コンピュータ理工学特別研究報告書

題目

移動ロボットのための
2D SLAM を使った地図作成に関する基礎的な研究

学生証番号　545213

氏名　平山　勇

提出日　令和 2 年 2 月 10 日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

移動ロボットは，介護や土木，運送などの現場において人間の肉体労働を代替する機械として期待されている．実用化が進められている自動走行車も移動ロボットの一つである．未知な環境を移動するロボットは，移動可能な経路や空間における自己位置を推定しながら作業を遂行する．本研究では，２次元 LIDAR を搭載した模型自動車を移動ロボットに見立て，概ね正方形のコースで自動運転させて得た周囲環境データから，２次元 SLAM を用いることで，コースの形状と模型自動車の移動軌跡を高精度に推定することに成功した．

本研究では，模型自動車を走らせながら LIDAR でコースを何度も測定する．模型自動車にはオドメトリセンサを搭載していないため，模型自動車の移動軌跡を直接，推定することができない．そこで，各スキャンデータからコースの４頂点をランドマークとして求め，連続するスキャンのランドマークを対応させることで，局所的な移動量（オドメトリ）を推定した．スキャンデータからコースの４頂点を求める処理にはハフ変換を用いた．推定したオドメトリを元のスキャンデータに加え，それらを用いて SLAM を実行した．SLAM には，オープンソースの Little-SLAM を用いた．その結果，コース全体の点群地図と移動経路を高精度化に求めることができた．

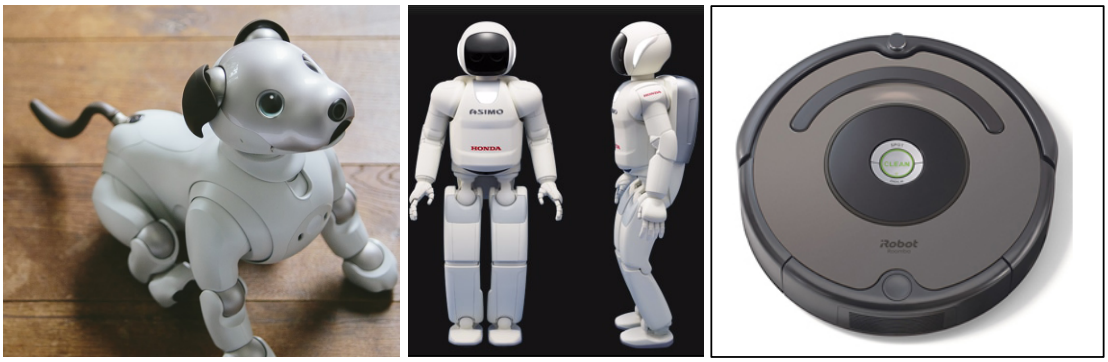
課題として複雑な形状のコースに対応することや，点群地図を用いてコース内を指定経路に従って自動運転する技術の開発が残った．

目次

1章 序論

本論文では，昨年度の特別研究 II[3]で開発した，レーザスキャナを用いた模型自動車を移動ロボットに見立て，移動ロボットが走行する空間の地図と，移動ロボットの走行経路を自動的に生成する SLAM 技術について研究した．

移動ロボットは空間を移動しながら，さまざまな作業を行う機械である．代表的なものに，ソニーの aibo やホンダの ASIMO，自動走行車，ルンバなどの掃除ロボット，Amazon の倉庫で商品棚を移動させるために使われるロボットなどがある（図 1.1）．さまざまなタイプの移動ロボットが存在するが，実用的なものは，車輪で移動するものが多い．



(a) ソニー　aibo
（ソニーのWebサイトの写真を利用）

(b) ホンダ　ASIMO
（本田のWebサイトの写真を利用）

(c) iRobot ルンバ
（iRobotのWebサイトの写真を利用）

図 1.1 代表的な移動ロボット

移動ロボットには，移動経路や作業空間，作業内容が事前に決められているものと，これらを適応的に判断するインテリジェントなものがある．後者の移動ロボットの場合，移動可能な空間をロボット自身が計測し，移動経路を自分で判断する必要がある．例えば，室内を掃除するロボットであれば，掃除できる範囲は壁面の内側で家具が置かれていない場所，ということになるが，壁面の位置や家具の配置は部屋によって異なるので，掃除ロボットがこれらの位置を測定し，幾何学的な情報を得る必要がある．また，移動経路を判断するためには，壁面と家具の位置に対して相対的な掃除ロボット自身の位置・姿勢を把握する必要がある．

移動ロボットに関する技術において，空間の形状と空間内の自己位置を推定する技術を SLAM（Simultaneous Localization and Mapping）とよぶ．SLAM は，周囲の空間情報を計測するセンサとロボットの移動量を測るセンサのデータを統合処理することで，ロボットが移動した空間全体の形状と，ロボットの移動経路を推定する．通常，空間情報を計測するセンサにはカメラや距離センサを使う．また，車輪の回転量や切れ角を測ったり，加速度・角速度センサなどを用いて局所的な移動量を得る．本研究では，空間情報を得るための２次元レーザスキャナーだけを用いた２次元 SLAM について研究した．以下，２章で SLAM について説明する．３章で，SLAM に必要なオドメトリの本研究での求め方を説明し，４章で Little-SLAM を使った実験結果を示す．５章で結論を述べる．

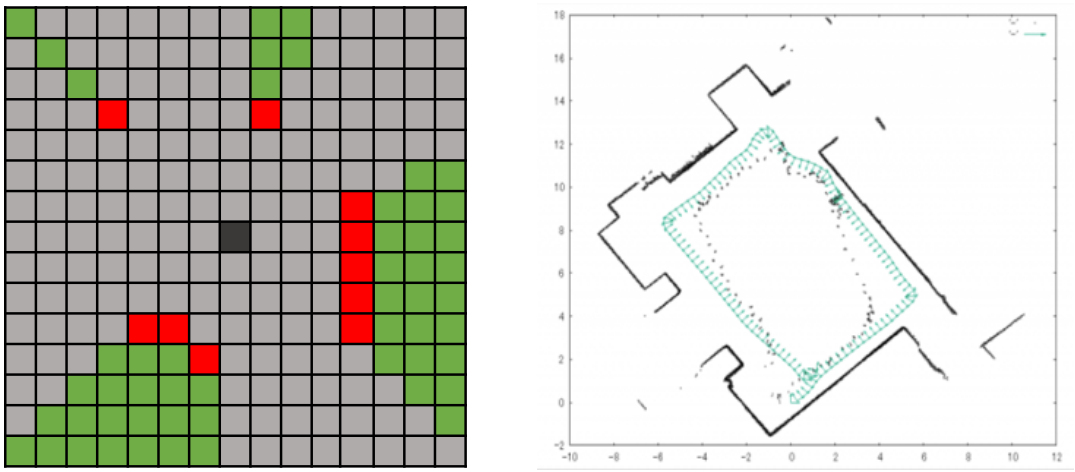
2 章 SLAM

移動ロボットは，移動しながら周囲環境の一部をカメラや距離センサで測定する．これと同時に自身の局所的な移動量を車輪の回転量などから測定する．SLAM はこの２種類のデータを統合することで，周囲環境全体の地図を作成するとともに移動経路を高精度に推定する技術である．２章では本研究を理解するために必要になる SLAM の原理や，SLAM に用いるセンサなどについて説明する．

2.1 移動ロボットの地図

SLAM は Simultaneous Localization And Mapping の頭文字を取った言葉である．直訳すると「自己位置推定(localization)と地図構築(mapping)を同時に行う」ことである．

ここでは自己位置と地図という言葉を説明する[4]．本論文における地図は，移動ロボットが自身のナビゲーションに利用するために周囲の障害物や壁面などの位置情報を表現したもの（環境地図）で，距離尺度を保存したもの（計量地図）という意味である．具体的なものに，グリッドマップや点群地図がある（図 2.1）．グリッドマップは，２次元平面での計量地図の一つで，平面を均等間隔のグリッドに分割し，センサから得た計測点を各グリッドに投票して作成する．点群地図は，計測点の生データである座標値をそのまま保持する形式の地図である．自己位置は地図内での自身の位置と姿勢である．自己位置や地図を平面的に表現したものが２次元地図、立体的に表現したものが３次元地図である．本研究では２次元の自己位置と点群地図を用いる．



(a) グリッドマップの例　　(b) 点群地図の例

図 2.1 移動ロボットが用いる地図の例

2.2 SLAM に用いるセンサ

SLAM は移動ロボットに搭載したセンサのデータを処理する．センサには周囲環境を測定する外界センサと，ロボットの局所的移動量を測定するセンサがある．

周囲環境を測定する代表的な外界センサは次のものである.

