—修士学位論文—

屋内用移動ロボットのための ICL-SLAM の精度の向上

Improvement of accuracy of ICL-SLAM for indoor mobile robots

平成 24 年度

三重大学大学院　工学研究科

博士前期課程　電気電子工学専攻

寺田　光博

目　次

第１章

緒　言

1.1　研究背景

これまで製造業の発展に多大な貢献をしてきたロボットの活動環境は，工場などといったロボットにとっては既知であり整備された環境が多かった．しかし昨今，ロボットの活動環境は，より身近な生活の場へと拡大している．その例としては，自動で部屋を巡回してゴミを吸引する『お掃除ロボット』や，主に企業や公共施設などで巡回警備の手助けを行う『警備ロボット』などが挙げられる．このように屋内環境で自律移動をしながら与えられたタスクを遂行する『自律移動ロボット』が実用化されてきている．これは，近い将来ロボットと人間が安心して共存できることを目標としている．

ここで，ロボットが自律移動するために必ず必要となる環境認識手法に着目する．環境認識とはロボット自身が周囲の環境形状，同時に環境の中での自己位置を把握することを意味する．この環境認識を適切に行わせることは，移動を前提とした，あらゆるタスクを実現させるための基礎となり非常に重要である．

現在のお掃除ロボットや警備ロボットの環境認識の方法を挙げれば，接触センサにより物体の配置を確認する接触認識や，建造物の設計図を事前知識として与え，その情報との対応関係から位置を認識する場合が多い．しかし，接触は環境物体あるいはロボットの破損の危険があり，厳密な位置関係の把握は困難である．また事前に環境の地図情報を登録しても実際の環境は備品などが存在し地図とは異なる可能性が高いため対応関係を必ずしも認識できない．ゆえにロボット自身が未知環境を移動する場合，非接触で事前情報を何も与えず，センサ情報から環境認識ができれば最も汎用性が高く，その実現が求められると言える．

1.2　研究目的

未知環境で移動ロボットが自己位置推定と地図生成を同時に行うことは，矛盾する要求でもある．なぜなら自己位置を推定するためには周囲環境の把握が必要であり，地図を作成するためには自己位置を同定していなければならない．この移動ロボットの未知環境における認識問題として移動ロボット工学において頻繁に用いられる言葉に SLAM (Simultaneous Localization And Mapping)がある．SLAM とは文字通り“同時の自己位置推定と地図生成”という前述の問題を表すとともに，問題解決のために各種センサを用いた手法自体の名称としても用いられる．

そもそも SLAM が，移動ロボットの普遍的な問題となる理由としては，あらゆる“不確かさ”が挙げられる．ロボットの移動には，車輪移動ロボットであれば床とのすべりやセンサによる測定の観測誤差，ロボットのモデル化誤差など，あらゆる不確かさが含まれる．その不確かさは短距離移動においては問題にならない場合も多いが，長距離移動では移動に伴い誤差が蓄積し，最終的に地図が作成できず自己位置推定も失敗に終わる．よって SLAM には，この不確かさの影響に対処できるような手法を考える必要がある．

SLAM には様々な種類が存在する[1]．拡張カルマンフィルタ(EKF: Extended Kalman Filter)を用いた EKF-SLAM や，パーティクルフィルタ(Particle Filter)を用いた Fast-SLAM がある．しかし，これらのフィルタを用いる SLAM は推定するシステムの運動を数式によってモデル化する必要があり，その際に近似などの仮定を伴う．すると，モデル化されていない想定外の事象に脆弱となってしまう．特に，タイヤのすべりや障害物への衝突によるロボットの急な姿勢の変化に対処しにくい．

それらとは別に，モデルを用いず，センサから得られた情報のみを用いて SLAM を行う ICP-SLAM が存在する．しかし，この ICP-SLAM は前述したモデル化の問題は無くなるが，センサ情報に忠実なため，センサの測定誤差に影響されやすいという欠点がある．そこで前任者はセンサの測定誤差をなるべく除去しつつ精度を向上させるために，点で SLAM を行う ICP-SLAM から，直線で SLAM を行う ICL-SLAM へと変更を行った[2]．しかし，現在の ICL-SLAM では，点ではなく直線を扱うことに起因する問題が残っている．その問題とは，移動ロボットの位置によっては他の障害物の陰に隠れるなどで同じ障害物を表す直線であっても長さが異なる場合が出てくるというものである．このような場合，現在の ICL-SLAM では正確に SLAM を行うことが出来ない．そこで，本研究はでは ICL-SLAM に付加的な手法を提案し，直線の長さが異なる場合においても正確に SLAM を行うことが出来ることを目的とし，実験によりその有効性を確認する．

1.3　研究の位置づけ

本研究で対象とするロボットは，屋内を移動対象とする．内界・外界センサにはロータリーエンコーダと Laser Range Finder(以下 LRF)を使用する．その際，重要な制約として LRF で取得される計測情報は，距離と角度で指定される数百という点情報となり SLAM において，その計測点すべてを入力として扱うことは計算負荷の面から困難である．そのため人間が環境にランドマークを設置し，環境の認識のし易さを任意に決定する．しかし，ランドマークを設置することは本当の未知環境とは言えず，研究の意図として適当ではない．よって，計測点から環境を表現する特徴を抽出する必要がある．この特徴には，移動ロボットの移動環境としては屋内環境を想定しているため壁や部屋など直線環境が多いことから，直線を採用した．この直線を用いた SLAM は，Line-based SLAM とも呼ばれ，研究が盛んに行われている[3].

また，本研究で用いる ICP-SLAM は，ICP アルゴリズム(Iterative Closest Point Algorithm)という手法を用いた SLAM である．ICP アルゴリズムは，第 3 章で詳しく説明するが，主に画像認識の分野において形状の位置合わせ手法として広く用いられている手法である．ある形状のモデルデータに対して，それと似た形状のデータを比較した際，似た形状のデータをどれだけ回転し移動すればモデルのデータに最も近づけることができるのかを計算する．ICP-SLAM はこれにより地図情報とロボットの移動パラメータを同時に推定する．しかし，この ICP アルゴリズムには，センサの誤計測に影響を受けやすい欠点が存在する．そのため，計測後に誤計測だと判断した部分を提案手法によって除去することで ICP-SLAM の精度を向上させたものが ICL-SLAM である．

しかし，現在の ICL-SLAM は直線の端点情報で ICP アルゴリズムを実行している．これでは，移動ロボットの位置によっては他の障害物の陰に隠れるなどで同じ障害物を表す直線であっても長さが異なる場合に不正確な結果となり，精度が低下してしまうという問題が発生する．通常の ICP アルゴリズムでは直線を点が一列に並んだものとして扱っている．そのため，長さが異なる場合であっても直線の一部の点が存在する部分や他の直線との位置関係を見ることで正しく位置を合わせることが出来る．しかし，直線の端点情報のみを使用する場合では，直線の長さが異なると端点の位置も異なってくる．このため，他の直線との位置関係を見たとしても長さの異なる直線の位置を正確に合わせることが出来ない．

そこで，本研究では点データではなく線データを用いて ICL-SLAM を行うことが出来るように ICP アルゴリズムの改良を提案する．本研究で注目するのは直線の角度を表す方向ベクトルである．方向ベクトルは直線の角度のみを表すもので，直線の位置や長さなどの情報を持たない．そのため，同じ障害物を表す直線で長さが異なる場合でも方向ベクトルを比較することで正確に回転量を求めることが出来る．これにより，ICL-SLAM での地図作成および自己位置推定を行った際の精度の向上を達成する．

第２章

問題設定と定義

本章では SLAM が扱う問題の設定や，使用するロボットの機構について定義する．

2.1　環境設定

- 単独／複数ロボット

ロボットの位置推定問題には，ロボットの数が大きく関係する．複数ロボットの位置推定に関する研究[4]は存在するが，コストの問題やロボット同士の通信による知識の共有など，とても複雑になる．よって，本研究では問題を簡単にするため扱わず，最も研究されている問題である単独ロボットの位置推定を扱う．

- 静的／動的環境

位置推定の難易度に影響を与える要因は，環境が静的か動的かによって分けられる．本稿では，静的環境を扱う．静的環境とは，変化するものがロボットの位置姿勢だけという環境を指す．言いかえれば，ロボット以外の物体は永遠に動かないため推定問題も動的と比べ簡単となる．

2.2　移動ロボットと座標系の定義

● 移動ロボットの機構

移動ロボットは図 2.1 に示す独立操舵二輪ロボット MIUBOT を用いる．移動機構として車輪を用いる理由としては，制御が比較的容易であることが挙げられる．MIUBOT の仕様を表 2.1 に示す．

