった。

6. まとめと今後の展望

今回、Kinectを用い、データの軽量化により全体の処理を高速化させることに成功した。また、複数の取得データから3次元の地図生成を行った。さらに、通行可能領域の自動判定に成功した。しかし、地図生成において直進時には良好な結果が得られたが、旋回時に問題が発生することが確認された。

今後の課題としては、地図生成アルゴリズムの改良によるさらなる高速化、旋回時の地図生成における精度の向上、取得した地図を用いた先読みによるロボットの移動経路の自動生成などが挙げられる。

参考文献

[1]Pioneer 3-DX,
http://www.mobilerobots.com/Libraries/Downloads/Pioneer3DX-P3DX-RevA.sflb.ashx
(2016年6月)
[2] Kinect Sensor v1 のスペックまとめ,
http://neareal.com/687(2016年1月)
[3] Point Cloud Library,
http://pointclouds.org (2016年1月)
[4] Kinect for Windows SDK,
http://www.atmarkit.co.jp/ait/articles/1107/28/news137.html (2016年1月)
[5] Radu Bogdan Rusu, Nico Blodow, Michael Beetz, “Fast Point Feature Histograms (FPFH) for 3D Registration”, Proceedings of the 2009 IEEE international conference on Robotics and Automation (2009)