① カラーカメラ／白黒カメラ
② 距離センサ
　（ア）LIDAR（Light Detector and Ranging）
　（イ）ToF カメラ（Time of Flight Camera）

カラーカメラ／白黒カメラは通常の可視カメラである．移動するロボットから撮影した複数の画像に SfM（Shape from Motion）とよぶ処理を加えることで，観察した環境の形状を点群として復元することが可能である.距離センサは対象物までの距離を測るセンサである．測定の原理はレーザ光を照射し，その反射時間や位相差を測定するものが主流である.その中で,LIDARはレーザ光源とそれをスキャニングするメカニズム，反射光の遅延時間を測定する回路から構成される距離センサである．ToF カメラは，LIDAR と同様に光の反射時間に基づいて距離を測定し，カメラのように画像データを出力するデバイスである．いずれのデバイスを使っても，出力は観察位置から見た周囲環境の点群データである.

本研究で周囲環境の点群データを測定するために用いたセンサは２次元の LIDAR センサ（Slamtec 社の RPLIDAR A2)である．この LIDAR は,赤外線レーザ光を周囲 360° 方向に時計回りに照射し,レーザ光の反射時間から物体までの距離を求める.図 2.2 に本研究で用いた LIDAR センサの外観を，表 2.1 に性能を示す.

図 2.2 本研究で用いる LIDAR センサ(Slamtec 社の RPLIDAR A2)

表 2.1 RPLIDAR A2 の性能

項目	数値	備考
距離	0.15m～12m	
角度	0°～360°	
距離分解能	0.5mm	
角度分解能	0.45°～1.35°	回転の速さによるが約0.9°
計測時間	0.125msec	回転の速さによる
1秒間のサンプル数	8000個	回転の速さによる
電圧	5V	

次に，ロボットの局所的移動量を測定するセンサに次のものがある．

① 車輪オドメトリ
② 慣性センサ

車輪オドメトリは車輪の回転量と車輪の切れ角から移動量を求める方式である．慣性センサは加速度センサや角速度センサのことである．これらのデータを時間方向に積分することで移動量を推定することができる．

ここでオドメトリという言葉について説明する．英語の odometer は自動車などの走行距離計である．オドメトリは，元々は，車輪の回転量から走行距離を求める方法を意味していたと思われる．移動量を求める方法には，これ以外にも慣性センサを使う方法やカメラ画像を使う方法などがあり，それらは慣性オドメトリ，ビジュアルオドメトリと呼ばれる．これらを組み合わせたビジュアル慣性オドメトリという技術もある．

本研究では局所的な移動量を求めるためのセンサは利用していない．

2.3 SLAM の原理

ここで，一連の点群と一連の局所的移動量をまとめて処理することで，周囲環境全体の地図を作成するとともに，移動経路を推定する SLAM の原理を説明する．

SLAM は，ロボットの移動経路と作成する地図を分類軸とし，それぞれを 2 次元か 3 次元に分けることができる．原理的には 4 通りの分類ができる．それを表 2.2 に示す．3D/3D 型はロボットが 3 次元的に移動し，3 次元地図を作成するタイプの SLAM である．3D/3D 型が最も高機能であるが，処理量が多いため，高性能なコンピュータが必要になる．

2 次元の SLAM を行う際には，レーザスキャナ(LIDAR)を用いることが多い．古くから研究が行われていたため，現在では実用化レベルになっている．3 次元の SLAM を行う際には，3D レーザスキャナ(3D LIDAR)かカメラ，あるいはその両方が用いられる．最近になって性能が向上してきているが，実用化はまだ限定的で現在も精力的に研究が進められている．本研究では 2D/2D 型の SLAM を行う．

表 2.2 SLAM の分類

型	ロボット位置	地図	センサ
2D/2D型	2次元	2次元	2Dレーザスキャナ, オドメトリ, ジャイロ
2D/3D型	2次元	3次元	2D/3Dレーザスキャナ, カメラ, オドメトリ, ジャイロ
3D/2D型	3次元	2次元	カメラ(ステレオカメラ, 距離画像カメラ)
3D/3D型	3次元	3次元	3Dレーザスキャナ, カメラ, IMU

SLAM は地図構築とロボット位置の推定を同時に行う技術である．以下の説明[1]で，地図を作るとはいくつか設定したランドマーク$\boldsymbol{q_i}$の座標を求めることであり，ロボットの位置を推定するとは，いくつかのロボット位置$\boldsymbol{x_i}$を求めることである．ここで，$\boldsymbol{q}_i = (q_{i,x}, q_{i,y})^T$は地図座標系での２次元座標，$\boldsymbol{x}_i = (x_i, y_i, \theta_i)^T$はセンサ座標系での２次元座標にロボットの向き$\boldsymbol{\theta_i}$を加えたものである． Tはベクトルの転置を表す.

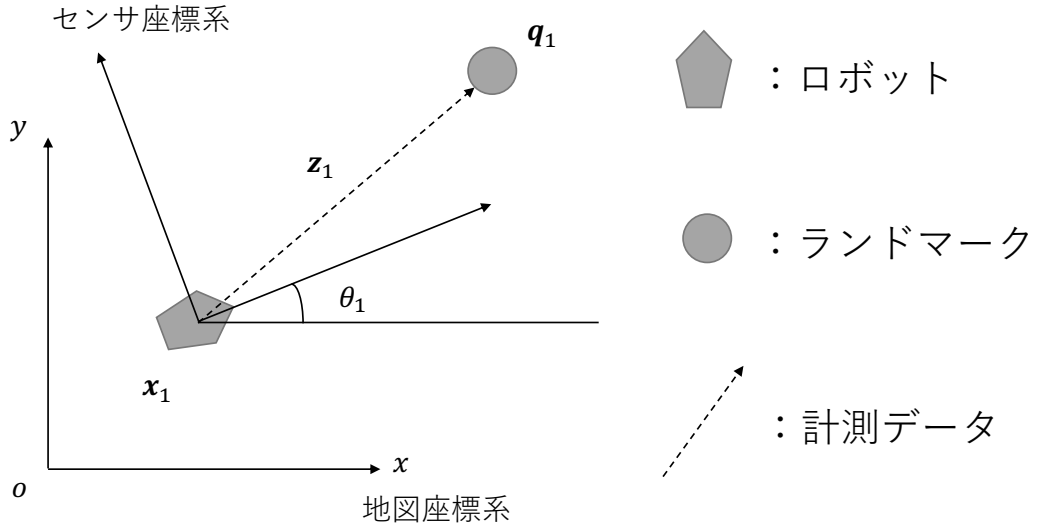


図 2.3　ランドマークの計測値$\boldsymbol{z_1}$と位置$\boldsymbol{q_1}$

図 2.3 において, $\boldsymbol{x_1}$をロボットの位置, $\boldsymbol{q_1}$をランドマークの位置, センサの計測データを$\boldsymbol{z_1}$とする. $\boldsymbol{q_1}$の地図座標系での位置を$\boldsymbol{q_1} = (q_{1,x}, q_{1,y})^T$, センサの計測データのセンサ座標系での値を$\boldsymbol{z_1} = (z_{1,x}, z_{1,y})^T$とする. ロボットの位置$\boldsymbol{x_1} = (x_1, y_1, \theta_1)^T$からランドマークの位置$\boldsymbol{q_1}$を求める式は式(2.1)のようになる.

$$\begin{pmatrix} q_{1,x} \\ q_{1,y} \end{pmatrix} = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{pmatrix} \begin{pmatrix} z_{1,x} \\ z_{1,y} \end{pmatrix} + \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad \cdots (2.1)$$

ここで$\boldsymbol{R_1}$をロボットの方向θ_1による回転行列, $\boldsymbol{t_1}$をロボット位置の並進成分とすると

$$\boldsymbol{R_1} = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{pmatrix}, \quad \boldsymbol{t_1} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

となり, 式(2.1)は式(2.2)のように表すことができる.

$$\boldsymbol{q_1} = \boldsymbol{R_1 z_1} + \boldsymbol{t_1} \quad \cdots (2.2)$$

ロボット位置の推定について説明をする. 前述にてロボットの位置とセンサの測定デ

ータからランドマークの位置を求めた. このことから, 地図を作るためにはロボットの位置の推定が必要である. ロボット位置の推定に関して有力な方法の 1 つとしてオドメトリがある. オドメトリによるロボット位置の計算方法の 1 つを図 2.4 を用いて説明する.