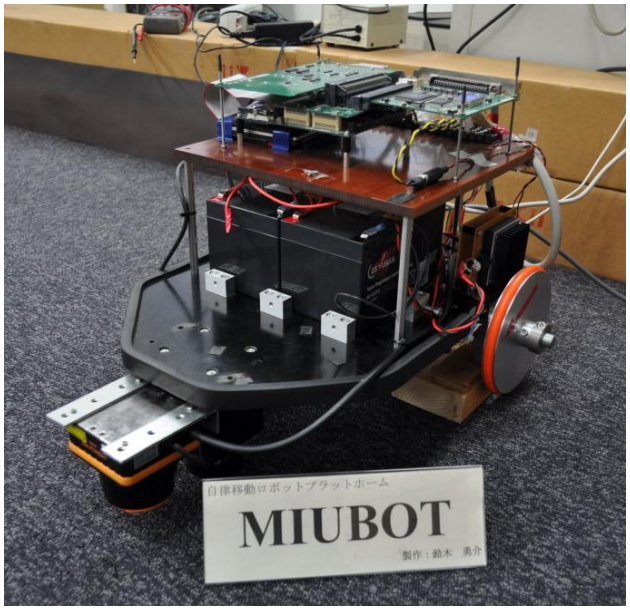


図 2.1　MIUBOT

表 2.1　MIUBOT 仕様

走行方式	独立二輪操舵方式
CPU ボード	LX-2020　（イノテック株式会社）
LRF	UHG-08LX　（北陽電機株式会社）
駆動装置	FHA-8C-50 (ハーモニックドライブシステムズ)
減速比	50
最大動作速度	1m/s 程度
車輪間距離	0.34m
車輪半径	0.07m
重量	17kg

● 使用センサ

通常，ロボットが周囲の環境を読み取るためには，センサからの情報を利用する．センサは大きく分けて，内界センサ(Internal Sensor)と外界センサ(External Sensor)の 2 種類に分けられる．

外界センサはロボットの外界の情報を知るためのセンサであり，タッチスイッチセンサ，距離センサ，カメラなどがある．タッチスイッチセンサは圧力センサを利用したもので，センサが物体に触れたときに圧力が変化することを利用してスイッチを入切するセンサである．距離センサは，障害物までの距離や，追跡物までの距離を計測することに使用する．超音波センサやレーザ測域センサなどがこれに含まれる．カメラは，画像として周囲の情報を取得することに用いることが多いが，画像を 2 枚利用して全画面の距離情報を得る(ステレオビジョン)こともできる．

しかし，タッチセンサは障害物に初めて接触を行ったときにしか情報が得られないため，ロボットや周りの環境を傷つけたり環境を変形させてしまう恐れがある．そのため，環境地図の作成には適さない．また，カメラから得られる情報量はとても多く，距離情報を得るにはその処理に時間がかかってしまう．以上の理由から，環境情報の取得に一般的に使用されているセンサは距離センサが多い．超音波センサは測定に音を使用するが，音は広がりやすいためロボットに対して横方向の細かい測定には向かない．

対して，内界センサは，ロボットの内部の状態を知るためのセンサであり，例えば関節角変位を測るエンコーダ，速度を測るタコジェネレータが相当する．内界センサを用いた自己位置推定の最も基本的な方法として，デッドレコニング(Dead Reckoning)がある．デッドレコニングは，移動ロボットが自己位置を推定しながら目的地まで移動する手法を指す．一般に，デッドレコニングは車輪型ロボットに適用されることが多い．その際，ロボットの初期位置が分かっている状態で，その値にロータリーエンコーダから取得した車輪の回転角度からロボットの移動量を求め，足し合わせていくことで自己位置を算出する．これをオドメトリ(Odometry)という．しかし，ここで注意しなければならないのは，内界センサの情報はあくまでもロボット内部の情報であり，前述したようにセンサの値を足し合わせているだけなので，長距離を走行すると位置推定の誤差が蓄積しまうという問題がある．特に，エンコーダの回転量と走行距離は，地面と車輪のすべりが生じてしまうために一対一に対応するとは言えない．そのため，通常は内界センサのみを信じることは行わず，外界センサと組み合わせることで，内界センサの誤差を補正する場合が多い．

本研究で使用するセンサは，内界センサにロータリーエンコーダ，外界センサに LRF を使用する．ロータリーエンコーダはロボットのサーボモータに初期搭載されているものが多く，手に入れやすい．また，LRF は非接触センサの中でも比較的精度が高いと言われているため，本研究ではこれを用いる．

● ロータリーエンコーダ

ロータリーエンコーダはモータの回転量を電気信号に変換し，タイヤの回転量を得るセンサである．MIUBOT には図 2.2，表 2.2 に示すハーモニックドライブシステムズ(株)の FHA-8C-50 が搭載されている．これは，厳密には AC サーボアクチュエータというもので，モータとエンコーダが一体となっている．これにより，タイヤの回転量からロボットの移動距離を推定する．

図 2.2　FHA-8C-50 （ハーモニックドライブシステムズ）

表 2.2　FHA-8C-50 仕様(出力軸における代表値)

電源	DC24V
最大トルク	3.3N・m
最高回転速度	120r/min
トルク定数	1.3N・m/A
最大電流	3.3A
許容連続電流	1.7A
相抵抗	0.54Ω (20℃)
相インダクタンス	0.22mH
慣性モーメント	0.0074kg・m^2
減速比	50
許容モーメント荷重	15N・m
モーメント剛性	2×10^4N・m/rad
モータ位置検出器	インクリメンタルエンコーダ
検出器分解能(4 逓倍時)	4000,000 パルス/回転
質量	0.40kg

- LRF (Laser Range Finder)

LRF は北陽電機株式会社の測域センサ UHG-08LX を用いる．測定領域は 2 次元および，図 2.4 のように右側から測定分解能ごとに測定可能距離内の距離を走査していき(2.1)式のようにi番目の点p_iに関する距離d_iと角度α_iを返し，処理周期ごとにこれを繰り返す．本研究で用いる LRF と測定領域のイメージ，その仕様を図 2.3，図 2.4 と表 2.3 に示す．なお，LRF は図 2.1 でも確認できるように MIUBOT の先頭部分に設置している．単周期における LRF の取得環境と結果例を図 2.5 と図 2.6 に示す．

$$p_i = [d_i \quad \alpha_i] \tag{2.1}$$

図 2.3　LRF (UHG-08LX)

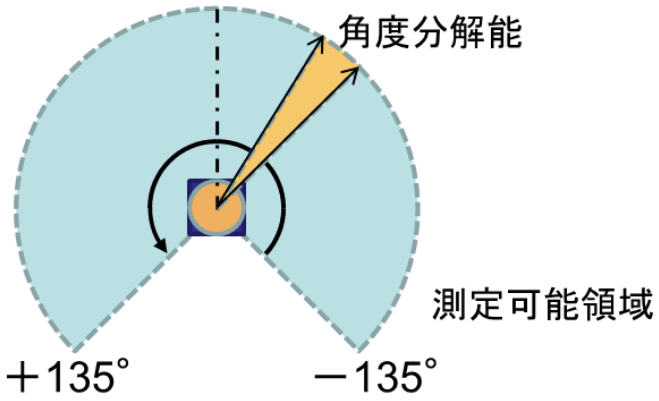


図 2.4　測定領域のイメージ

表 2.3　LRF (UHG-08LX)仕様

電源	DC12V±10%
測距範囲	距離：30～11,000mm 角度：270°
測距精度	100～1,000mm：±30mm 1,000～8,000mm：距離の±3%
角度分解能	約 0.36°(360°/1,024 分割)
光源	半導体レーザ　λ=785nm (FDA 認証　レーザ安全クラス 1)
走査時間	67ms/scan
騒音	25dB 以下
インタフェース	USB2.0(Full Speed)
使用周囲照度※1	ハロゲン・水銀灯：10,000lx 以下 蛍光灯：6,000lx(最大照度)
使用周囲温度・湿度	-10～+50℃、 85% RH 以下 (但し、結露・氷結しないこと)
耐振動	10～55Hz、複振幅 1.5mm、X・Y・Z 方向、各 2 時間
耐衝撃	196m／s²、 X・Y・Z 方向、各 10 回
質量	約 500g(ケーブル含む)

※1.太陽光など強い光を直接受けた場合は誤出力する場合がある.