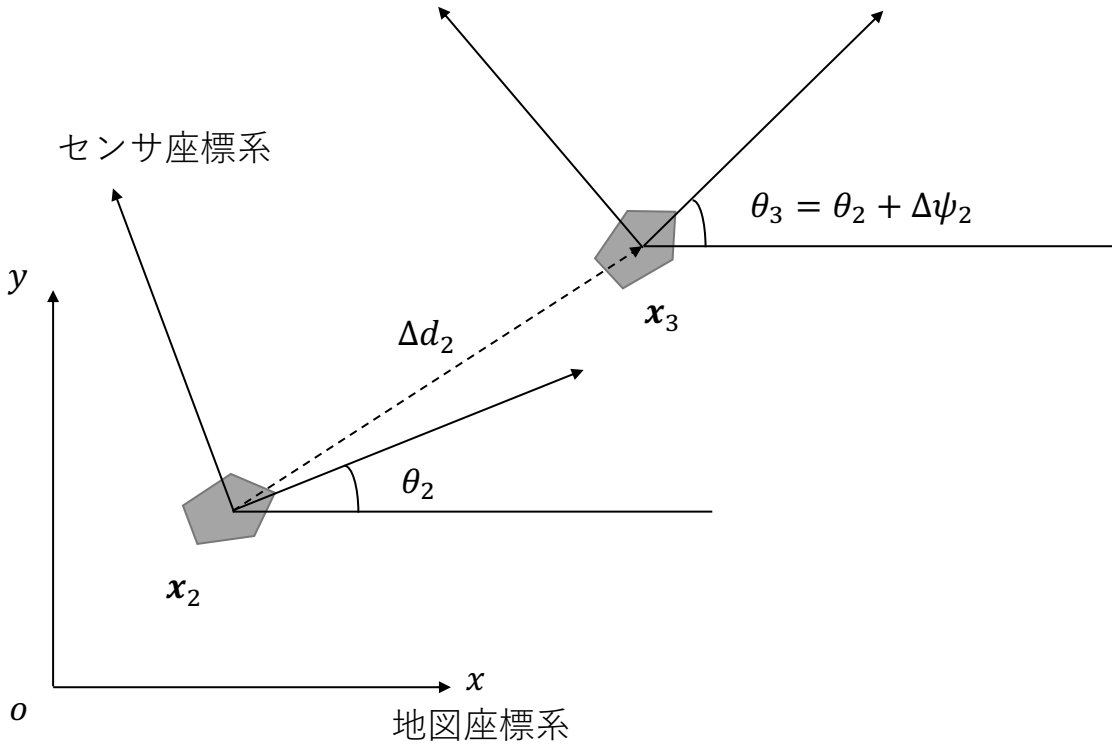


図 2.4 オドメトリによるロボット位置の推定

図 2.4 はロボットが$\boldsymbol{x}_2$から$\boldsymbol{x}_3$に移動する様子を表している. オドメトリで求めたロボットの移動量を$\boldsymbol{a}_2 = (\Delta d_2, 0, \Delta\psi_2)^T$とする. y 成分が 0 なのは, オドメトリで得る移動量は短時間で微小量なもので瞬間的にはロボットは直進しているとみなしているからである. これらを用いるとロボット位置$\boldsymbol{x}_3$は式(2.3)のように計算することができる.

$$\begin{pmatrix} x_3 \\ y_3 \\ \theta_3 \end{pmatrix} = \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta d_2 \\ 0 \\ \Delta\psi_2 \end{pmatrix} + \begin{pmatrix} x_2 \\ y_2 \\ \theta_2 \end{pmatrix} \quad \cdots (2.3)$$

この計算は回転成分を含んだ座標変換であり, compounding 演算子⊕で表すことがある. これを用いると式(2.3)は $\boldsymbol{x}_3 = \boldsymbol{x}_2 \oplus \boldsymbol{a}_2$ と表せる.

オドメトリで位置を求めるためには最初にロボットの初期位置を与え, そこを起点として式(2.3)によって微小な移動量を次々と加算していく. そうすることでロボットの位置を計算することができる. しかしオドメトリは移動量を加算していく積分計算であるため, 移動量が増えるに連れて誤差も加算されていく. この誤差のことを累積誤差という. これを減らすためにはセンサデータをランドマークと照合してロボットの位置を

修正するという手法が有効である. この方法を用いる際に重要なことは同じランドマークを複数回計測することである. その例を図 2.5 を用いて説明する.

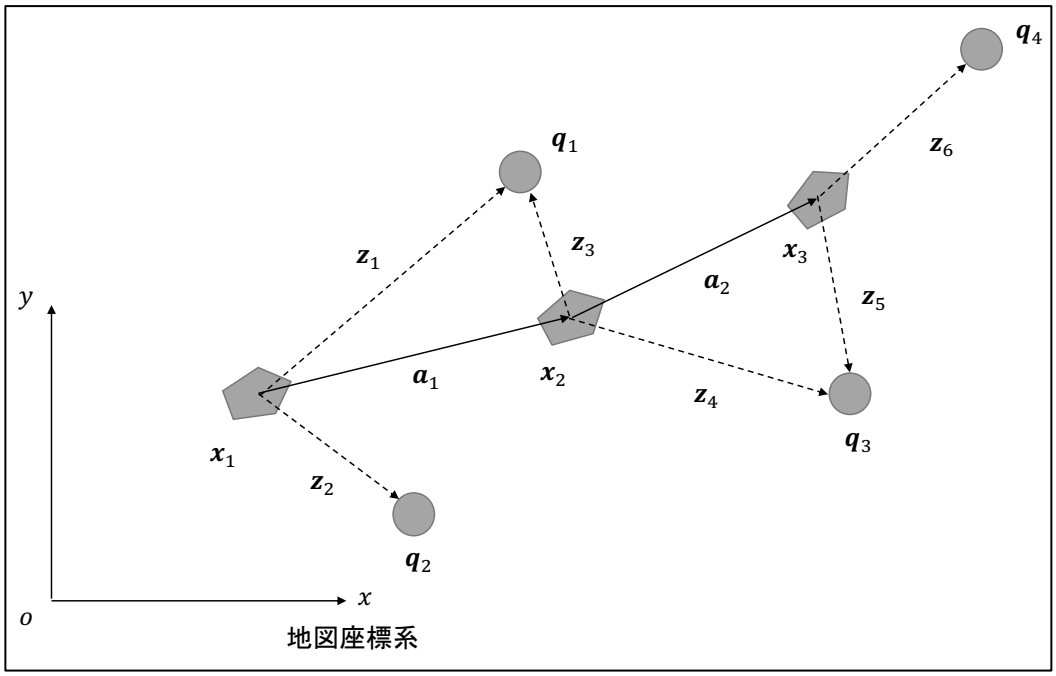


図 2.5　ロボット位置とランドマーク位置の推定

既に計測したランドマークからロボットの位置を逆算して修正していく手法を用いるとして, 式(2.2)によって$\boldsymbol{q_1}$が既知になったとする. $\boldsymbol{q_1}$の値を使って計測データ$\boldsymbol{z_3}$からロボット位置$\boldsymbol{x_2}$に関する式 $\boldsymbol{q_1 = R_2 z_3 + t_2}$ を用いて$\boldsymbol{x_2}$を逆算して求めればロボットの位置を修正することができる. このような式を図 2.5 においてオドメトリとランドマーク計測の全てについて作ると,

① $\boldsymbol{q_1 = R_1 z_1 + t_1}$
② $\boldsymbol{q_1 = R_2 z_3 + t_2}$
③ $\boldsymbol{q_2 = R_1 z_2 + t_1}$
④ $\boldsymbol{q_3 = R_2 z_4 + t_2}$
⑤ $\boldsymbol{q_3 = R_3 z_5 + t_3}$
⑥ $\boldsymbol{q_4 = R_3 z_6 + t_3}$
⑦ $\boldsymbol{x_2 = x_1 \oplus a_1}$
⑧ $\boldsymbol{x_3 = x_2 \oplus a_2}$

という連立方程式になる. これらの式の変数は$\boldsymbol{x_i}$がロボットの位置, $\boldsymbol{q_j}$がランドマークの位置となる. 地図がない初期状態ではロボットの位置$\boldsymbol{x_1}$を決める手がかりがないため, 適当な定数を代入する. 多くの場合は地図座標系での原点になる. このため変数としてのロボットの位置は 2 個になる. そのようにするとロボット位置は 3 次元, ランドマーク位置は 2 次元であることからこれらの式の変数は全部で$3 \times 2 + 2 \times 4 = 14$個, 式は

$2 \times 6 + 3 \times 2 = 18$個となり,変数より式の方が多い連立方程式となる.このようなとき,多くの場合には厳密な解が存在しないため,最小二乗法を用いて解くことが一般的である.センサデータには誤差があり,ロボット位置や地図の推定値にも影響が出てくる.最小二乗法を用いることでそれらの推定値の誤差を減らすことができる.

SLAM の処理にスキャンマッチング, センサ融合,ループ閉じ込みがある.スキャンマッチングとは,2 つのスキャンの形状が合致するように位置合わせを行う技術のことである.ロボットが走行しながら得たスキャンを次々とつなぎ合わせることでロボットの位置と地図を同時に推定することができる.その主な手法として ICP がある.

ICP とは Iterative Closest Points の略で,現在スキャンと参照スキャンの 2 つのスキャンの間でスキャン点の対応づけとロボットの位置推定を交互に繰り返すことでスキャンマッチングを行う方法である.手順として,①データ対応づけ,②ロボット位置の推定, を位置合わせのスコアが変化しなくなるまで繰り返す.データ対応づけの基準としてはユークリッド距離を用いてその距離が最も近いもの同士を対応づける.

センサ融合とは複数のセンサデータからの推定を融合することで,退化への対処として用いられる.退化とは,例えば一度に大量のデータを計測することができるセンサを用いて,その中にいくつかのランドマークが含まれていたとしても,状況によってはロボットの位置を推定できない場合がある.そのような状況を退化という.

ループ閉じ込みとは,周回して同じ場所に戻ってきたことを検出して,その位置を合わせてループを閉じるようにすることである.このループ閉じ込みによって SLAM の累積誤差を減らして歪みの少ない地図にすることが可能である.

2.4 LittleSLAM

本研究では SLAM のプログラムとして,オープンソースの LittleSLAM[2]を使用した.LittleSLAM は,２次元 SLAM のプログラムであり, 次のように動作する.

① ２次元 LIDAR のスキャンデータと局所的な移動量であるオドメトリデータを格納したファイルを入力する.
② 1 回のスキャンデータとオドメトリデータを入力するごとに, SLAM の動作を行い, 地図と経路を更新する.
③ データがなくなれば終了.

このプログラムは Windows と Linux で動作する.

3 章 画像処理によるオドメトリ

3.1 スキャンデータの取得と予備実験

LittleSLAM を実行するために,スキャンデータファイルを作成する必要がある.そのために，2018 年度の特別研究Ⅱで開発したプログラムで，模型自動車をあるコース内で自動運転させた．この時の模型自動車の動作は，単に障害物を避けながら走行を続けるというものである.そして,スキャンデータを LittleSLAM に入力できるテキストファイルとして保存した．ファイルのフォーマットは,

$LASERSCAN\ sid\ t1\ t2\ n\ \phi_0\ d_0 \cdots\cdots \phi_{n-1}\ d_{n-1}\ odom_x\ odom_y\ odom_\theta$

を単位レコードとして,これをスキャンの個数だけ繰り返したものである.ここで,“LASERSCAN”はレコードの先頭を示す文字列．sid はスキャン番号,t1 と t2 は 2 つセットでスキャンのタイムスタンプ,n はスキャン点数,ϕ_iとd_iは i 番目のスキャン点の方向と距離,$odom_x$~$odom_\theta$オドメトリデータである.タイムスタンプは LittleSLAM の実行に必要ないため本実験では 2 つとも 1 を代入した.模型自動車にはオドメトリを測定するセンサは搭載していないので，$odom_x$~$odom_\theta$は 0 とした．上記のフォーマットには記載していないが,今回の実験で用意したテキストファイルにはオドメトリデータの後ろにもタイムスタンプとなる数値の 1 を 2 つ格納している.これは LittleSLAM のテスト用データにオドメトリデータの後ろにタイムスタンプと見られるデータが格納されており,そのテスト用データと今回の実験で用意するデータを同じ形にするために追加した.このタイムスタンプも LittleSLAM を実行するにあたって結果に影響を及ぼすことはない.

予備実験としてこの状態で LittleSLAM を実行させた．その結果，連続するスキャンデータの位置合わせが実行されず，SLAM に失敗した．このことから，LitteSLAM が正しく動作するためには,ある程度正しいオドメトリデータを設定しなければならないと判断した.

3.2 オドメトリデータの計算法

オドメトリデータを得るために，まず,模型自動車に慣性センサや車輪の回転量センサ（ロータリエンコーダ）を設置することを検討した．しかし，これらはハードウエアや機械部品の追加になるのでやや難度が高いこと,慣性センサから移動量を求める場合,センサ値を 2 回積分する必要があり誤差が生じやすいこと，などから，次に述べる画像処理を利用する方法で研究を進めた.

今回は，図 3.1 のコースで実験を行なった．オドメトリデータは連続するスキャン間でのロボットの移動量である．SLAM はこれを高精度化する技術であるが，ある程度の精度を有する初期値が必要となる．それを画像処理的に求める方法として，① コース内に数点のランドマークを設置，② 走行中にランドマーク位置を検出，③ 連続するスキャンでランドマークをマッチングさせることでオドメトリデータを求める,という方

法を考えた.

図 3.1 実験コース

ランドマークは位置の目印となるもののことである.具体的には，ポールのようなものを立てておき，それをカメラの映像から検出するのである．今回は，模型自動車を走らせるコースの壁面の角をランドマークにすることにした.角の検出を容易にするためにコースをほぼ正方形にした.模型自動車を自動運転させ,LIDARのスキャンごとに４頂点の座標を推定し,連続するスキャンにおける４頂点の対応から局所的な移動量を求めることにした.

図 3.2 に１回のスキャンによる点群データの例を示す．コースが正方形であることがわかっているので，点群データに４本の直線をフィティングし，それらの交点として４頂点を求めることにした．直線フィティングには，ハフ変換を用いた．ハフ変換による直線検出は，画像や点の集合から多数決原理で直線を決定するためノイズに強い.また，直線が複数存在していてもそれらを区別して検出することが可能である.

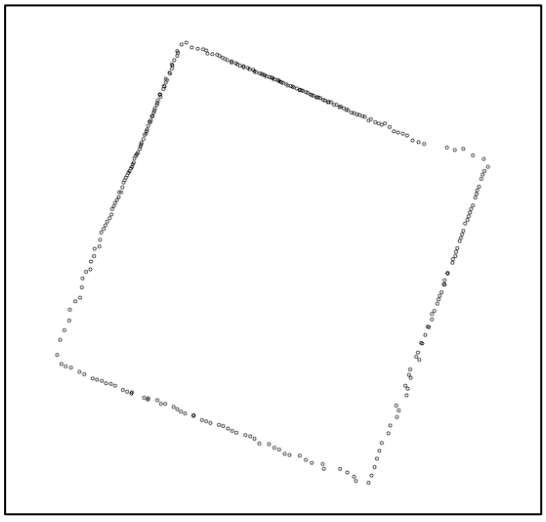


図 3.2 １回のスキャンの点群データ

3.3 ハフ変換による直線検出と頂点座標の算出法

直線を $y=ax+b$ と表現した時（図 3.3），(a,b)を直線のパラメータとよぶ．直線上の一点(x_i,y_i)を直線の式に代入すると $y_i=ax_i+b$ となる．この式はパラメータ空間(a,b)における直線を表す．直線上の多数の点についてこのような変換を行うと，パラメータ空間に多数の直線が現れ，それらは一点で交わる（図 3.4）．つまり，直線上にあると思われる多数の点(x_i,y_i)，{i=1…N}（ただし N は点の数）をパラメータ空間の直線群に変換し，直線群の交点を求めれば元の直線のパラメータを決めることができる．しかしこの式を用いた場合，a と b の取り得る範囲は−∞から+∞となり，パラメータ空間に変換した時に扱いにくい．そこで x, y 座標空間での直線を以下の式で表現する．