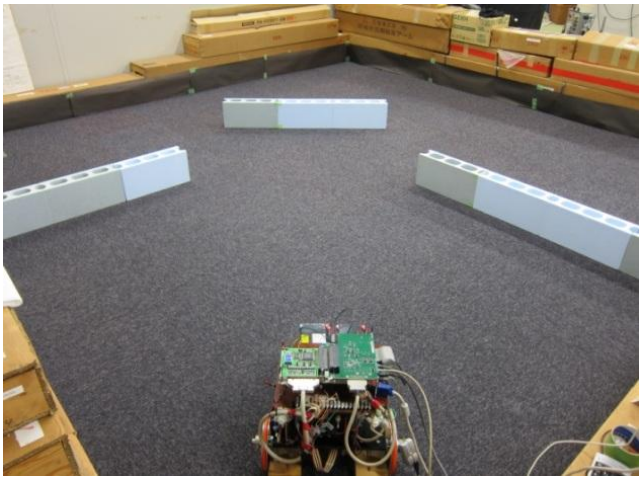


図 2.5　LRF 計測環境

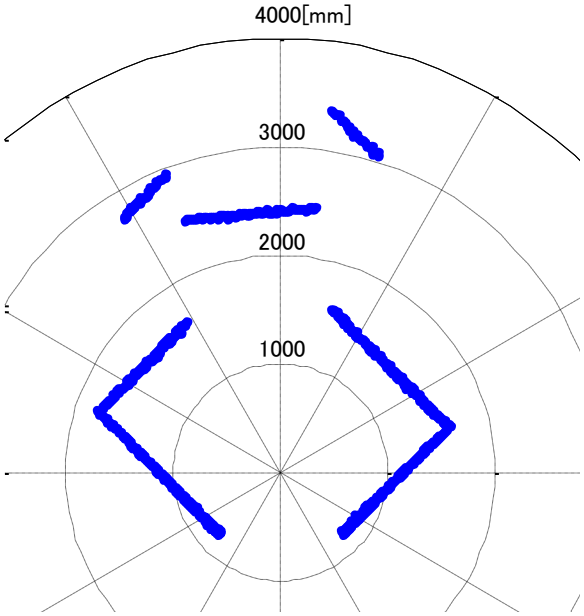


図 2.6　LRF 計測点

● ロボット座標系と変数

ロボットの座標系と変数を図 2.7 に示す. 移動ロボットの位置姿勢はグローバル座標系の原点O_Gから車軸中心であるローカル座標系の原点O_Lまでの相対的な位置関係である. SLAM ではロボット変数行列$\boldsymbol{R}$を(2.2)式と定義する．x, y, θはグローバル座標系におけるロボットの座標位置と角度である．またvとωはそれぞれ(2.3)式と(2.4)式で表現されるロボットの移動方向における直進速度と角速度であり，v_rとv_lは左右車輪の直進速度，bは車軸半径である.

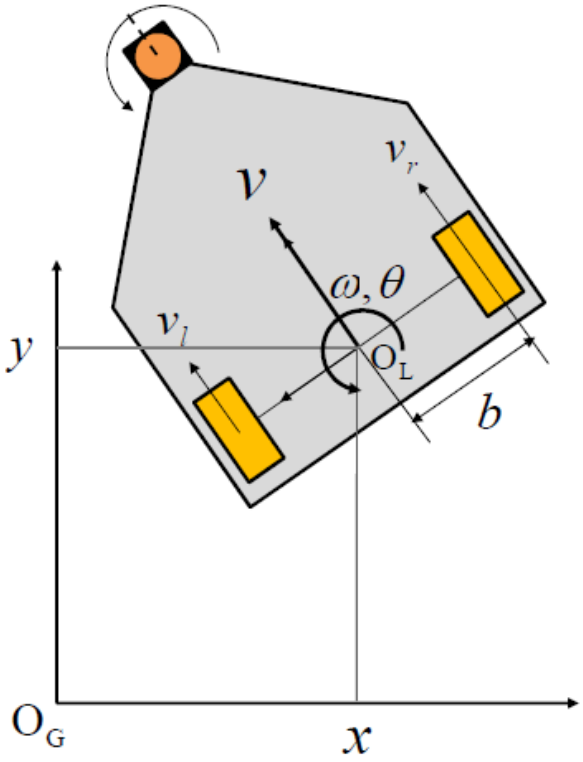


図 2.7　ロボット座標系

$$\boldsymbol{R} = [x \quad y \quad \theta \quad v \quad \omega]^T \tag{2.2}$$

$$v = \frac{v_r + v_l}{2} \tag{2.3}$$

$$\omega = \frac{v_r + v_l}{2b} \tag{2.4}$$

第 3 章

ICP－SLAM

本章では ICL-SLAM の基となる ICP-SLAM について説明する．3.1 節で ICP-SLAM の基となる ICP アルゴリズムについて示し，3.2 節 SLAM の手法について具体的に示す．また，ICP アルゴリズムの問題点について 3.3 節に述べる．

3.1　ICP(Iterative Closest Point)アルゴリズム

外界センサを用いて環境中から情報を取得し，その中からいくつかの特徴点を抽出することを考える．その特徴点の位置がセンサによって取得可能でかつ不動であると仮定すると，計測領域中での基準特徴点の運動量は相対的にセンサ自身の運動量と考えることが可能となる．この運動量を求める手法の一つに ICP アルゴリズムがある[5].

ICP アルゴリズムは，複数の距離画像の間で重複して計測された部分を利用し，反復計算により対応点間の誤差関数を最小とするような移動パラメータを求める．具体的には，図 3.1 のようにモデルとなる点群$\boldsymbol{m}$とそれに似た形状の点群$\boldsymbol{d}$のマッチングを考える．まず，点群$\boldsymbol{d}$中の各点$d_i(1 \le \mathrm{i} \le N_2)$について点群$\boldsymbol{m}$中で最も近い点$m_j(1 \le \mathrm{j} \le N_1)$を探索し，対応点とする．そして，(3.1)式で表現される各対応点間の距離の 2 乗和Eが最小となる移動パラメータを求め，求められたパラメータで点群$\boldsymbol{d}$を変換する．移動パラメータ回転量$\boldsymbol{R}$と平行移動量$\boldsymbol{t}$はそれぞれ(3.2)式と(3.3)式に示す.

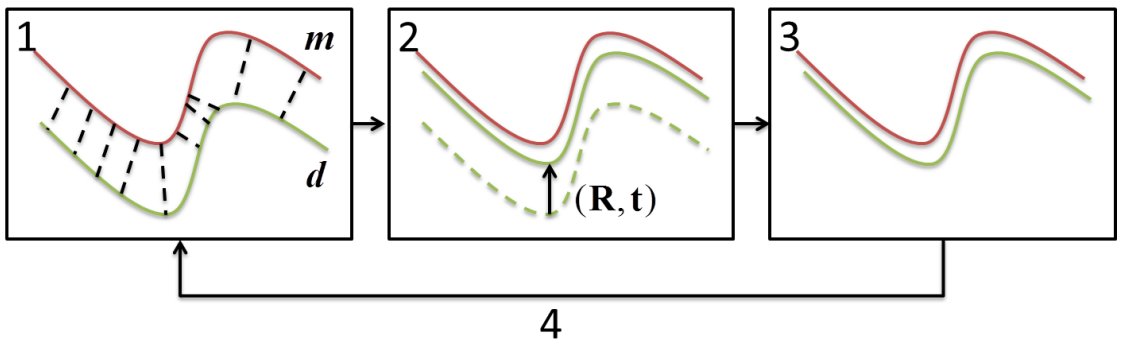


図 3.1　ICP アルゴリズムの手順

$$E(\boldsymbol{R},\boldsymbol{t}) = \sum_{j=1}^{N_2}\sum_{i=1}^{N_1}\left\|m_j - (\boldsymbol{R}d_i + \boldsymbol{t})\right\|^2 \qquad (3.1)$$

$$\boldsymbol{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{3.2}$$

$$\boldsymbol{t} = \begin{pmatrix} T_x \\ T_y \end{pmatrix} \tag{3.3}$$

まとめると手順は以下のようになる．

1. 点群$\boldsymbol{d}$の各点に対して最近傍点群$\boldsymbol{m}$を求める

2. 最近傍点群$\boldsymbol{m}$と点群$\boldsymbol{d}$の距離の誤差関数Eを最小にするパラメータ$(\boldsymbol{R}, \boldsymbol{t})$を求める

3. 点群$\boldsymbol{d}$を求まったパラメータ$(\boldsymbol{R}, \boldsymbol{t})$で変換する

4. 誤差関数Eが閾値以下になるまで反復計算

反復計算は，移動後に最初に見つけた対応点よりもより近い対応点を見つけた場合に対応点を変更してより近くへとマッチングさせるために行われる．

なお，ICP アルゴリズムにより位置合わせを行うためには，モデルの形状と比較する形状の間に十分な大きさの共通領域が含まれること，及び初期状態においてそれらの形状が大まかに位置合わせされていることが必要となる．

前者については，データ取得の回数を細かく取る，つまりロボットがそれほど移動していない状態で環境情報を取得する．ただし，あまり移動しなさ過ぎても逆にデータの処理回数が多くなり，無駄が生まれてしまうため，ロボットの移動速度や環境の複雑さによってデータ取得回数を変更したほうが良い．