$$\rho = x\cos\theta + y\sin\theta \qquad \cdots (3.1)$$

この式を図で表したものを図 3.5 に示す．また直線上のいくつかの点を曲線で表現したものを図 3.6 に示す．

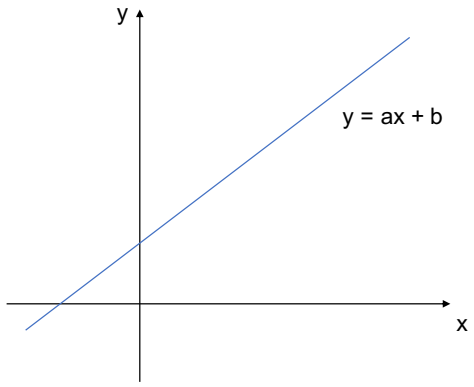


図 3.3 直線

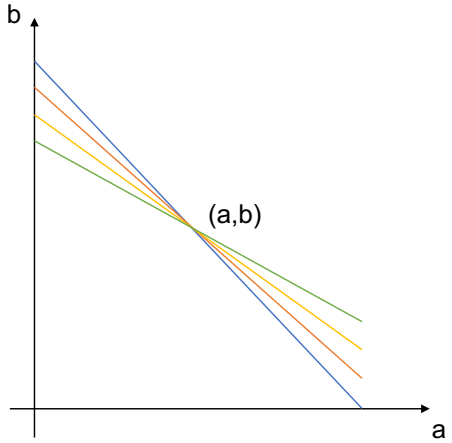


図 3.4 パラメータ空間

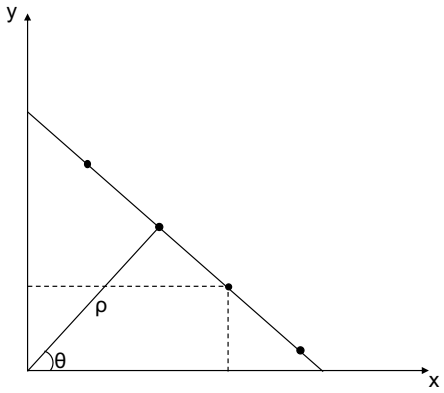


図 3.5 ρ，θ を用いた式を表した図

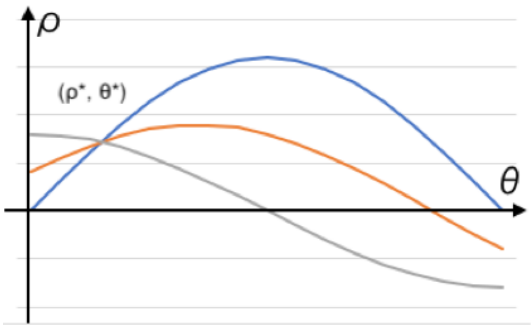


図 3.6 直線上に存在するいくつかの点を ρ，θ パラメータ空間にて描画した図

(3.1)式を三角関数の公式を用いて変形すると以下のようになる．

$$\rho = A\sin(\theta + \alpha)\ ,\ A = \sqrt{{x_i}^2 + {y_i}^2}\ ,\alpha = \cos^{-1}\left(\frac{y_i}{A}\right) \quad \cdots (3.2)$$

ここで A は ρ , θ 平面で描かれる曲線の振幅, α は位相を示す. ある直線上に乗っているいくつかの点は図 3.6 のように異なった振幅と位相を持った正弦波で表すことができる.

今, 直線 l 上に (x_1, y_1)～(x_N, y_N) の点が乗っているとして, それぞれの点を(3.2)式に代入して ρ , θ パラメータ空間のグラフに描画すると, ある 1 点に集中した交点 (ρ^*, θ^*) が求まる. その値を(3.1)の式に代入すると以下の式になる.

$$\rho^* = x\cos\theta^* + y\sin\theta^* \quad \cdots (3.3)$$

この式が, (x_1, y_1)～(x_N, y_N) の座標が乗っている直線である.

このような処理をプログラムで行い, 正方形コースの 4 辺を求めた.

ハフ変換で直線を検出するプログラムを擬似コードで書くと次のようになる.

① ハフ変換した正弦波を記録するための整数配列 img[360][360]を確保し, 0 で初期化する. ここで正弦波を 1 度刻みで描画するために 360 として, 縦方向の解像度はそれに合わせた.

② 点群の座標データ (x_i, y_i), i = 1～N を入力する. ここでは N は点の個数である.

③ i = 1～N のそれぞれの点について以下の処理を行う.

・i 番目の点に対応する正弦波 $A_i\sin(\theta + \alpha_i)$ を求める.

・配列 img において, $A_i\sin(\theta + \alpha_i)$ が通過する画素の値を+1 する.

④ 配列 img の中で最も値の大きい画素位置を求め, それに対応する (ρ , θ) を求める.

⑤ 求める直線は　$\rho = x\cos\theta + y\sin\theta$ である.

求めた正方形コースの 4 辺から, 4 つの頂点の座標を算出する.

3.4 本研究でのオドメトリデータの算出法

3.3 節で算出した正方形コースの 4 頂点座標を用いてオドメトリデータを算出する. ここでのオドメトリデータは, 連続するスキャンにおける模型自動車の局所的な移動量である.

LIDAR が 1 回転する間に約 300 個のデータを取得する. その点群データを可視化すると模型自動車を走らせた正方形のコースが見て取れる. 正方形コースの 4 頂点の座標をスキャンごとに取得し, 連続したスキャンデータを比較して位置合わせを行う.

位置合わせとは, 4 頂点の間での平行移動量と回転量を求めることである. まず, 平行移動量を 4 頂点の重心の移動量として求め, 次いで, 重心を中心とした 4 頂点間の回転角度の平均として回転量を求めた. そしてこれらの平行移動量と回転量をオドメトリデータとした. その様子を図 3.7, 図 3.8 に示す.