また，後者については先ほど述べた内界センサであるエンコーダの情報を用いて，ある程度形状を近づけておくことを考える．

● 距離関数の決定

ICP アルゴリズムでは,位置合わせの評価基準に対応組間の距離の総和の大小を用いる．その対応間の距離は，小さいほどより正しく重ねあっていることが保障されているならばどのように定義しても良い．最も一般的な距離としてユークリッド距離を定義する場合が多い．2 次元空間中の 2 点P,Q間のユークリッド距離をd(P,Q)は(3.5)式のように表せる．

$$\vec{P}=(p_x,p_y),\ \vec{Q}=(q_x,q_y) \tag{3.4}$$

$$d(P,Q)=\sqrt{(p_x-q_x)^2+(p_y-q_y)^2} \tag{3.5}$$

● 対応する組の決定

ICP アルゴリズムでは，モデル形状間の対応する組を決定する．その際，比較を行う形状中のある点Pに対応した点を，モデル形状中の全点から距離の最小となる点を選び出す．これにより，対応組間の距離の総和が単調減少することが保障され，かつ ICP アルゴリズムの局所解への収束性を保証することができる．比較を行う形状中の 1 点に対し，距離が最小となるモデル形状上の点を発見するためには，その一点とモデル形状中の全点との距離を計算することになる．さらにそれを，比較を行う形状中の全点について行う必要があるため，この工程は ICP アルゴリズムにおいて最も計算量を必要とする部分となる．

この最近傍点を探索する部分を高速化する手法にK－d木がよく用いられている．

● 最近傍探索手法(K－d木)

図 3.2 に示すように，全ての点に対して総当たりで距離を計算し，最も近いものを探す手法を文字通り全探索あるいは線形探索という．元となるデータベースが小さい場合は問題ないが，その大きさや次元が増大すると急激に遅くなってしまう．全探索の処理時間はO(n)となる．

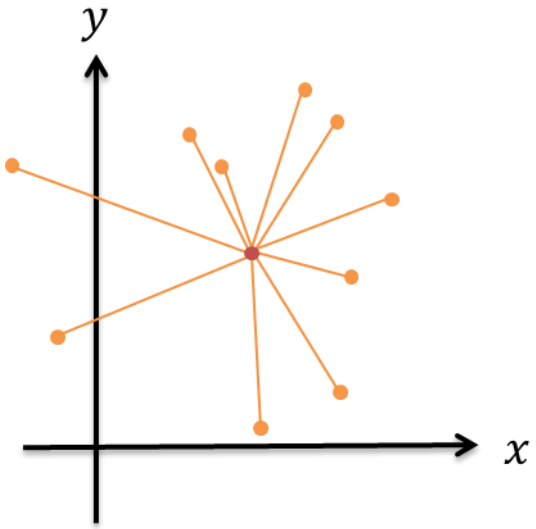


図 3.2　全探索

探索空間を分割することで効率よく探索を行う手法を空間分割，あるいは空間索引，空間アクセス法と呼ぶ．その中で最も単純な手法としてK－d (K-dimensional)木がある．K－d木は,k次元のユークリッド空間にある点を分類するデータ構造を扱う．探索空間を再帰的に半分に 2 分割していき，この木を根から葉に向かってたどることで処理される．つまり，2 分探索木の多次元版となる．

今回のように 2 次元の場合は以下のようになる．また，図 3.3 と図 3.4 にK－d木の探索例を示す．

1. x軸に平行に空間を 2 分割し，上下でノードを作成

2. y軸に平行に空間を 2 分割し，左右でノードを作成

3. x軸に平行に空間を 2 分割し，上下でノードを作成

4. 領域に点が 1 つになるまで繰り返し

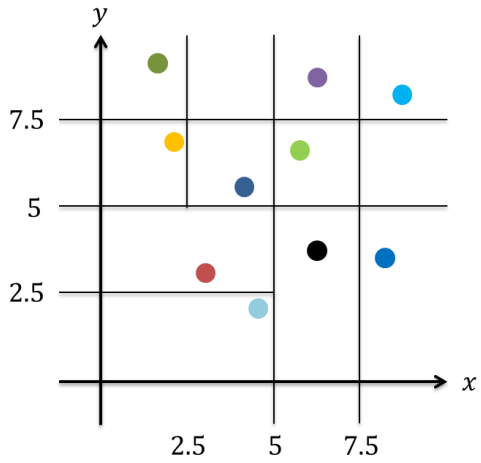


図 3.3　K－d木のイメージ(a)

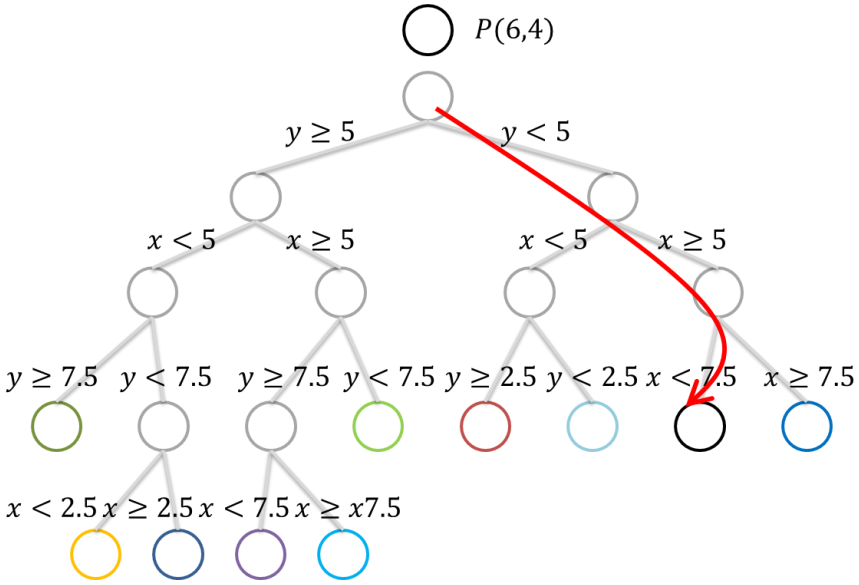


図 3.4　K－d木のイメージ(b)

このように分割を行うと，各ノードの段階で探索対象を絞ることができるため，効率が良い．今回の例では分かりやすくするために単純に真ん中で空間を分割しているが，点の座標値の中央値で分割するとK－d木は平衡木(深さがほぼ同じ木)となる．平衡木の場合，計算量は$O(\log n)$となるため，線形探索より早い．そのため，特にICPアルゴリズムの最近傍探索にはK－d木を用いることが多い．

3.2　ICP-SLAM

3.1 節で説明した ICP アルゴリズムは, 一般的に画像処理の位置合わせの分野で用いられることが多く，発掘された遺物の計測結果の統合等に広く用いられている．この ICP アルゴリズムを移動ロボットの SLAM に適用する手順を図 3.5 に示す.

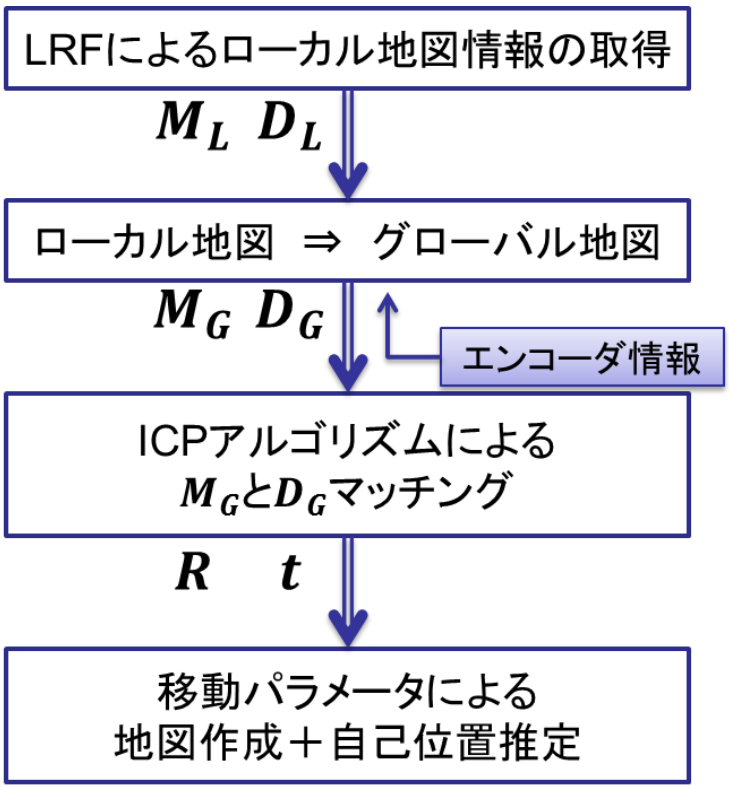


図 3.5　ICP-SLAM の手順

ある時刻tに取得したローカル座標系での LRF 情報を$\boldsymbol{M_L}$，ロボットが少し移動した後の時刻$t+1$に取得した LRF 情報を$\boldsymbol{D_L}$とする. この時点では各データはローカル座標系のため, それぞれグローバル座標系$\boldsymbol{M_G}, \boldsymbol{D_G}$に変換する．ローカル座標系からグローバル座標系への変換にはエンコーダの移動情報を用いる．エンコーダの情報は誤差が蓄積してしまうため完全ではないが，ICP アルゴリズムの性質上，マッチングを行う際の各データの初期位置はそれらの形状がおおまかに合わせられていないと計算時間がかかり，最悪の場合収束しないこともある．そのため，あらかじめおおまかな位置合わせをローカル座標系からグローバル座標系に変換することによって行う.