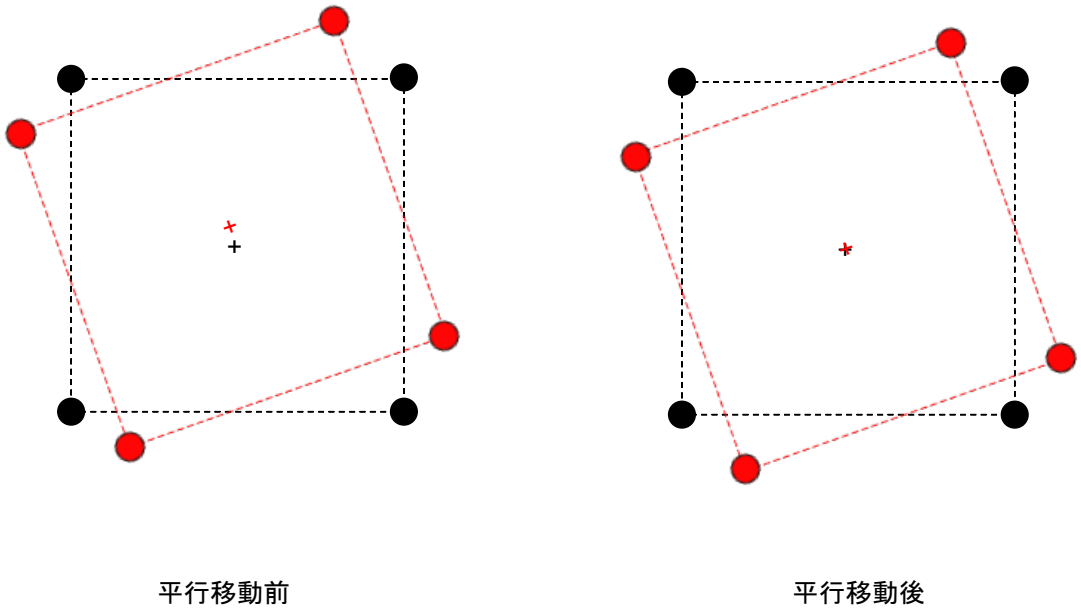


図 3.7 平行移動のイメージ図

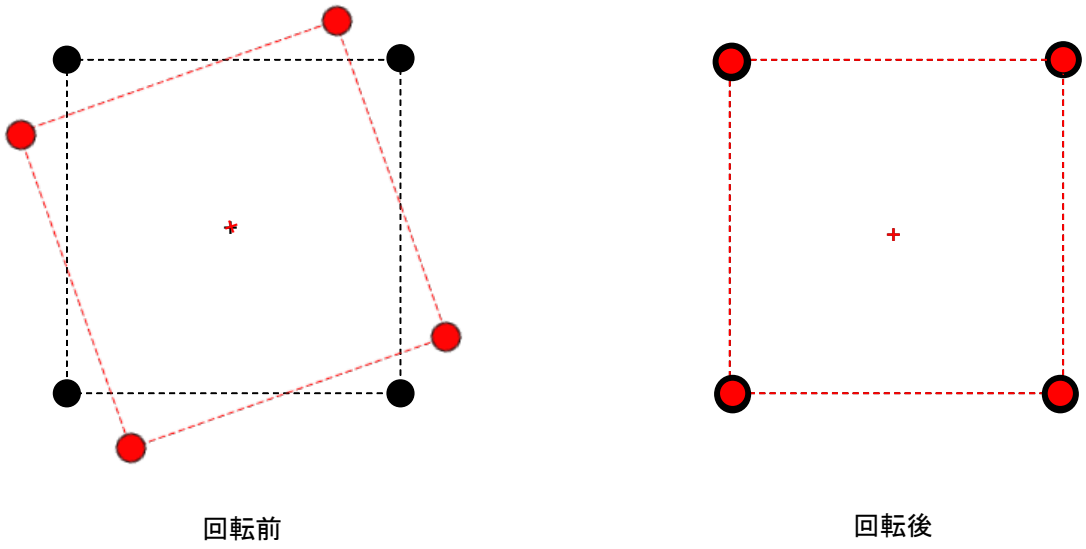


図 3.8 回転のイメージ図

4章 LittleSLAMを使った点群地図と移動軌跡の計算

4.1 LittleSLAMのセットアップ

LittleSLAMのGithubサイト[2]からLittleSLAM-master.zipをダウンロードする.今回はWindowsドライブ下にkenkyuというフォルダを作成して,更にその下にLittleSLAMというフォルダを作成し,そこに解凍したLittleSLAM-masterフォルダの4つのフォルダと3つのファイルをコピーして,同じ場所にbuildフォルダを作成した.

LittleSLAMを実行するために必要なソフトウェアを表2.3に示す.これらも,事前にインストールしておく.

表4.1 LitteSLAMの実行に必要なソフトウェア

ソフトウェア	内容
Boost	C++汎用ライブラリ
Eigen3	線形代数ライブラリ
p2o	ポーズ調整ソルバ
Cmake	ビルド支援ツール
gunplot	グラフ描画ツール

CMakeでLittelSLAMをビルドす.CMakeのGUI画面で,“Where is the source code”の欄にLittleSLAMのフォルダがある相対パスを,“Where to build the binaries”の欄にbuildフォルダがある相対パスを指定した状態でConfigureを押して最後にGenerateを押す.この作業を行うことで,LittleSLAM.slnが生成される.ついで,Visual studioを起動し,Release,x64を指定してBuild Solutionを実行するとbuild/cui/ReleaseフォルダにLittleSLAM.exeが作成さる.

ビルドしたプログラムはコンソールからコマンドを打ち込んで起動する.そのコマンドにはいくつかのオプションを設定することができる.コマンドオプションについての説明を表2.4に示す.

表4.2 コマンドオプション

オプション	内容
無し	オドメトリデータとスキャンからSLAMを行う
-s	個々のスキャンの描画をする(SLAMは行わない)
-o	オドメトリデータに沿ってスキャンを並べて地図を作る(SLAMは行わない)

4.2 スキャンデータからのオドメトリデータの算出

3.1節で述べたように自動運転のプログラムを改良し,正方形コースを自動運転させた.正方形コース内を1周させると約130個のスキャンデータを得ることができた.これは模型自動車がコースを1周する間にLIDARが130回転したことを意味する.LIDARが1回転する間に得るスキャンデータは約300個であった.このようなスキャンデータ

を LittleSLAM で実行できる形式のファイルとして保存した. スキャンデータの一部を図 4.1 に示す.

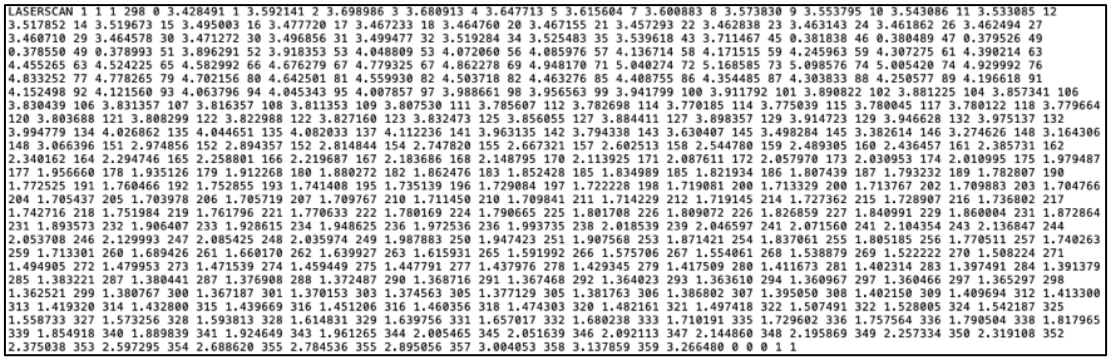


図 4.1 スキャンデータを保存したテキストファイルの一部

作成したスキャンデータファイルを, 4 頂点を算出するプログラムで処理した. 実行結果を可視化したものの一部を図 4.2 に示す. 全てのスキャンデータに対して, このように 4 頂点の座標を算出することができた.

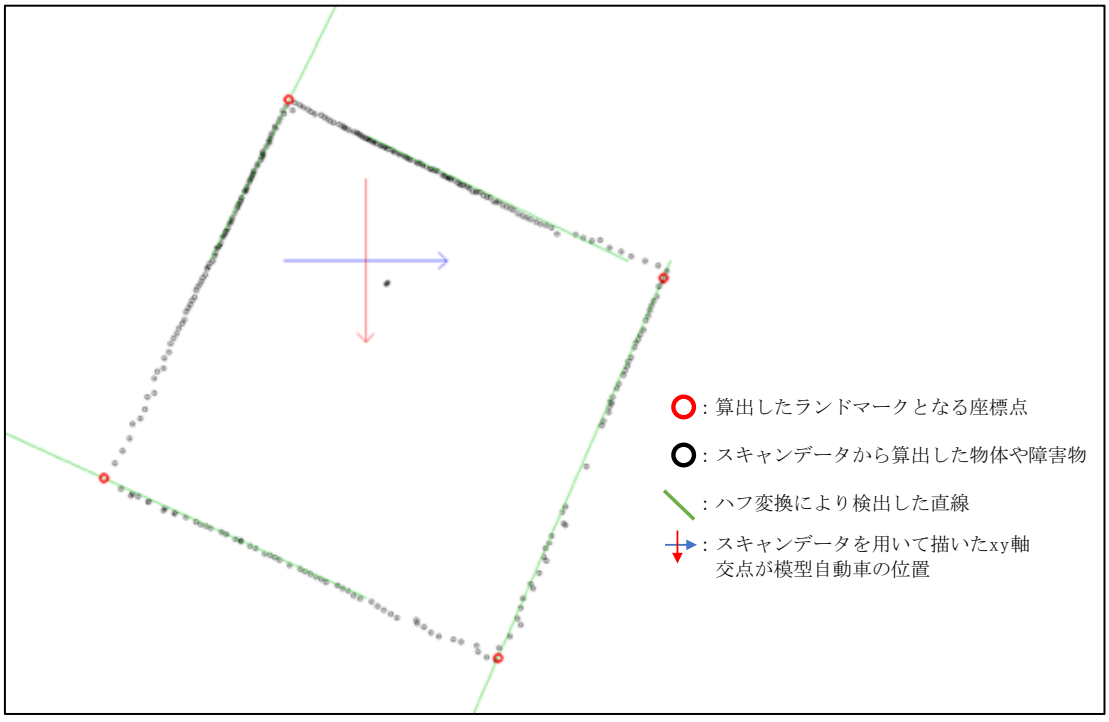


図 4.2 ハフ変換により正方形コースの 4 頂点を求め可視化した図

図 4.2 に関して, 赤と青の軸の交点が模型自動車の位置を表している. 点群地図の上側は多くの点が集まっており, 下側は 1 つ 1 つの点が離れているように見える. これは, 点群地図の上側は, 比較的近い距離で正方形の辺を検出しているため点と点の距離が小さく, 多く密集したような形になり, 下側は離れた位置に存在しているため, 点と点の距離が離れ, 点の密度が疎になる. 求めた 4 頂点の座標列は, 入力したスキャンデータと

は別のテキストファイルとして保存した.

4 頂点の座標列を入力し，連続するスキャンの間のオドメトリデータを算出するプログラムを実行した．図 4.3 に, その中のある連続する 2 つのデータを画像化したものを示す．赤丸が先行するスキャンのデータ, 緑丸がそれに続くスキャンのデータ, 青丸が 2 つ目のデータを 1 つ目のデータに合わせるように平行移動と回転を行い, 位置合わせを行なったものである．多少の誤差は生じているが位置合わせに成功している.

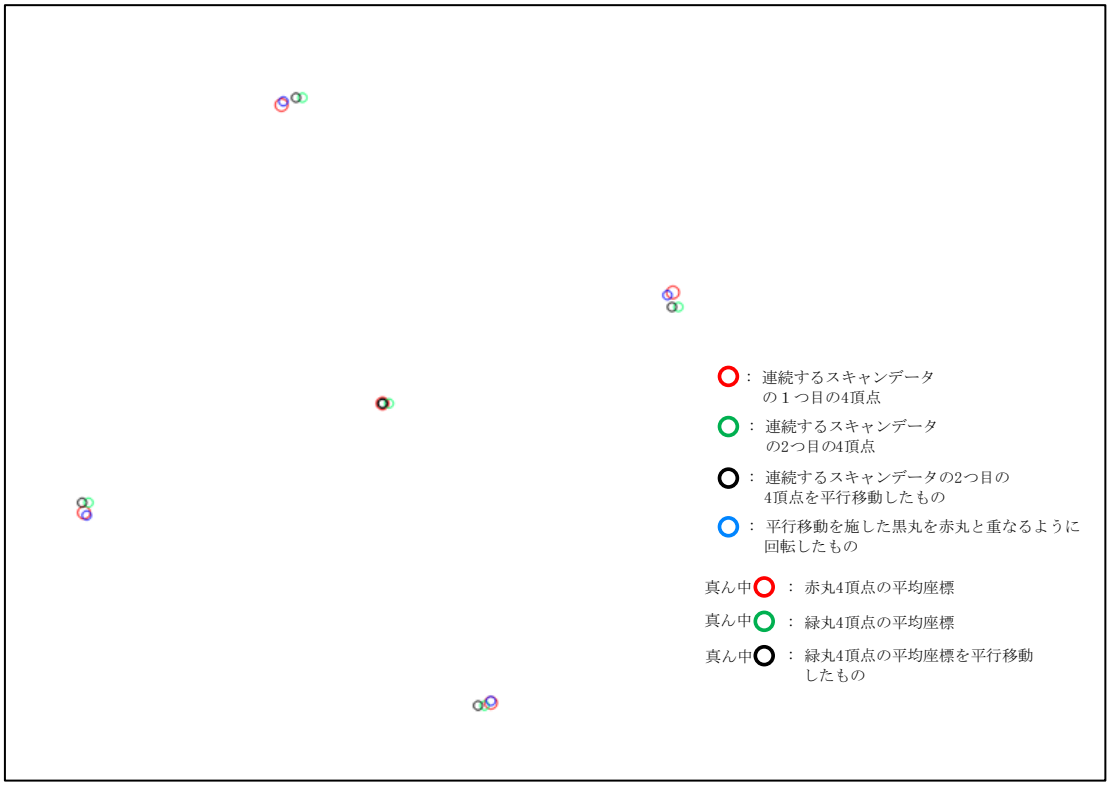


図 4.3 連続するスキャンデータの位置合わせを行ったものを可視化した図

図 4.4 は，図 4.2 の処理を全てのスキャンについて繰り返したもので，４つの青い塊は，連続するスキャンごとに重ね合わせた青丸が，それぞれの場所を埋め尽くしたものである．また，緑丸は模型自動車の軌跡を表している．これらのことから全てのスキャンについて位置合わせが成功していると言える. この模型自動車の軌跡を描くのに用いた値を, オドメトリデータとして，元のスキャンデータの末尾に書き加え，littleSLAM に入力するファイルとした.

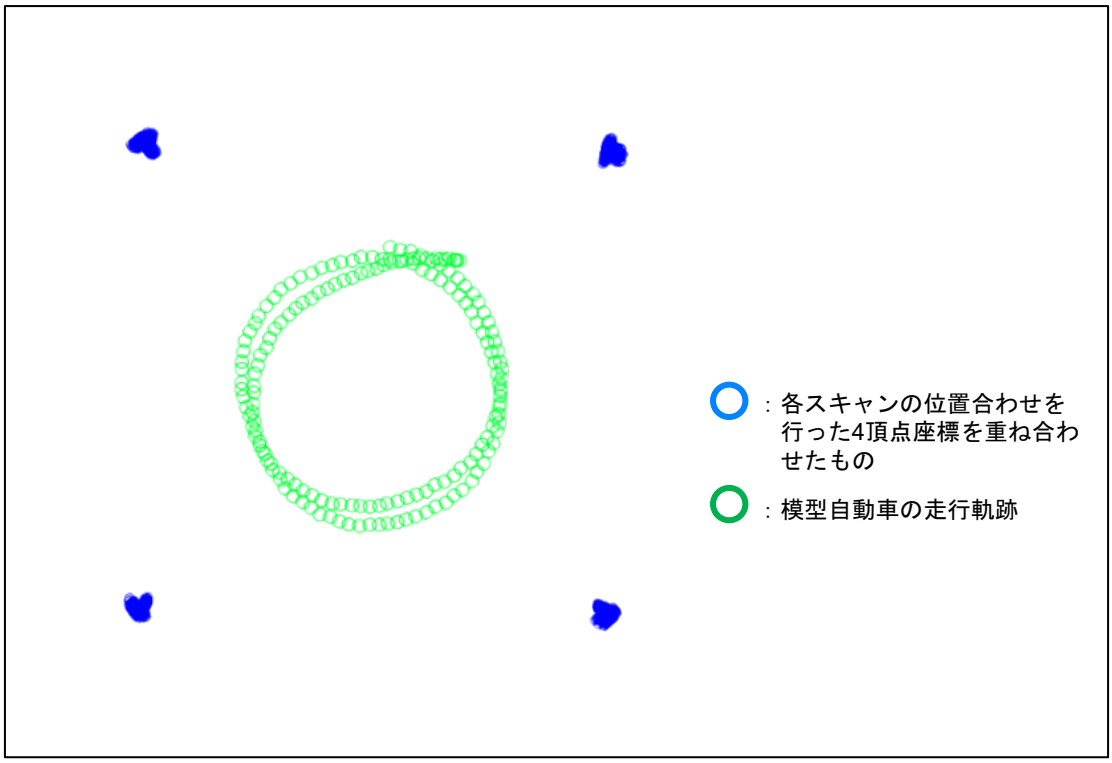


図 4.4 全てのスキャンデータを位置合わせしたものを可視化した図

4.3 点群地図と移動軌跡の生成

LitteSLAM の機能の一つに，SLAM を実行しないで，オドメトリデータだけを使ってスキャンデータを重ね合わせる，というものがある．今回のスキャンデータにこの処理を施した結果を図 4.5 に示す．この図から，オドメトリデータだけでも，概ね良好に位置合わせができていることがわかる．

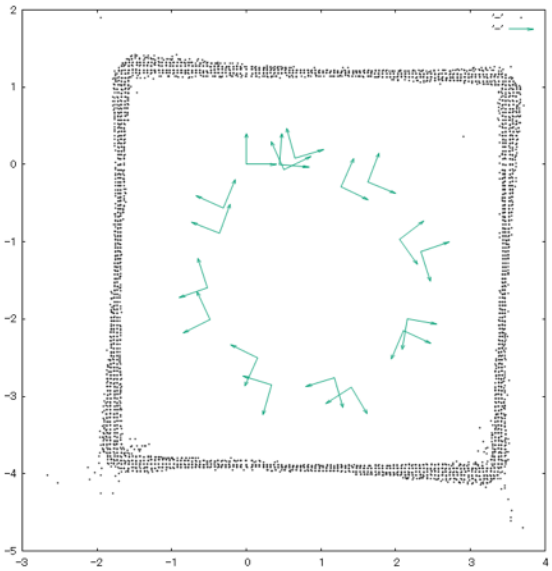


図 4.5 オドメトリデータに沿った点群地図の描画結果

次に,LittleSLAM で SLAM を実行させた.その結果を図 4.6 に示す.図 4.5 と図 4.6 を比較すると次のことに気がつく.

① 図 4.5 の点群の左下と右下に見られる点の散らばりが，図 4.6 には見られない.
② 図 4.5 と図 4.6 の両方とも四隅の近辺で点群が広がる傾向にあるが，図 4.6 の方が広がり方が狭い.
③ 図 4.5 の点群地図はやや傾いている.