そして, この$\boldsymbol{M_G}, \boldsymbol{D_G}$に ICP アルゴリズムを適応する. ここでの目標は, 既存の地図情報$\boldsymbol{M_G}$に対し，新規に得られた地図情報$\boldsymbol{D_G}$をマッチングすることである．ICP アルゴリズムによって得られた移動パラメータにより$\boldsymbol{D_G}$を変換し，$\boldsymbol{M_G}$と統合することで新たにセンサによっ

てとらえられた情報を地図として書き足していく．その際に，ICP アルゴリズムによって得られた移動パラメータは自己位置推定の補正にも用いる．地図の更新は，図 3.6 に示すように，$\boldsymbol{D_G}$の各点からある一定距離に$\boldsymbol{M_G}$の点がない場合はその点を$\boldsymbol{M_G}$に追加する．そして，この$\boldsymbol{M_G}$をこの周期の地図として保存し，これを次の周期のモデルデータとして用いる．また，自己位置は(3.6)式により推定する．ここで，x，y，θはロボットの位置と向き，kは時間，Δx，Δy，$\Delta\theta$は ICP アルゴリズムの出力$(\boldsymbol{R},\boldsymbol{t})$，$\Delta$odmはオドメトリ計算によるロボットの位置と向きである．

このように ICP-SLAM では，時間差で取得された環境情報の形状をマッチングすることで地図作成と自己位置の推定を行っている．

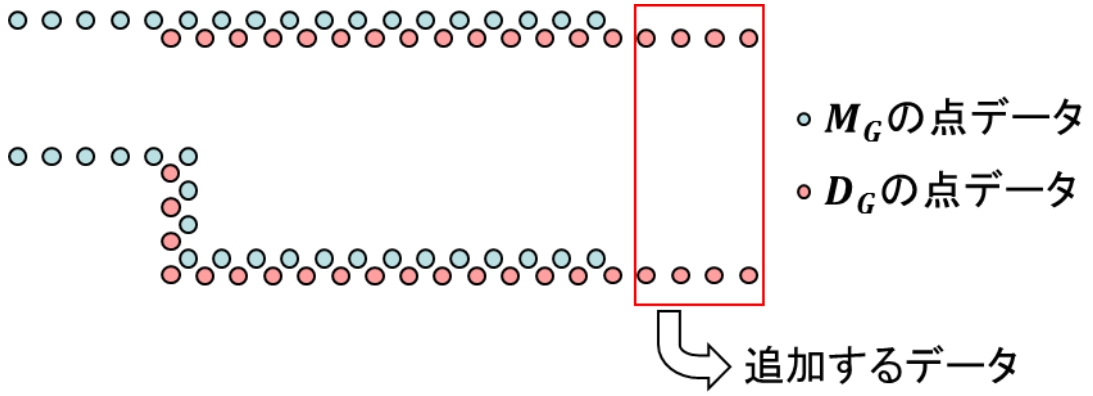


図 3.6　点データの追加

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} \cos\theta_k & \sin\theta_k & 0 \\ -\sin\theta_k & \cos\theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} + \Delta\mathrm{odm} \qquad (3.6)$$

3.3　ICP アルゴリズムの問題点

これまでに説明した ICP アルゴリズムの基礎は Chen and Medioni 及び Besl and McKay により同時期に考案された方法である[6][7]．発表以後，ICP アルゴリズムには様々な改良がなされており，多くの変種がある．Rusinkiewicz and Levoy は 2001 年の時点でこれらを分類し，比較することで高速な ICP アルゴリズムを実現するための指針を示している[8]．

文献[8]において ICP アルゴリズムの問題点として，低速なこととノイズに弱いことが挙げられている．また，その原因はデータ量が多いこととノイズによって間違った対応点を選んでしまうことであると述べられている．これは，図 3.7 のように最近傍の点を求める段階で，もしノイズなどによる局所的な出っ張りなどが片方のデータにのみ存在した場合，そこに間違った対応点が集中してしまうために，正しい対応点を見つけるまで計算の繰り返し回数が多くなり，移動パラメータを求めるのに時間がかかってしまうからである．

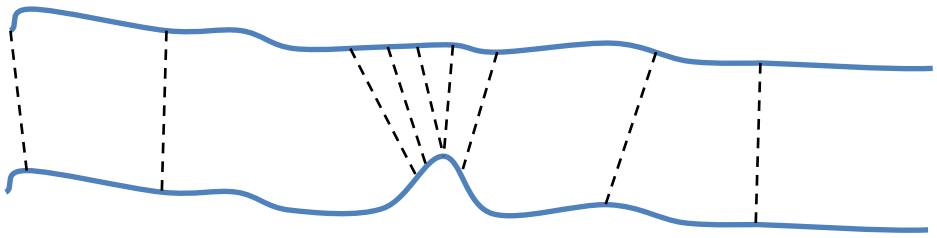


図 3.7　最近傍点探索の弱点

- LRF の誤計測

前述のような突発的なノイズは LRF の計測においてよく計測されることが実験から分かっている．図 3.10，図 3.11 に例を示す．計測環境は図 3.8，図 3.9 に示すように直線のみで表現できるような形状であるにも拘わらず，LRF の計測結果によっては，図 3.10 のように壁を歪んで計測してしまうことや，図 3.11 のように明らかに環境外の本来計測されることのない位置に数点のデータを計測してしまうことがある．