①のようになる理由は，SLAM の処理の過程で，スキャンデータに含まれる大きな誤差を持つ計測点が除去されたことである，と考えられる．②のようになる理由は，SLAM で行われるスキャンデータ間のマッチングが,ハフ変換で抽出した 4 頂点のマッチングよりも高精度である，ということである．③については，理由がわからなかった.

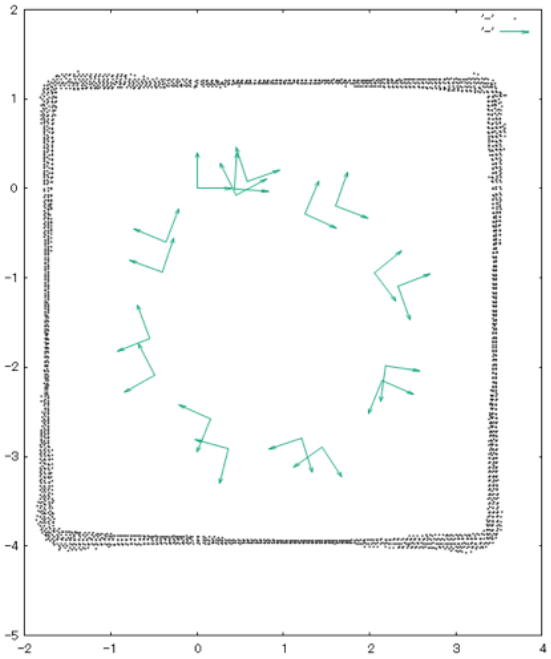


図 4.6 SLAM を実行した際の点群地図の描画結果

最後に SLAM を実行した点群地図に 4.3 節で算出した模型自動車の軌跡を重ね合わせたものを図 4.7 に示す.

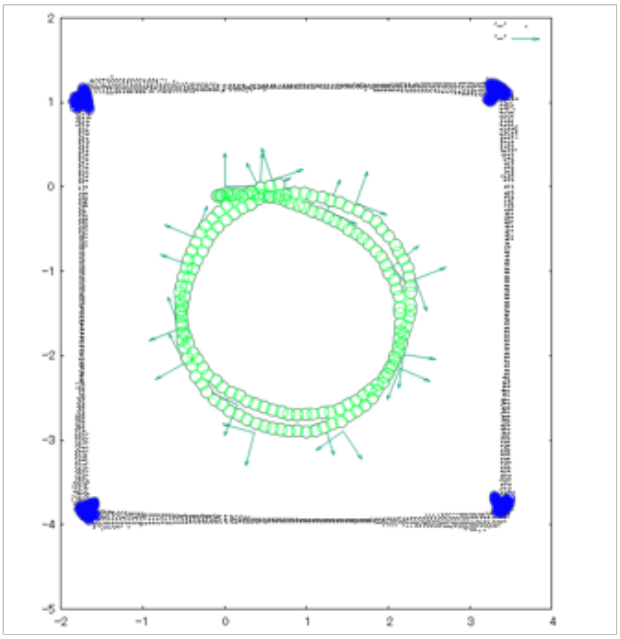


図 4.7 SLAM を行なった点群地図と図 4.4 の模型自動車の経路を重ね合わせた図

5章　結論

本研究では,LIDARセンサを搭載した模型自動車を正方形のコースで自動運転させ,その際に得られるスキャンデータを用いて画像処理によってオドメトリを算出した.そのオドメトリを元のスキャンデータに加えたものを使って SLAM を実行することで，コースの 2D 点群地図を作成することに成功した.

課題として残ったことの一つは，オドメトリの算出を簡単化するために，走行コースを単純化したことである.コースの形状によらずオドメトリを算出できることが望ましい.模型自動車に慣性センサを取り付けて,そのセンサ信号を処理してオドメトリを求めることが一つの解決策である.

当初の研究計画では，作成した地図を利用し，設定した経路で模型自動車を走行させることが最終目標であった．それが実現できなことも課題である.

参考文献

[1] 友納正裕, 「SLAM 入門　ロボットの自己位置推定と地図構築の技術」, 2018 年.
[2] furo-org/LittleSLAM, LittleSLAM, LittleSLAM について.
(https://github.com/furo-org/LittleSLAM)
[3] 白石風太, 「レーザスキャナを用いた模型自動車の制御に関する研究」, 京都産業大学コンピュータ理工学部特別研究報告書, 2018 年度.
[4] 山崎公俊, 倉爪亮, 「移動ロボットのための地図の表現と獲得」, 日本ロボット学会誌, Vol.33 No.10, pp.748-753, 2015 年.

謝辞

本論文を作成にあたり, 丁寧なご指導を賜りました蚊野浩教授に感謝いたします.

付録 1 Raspberry Pi と Macbook 間の WiFi 経由の SSH 通信手順

Macbook 側の設定

・システム環境設定から「ネットワーク」を開き,iPv4 の設定を「DHCP サーバーを利用」にする.

・Macbook に Ethernet ケーブルでネットワークを有線接続にする.

・システム環境設定から「共有」を開く.

・インターネット共有を選択し(この時点ではチェックボックスはオフ),

・共有する接続経路で「Thunderbolt Ethernet」を選択

・相手のコンピュータでのポートで「WiFi」を選択

・同画面下の WiFi オプションを選択

・ネットワーク名とパスワードを設定

・チャンネルは 11,セキュリティは WPA2 パーソナルを選択して OK ボタンを押す

・ネットワーク共有のチェックボックスをオンにして共有を開始する.

・共有画面のインターネット共有がオンになり,緑色の丸があることと,ツールバーの WiFi の扇型で上向きに矢印が出ていることが確認できる.

・Raspberry Pi に挿さっている microSD を Macbook で読み取り,以下のコマンドで wpa_supplicant.conf を編集する.

$vi/volumes/boot/wpa_supplicant.conf

以下が wpa-supplicant.conf の内容

```
---------------------------------------------------------------------------
ctrl_interface=DIR=/var/run/wpa_supplicant.confGROUP=netdev
updata_config=1

network={
        ssid=“設定したネットワーク”
        psk= “設定したネットワークのパスワード”
        key_mgmt=WPA-PSK
}
---------------------------------------------------------------------------
```

・以下のコマンドで空の ssh ファイルの作成

$touch/volumes/boot/ssh

Raspberry Pi 側の設定

・microSD を挿し込み,電源を入れる.

・ターミナルで以下のコマンドを実行し,wpa_supplicant.conf を編集する

$sudo vi/etc/wpa_supplicant/wpa_supplicant.conf

以下が wpa_supplicant.conf の内容

```
ctrl_interface=DIR=/var/run/wpa_supplicant.confGROUP=netdev
updata_config=1

network={
        ssid=“設定したネットワーク”
        psk= “設定したネットワークのパスワード”
        key_mgmt=WPA-PSK
}
```

・Raspberry Pi の画面左上のマークを押して,設定から Raspberry Pi の設定を開く.

・インターフェイスを選択して SSH を有効にする.

・上記を終えたら Raspberry Pi を再起動する.

・画面右上にある WiFi の扇型マークにカーソルを合わせると現在接続されている IP アドレスを見ることが出来る. または以下のコマンドの実行結果の wlan0 の項目にも書かれている.

$iwconfig wlan0

・SSH 接続するため,Macbook 側のターミナルで以下のコマンドを実行する.

$ssh pi@raspberrypi.local

または

$ssh pi@(Raspberry Pi が接続している WiFi の IP アドレス)

・パスワードを入力(初期設定では raspberrypi)

・Macbook のターミナル上で Raspberry Pi の操作ができるようになる.

付録 2 開発したプログラムの説明

[フォルダ名]

Odometry

[プログラム名]

Source.cpp

[内容]

模型自動車を自動運転させて LIDAR センサから得たスキャンデータを読み込み,角度と距離のデータから x,y 座標を算出する.得た座標を用いてハフ変換を行い,正方形コースの 4 つの辺を算出する. 4 つの辺の交点となる座標を求めることでオドメトリを算出するための目印となるランドマークの 4 頂点座標を算出する.

ハフ変換にて算出した正方形コースの 4 頂点座標を格納したテキストファイルを読み込んで,連続するスキャンデータに関して,4 頂点座標の平均座標を算出し,その平均座標を中心に,1つ目のスキャンデータに合わせるように平行移動を行う.また平行移動させた後のデータに関して平均座標と 1 つ目と 2 つ目のそれぞれの 4 頂点座標の内積から角度を算出して回転を行う.この平行移動と回転を施すことで 4 頂点座標の位置合わせを行うプログラム.