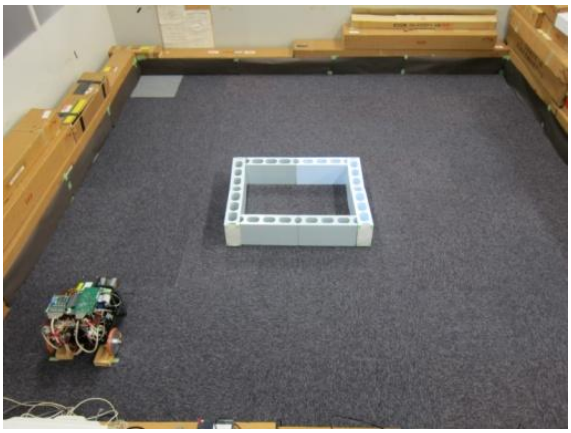


図 3.8　実験環境

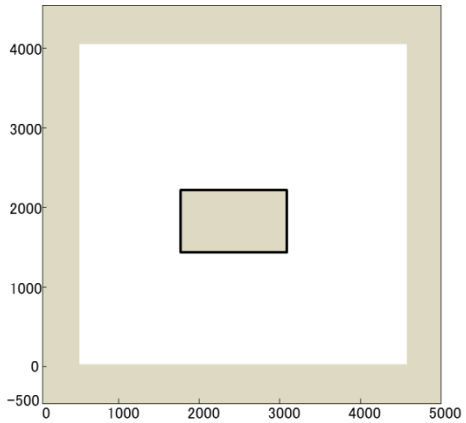


図 3.9　実験環境イメージ

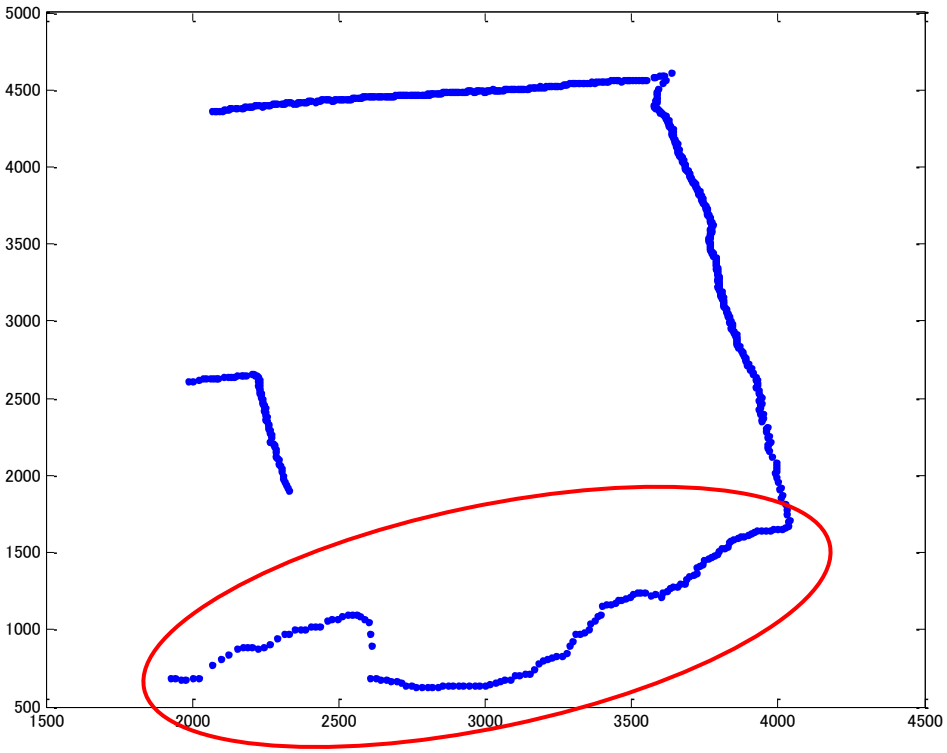


図 3.10　LRF の誤計測の一例(直線の誤計測)

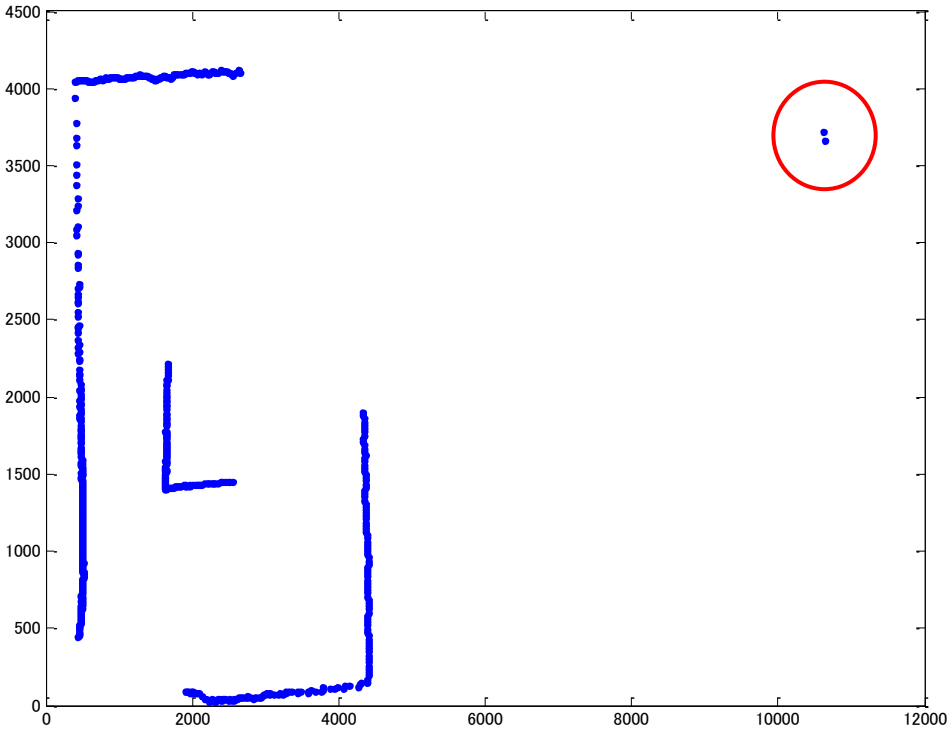


図 3.11　LRF の誤計測の一例(計測されるはずのない位置の点)

LRF が誤計測をする原因としては様々考えられる．例えば，LRF はレーザーの反射を利用して環境情報を計測しているため，反射率の違う物質の場合うまく計測できない．また，照明の影響を大きく受けてしまうことも知られており，その影響で太陽光に左右される野外での使用は向いていないといった意見もある．その他に，ロボットが動きながら計測しているためロボットの振動に影響されることや，使用している LRF 自体の精度の問題などが挙げられる．しかし，はっきりとこれが原因と述べられるようなものはなく，さらに毎回の計測に同じ傾向のノイズが計測されることもない．

上記に述べたようにこれらのノイズにより，ICP アルゴリズムで間違ったマッチングを行ってしまい，その影響で ICP-SLAM の精度を下げてしまう．そこで，ノイズの消去，線抽出と線の質判定，限定により ICP アルゴリズムの問題点である計算時間の問題を解決する．

第 4 章

ICL－SLAM

本章では前章の 3.3 節で述べた ICP アルゴリズムの問題点を解決した ICL-SLAM について説明する．4.1 節では問題解決の流れ，4.2 節ではデータ量の削減について示す．4.3 節で点データから直線生成を行う際の全体構成を簡単に示し，4.4 節で物体検出の方法，4.5 節で直線抽出アルゴリズム，4.6 節で限定アルゴリズム，4.7 節で修正アルゴリズムの説明を行う．また，4.8 節で ICL-SLAM の問題点について述べる．

4.1　ICP アルゴリズムの問題点の解決

ICP アルゴリズムにおいて問題となっていることは，計算時間が遅いことと，マッチングが局所的なノイズに影響されてしまうことの 2 つである．そこでまず，計算時間が遅くなってしまう問題を計測データ量の削減で解決する．次に ICP アルゴリズムを用いる前に取得した環境情報のノイズをある程度判別し，消去する．図 4.1 に ICP-SLAM の流れを，図 4.2 に ICL-SLAM の流れを示す．

図 4.1　ICP-SLAM の流れ

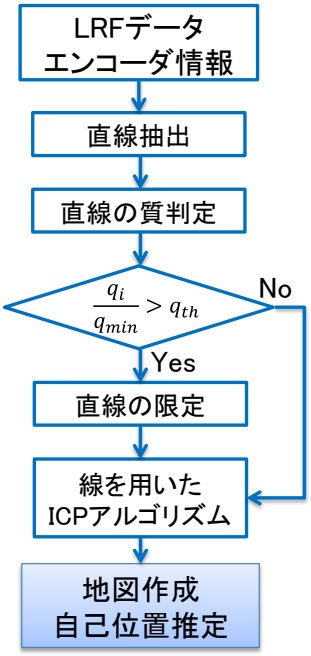


図 4.2　ICL-SLAM の流れ

ICP-SLAM では，図 4.1 のように LRF とエンコーダで取得した情報をそのまま ICP アルゴリズムに使用して地図作成と自己位置推定を行っている．図 4.2 の ICL-SLAM では，LRF から得られた点データから直線を抽出する．この段階で，LRF の点データを直線化することにより，図 3.11 のような誤計測された小さなノイズを除去でき，かつデータ量を大幅に減らすことができる．データ量の削減に関しては 4.2 節で詳しく説明する．

次に，抽出された線の各々に対して品質を決定し，その品質が低いと判断された直線を限定することにより，品質の低い直線を除去する．直線抽出の具体的な方法と直線の品質については第 5 章で詳しく説明する．その後，残った直線の始点と終点の点データを ICP アルゴリズムに用いて地図作成と自己位置推定を行う．

このように，“信頼性”の低い直線を除去したデータから ICP アルゴリズムを用いて作成した地図と，元のデータから直接 ICP アルゴリズムを用いて作成した地図を比較する．本研究では，図 4.1 を従来の ICP-SLAM，そして図 4.2 の一連の流れを ICL-SLAM (Iterative Closest Line SLAM)と呼ぶこととする．

4.2　データ量の削減

- 点地図

LRF から得られる点情報をそのまま地図として用いる場合，地図の座標系は，図 4.3 に示すようになる．各計測点はグローバル座標系のx座標とy座標の組として(4.1)式のように保存される．また，それらの点の集合として(4.2)式のような地図表現M_Pを保存する．nは点の数を意味する．

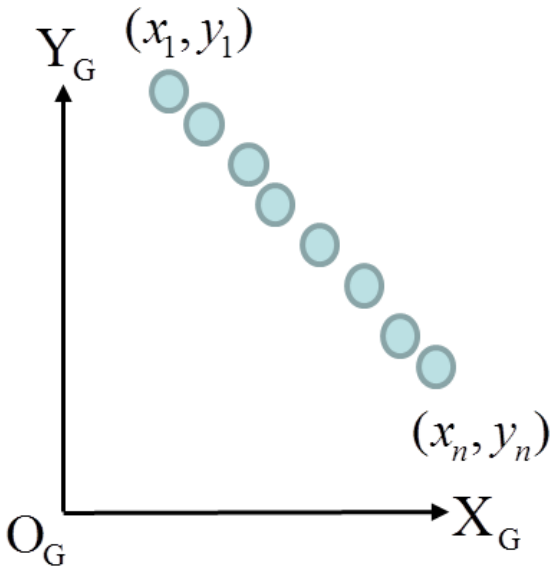


図 4.3　点地図パラメータ

$$\boldsymbol{P}_i = [x_i, y_i]^{\mathrm{T}} \tag{4.1}$$

$$\boldsymbol{M}_P = [\boldsymbol{P}_1 \ \ \boldsymbol{P}_2 \ \ \cdots \ \ \boldsymbol{P}_n \ ]^{\mathrm{T}} \tag{4.2}$$

点地図の場合，LRF の一度のスキャンにより約 750 点のデータを得る．ロボットが大域的地図を作成する場合，このスキャンを何度も繰り返して地図を更新する．そのため，地図の情報量がとても大きくなってしまうという問題がある．地図を保存しておくメモリ容量や，地図を更新する際の処理を考えると，地図の情報量は少ないほうが良い．

● 線地図

これに対し，本研究では LRF の計測点から抽出される直線を用いる．グローバル座標系の直線地図のパラメータを図 4.4 に示す．直線を表現するパラメータは，各座標系の原点から直線までの垂直距離ρとx軸から垂直線までの角度φ，そして直線の始点と終点の座標のそれぞれ$(start_x, start_y)$, (end_x, end_y)とする．(4.3)式のように直線を取得し，後に(4.4)式のように地図表現M_Lとして保存される．mは直線の数を意味する．

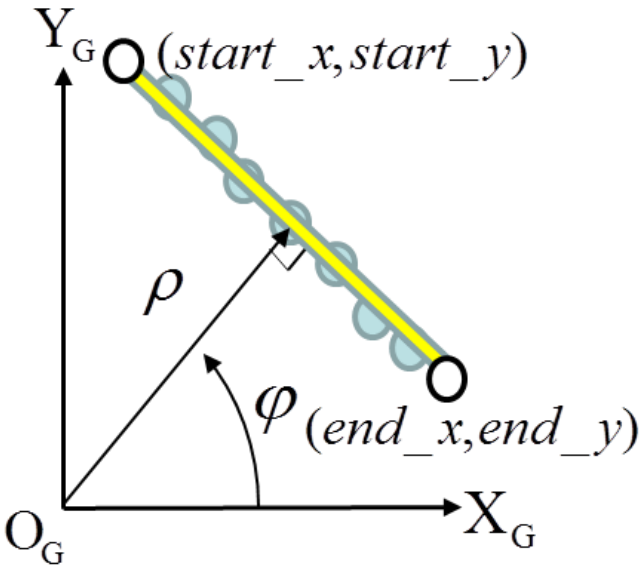


図 4.4　線地図パラメータ

$$L_i = \begin{bmatrix} \rho_i \\ \varphi_i \\ start\_x_i \\ start\_y_i \\ end\_x_i \\ end\_y_i \end{bmatrix} \tag{4.3}$$

$$M_L = [L_1 \quad L_2 \quad \dots \quad L_m]^T \tag{4.4}$$

線地図へ変換した場合，一度のスキャンで多い場合でも数十本の線データで環境を表現できる．そのため，直線抽出によってかなりのデータの削減が可能である．これにより ICP アルゴリズムの計算時間の低減を行う．

4.3　直線生成の構成

直線生成の流れを図 4.5 に示す．以降で，各処理の詳細について解説する．

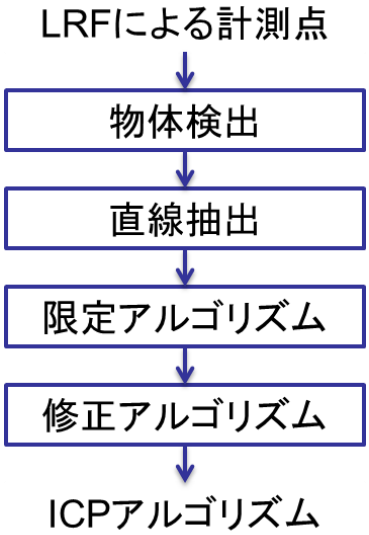


図 4.5　直線生成の流れ

- 物体検出
  連続する点群に物体の区切り点を見つけることで同じ物体に属する点集合を検出
- 直線抽出
  物体検出により判別した点集合に関して地図の特徴である直線を抽出
- 限定アルゴリズム
  抽出した直線が SLAM に必要か判定し，評価の悪い直線を除去
- 修正アルゴリズム
  屋内環境の幾何学的な条件を用いて各直線の角度を修正

4.4　物体検出

LRF 計測点から直線抽出を行うためには，全ての計測点をどこまでが同じ物体に属する点かを判別する．すなわち，物体との間の分割点を計測点列の距離の変化により決定する．LRFの情報取得の順番と同じようにLRFから向かって右側の点から順番に見ていくことを考えたとき，分割点と考えられる点は著しい距離の変化が存在する．このような距離の変化は，LRF の計測可能領域に物体がない場合に点が途切れた場合や，LRF から見て奥の物体の手前に部分的に異なる物体が位置している場合に著しく計測距離が変化する場合が考えられる．よって，その距離の変化を利用して連続する点の距離の変化を閾値D_{max}と比較し，その値より変化が大きかった場合，分割点とする．すなわち(4.5)式ように連続する点が閾値に収まれば同じ物体に属する点と判定する．

$$\|p_i - p_{i-1}\| \leq D_{max} \Rightarrow \{p_i, p_{i-1}\} \in OBJ_i \tag{4.5}$$

ゆえに，閾値D_{max}をどのように設定するかが，物体検出で最も重要となる．閾値は実行環境など様々な要因によって適切な値が変化するために一定の値に決定するのが難しい．そこで本研究では，可変閾値を用いる[9]．LRF の位置を原点とした可変閾値D_{max}の設定方法を図 4.6 に示す．

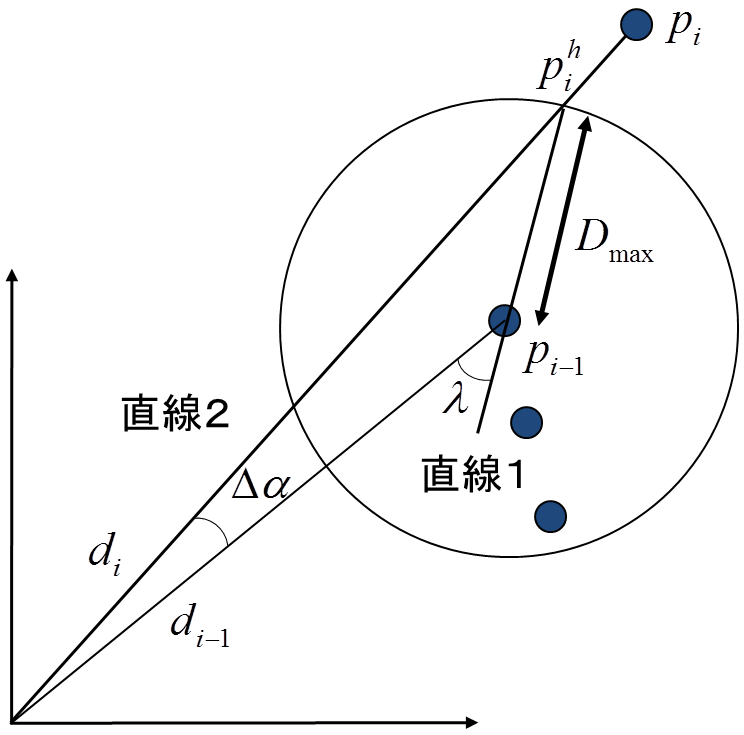


図 4.6　可変閾値D_{max}の設定

図 4.6 より連続する 2 点p_{i-1},p_iの原点からの距離は，それぞれd_{i-1},d_iで，その間の角度は$\Delta\alpha$である．可変閾値D_{max}による判定は連続する点p_{i-1},p_iの間の角度$\Delta\alpha$と点p_{i-1}の距離d_{i-1}によって行われる．閾値D_{max}は，前の点p_{i-1}の点を通り原点と点p_{i-1}を結んだ線分に対して偏差角度λ傾けた仮想線(直線 1)と原点と点p_iを結んだ延長線(直線 2)との交点をp_i^hとすると，交点p_i^hと点p_{i-1}間の距離とする．この距離より遠い場合，すなわち図 4.6 のように点p_{i-1}を中心とした円の外に点p_iが存在する場合，2 つを区切り点として決定する．ゆえに閾値D_{max}は次式で得られる．

$$D_{max} = \left\| p_i^h - p_{i-1} \right\| = d_{i-1} \frac{\sin(\Delta\alpha)}{\sin(\lambda - \Delta\alpha)} \tag{4.6}$$

しかし，この閾値をそのまま適用すると物体との距離が近い場合，点間の距離がきわめて近く閾値が適切に設定されにくいため，(4.7)式のように閾値に一定値kを加える．ここで，一定値kは 45[mm]とした．

$$D_{max} = \left\| p_i^h - p_{i-1} \right\| + k = d_{i-1} \frac{\sin(\Delta\alpha)}{\sin(\lambda - \Delta\alpha)} + k \tag{4.7}$$

4.5　直線抽出

物体検出で計測点から分割点を見つけたら，次に各々の分割点間の点集合から物体を表現する直線を抽出する．直線を抽出する方法として，移動ロボットにおいてよく用いられている Split and Merge を用いる[10][11]．この方法を用いる理由は他の主な直線抽出アルゴリズムに比べて計算時間が短く，正確に環境を直線で表現できる傾向が強いからである．Split and Merge の処理手順をそれぞれ図 4.7 と図 4.8 に示す．

1.　Split (分割)

(a). 点集合の両端点に直線を引き，その直線から最大垂直距離の点d_{max}を探索

(b). 点d_{max}までの垂直距離を閾値D_{th}と比較し閾値より大きい場合，点d_{max}で点集合を分割

(c). (a).(b)の処理を閾値D_{th}より遠い点がなくなるまで繰り返す

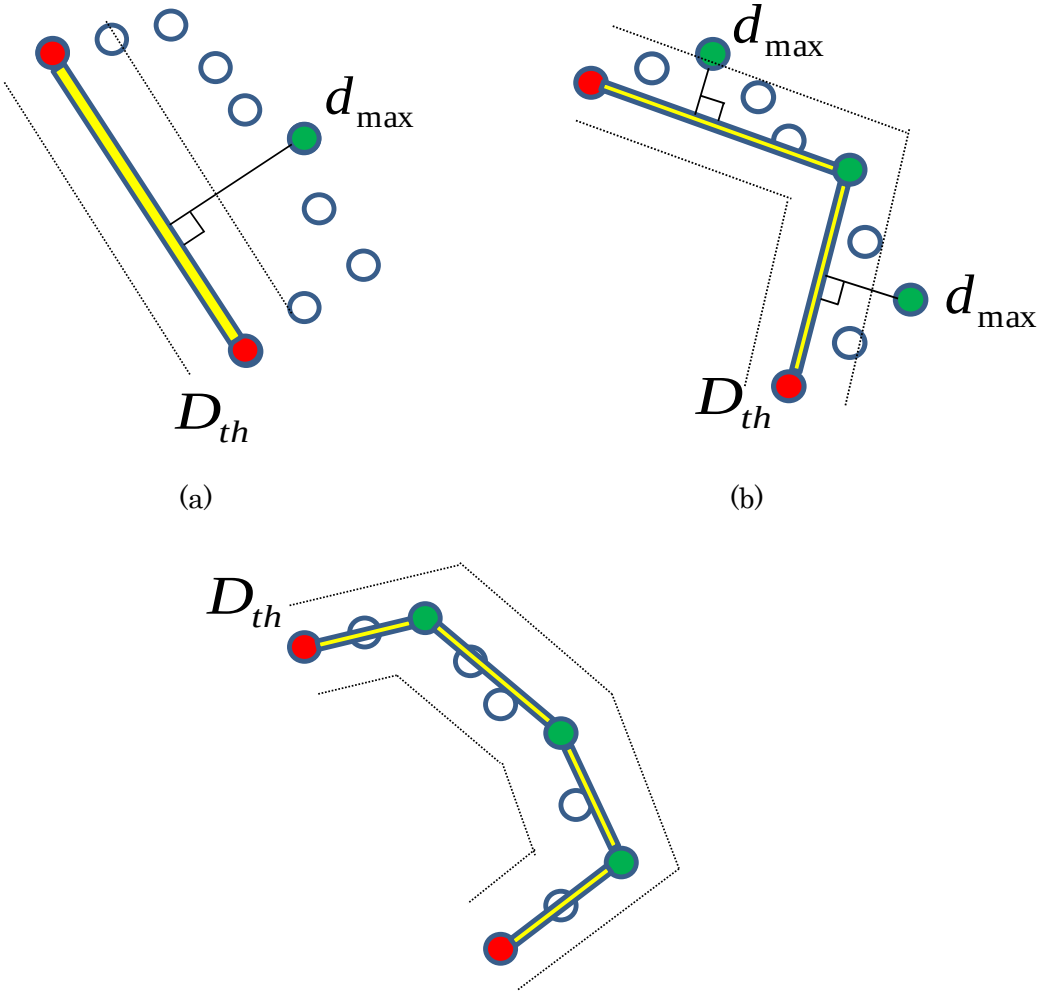


図 4.7　Split 処理の流れ

２．　Merge (結合)

(a). Split による直線の結果から連続する直線の幅が閾値D_{th}の中に収まるかを探索

(b). 閾値D_{th}に収まったとき直線を結合

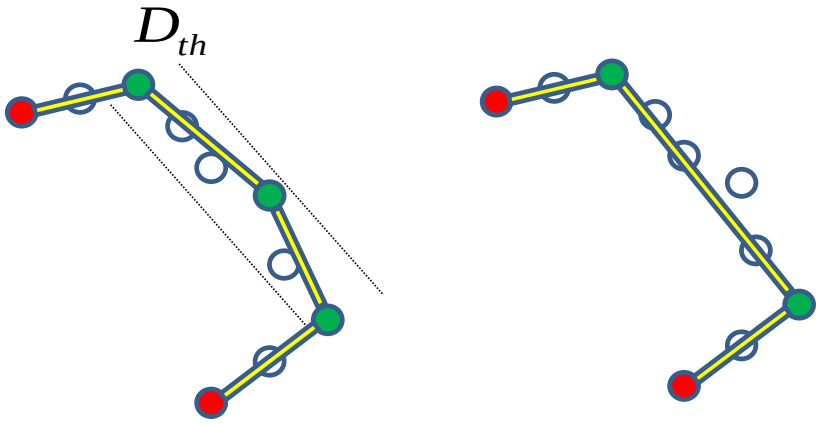


図 4.8　Merge 処理の流れ

以上の処理により大まかな直線抽出作業が終了するが，この時点では分割点同士を単純に繋げただけであるため，点集合に対する直線の引き方としては適当ではない．よって，最終的に直線をどのように引くのかを決定する．その方法には最小二乗法を用いる．直線から見た直線に属する一つの計測点$\{x_i, y_i\}$までの垂直距離は，図 4.9 から(4.8)式のように表現される．

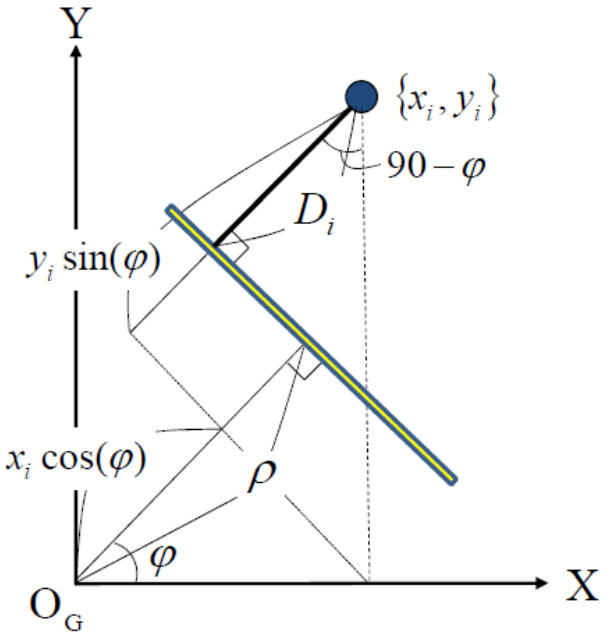


図 4.9　直線と点の距離関係

$$D_i = |\rho - x_i \cos(\varphi) - y_i \sin(\varphi)| \tag{4.8}$$

最良な直線は全ての計測点において直線からの距離の誤差の二乗の合計である次式を最小化することである.

$$D_0 = \sum D_i^2 \tag{4.9}$$

次にD_0を直線のパラメータρ, φに関して偏微分し，それが 0 になる$\hat{\rho}, \hat{\varphi}$を求める．すなわち最良な直線のパラメータは次式のように求められる．ここで，$\bar{x}, \bar{y}$は全ての計測点の平均値，σは計測点の標準偏差を表す.

$$\hat{\rho} = \bar{x}\cos(\hat{\varphi}) + \bar{y}\sin(\hat{\varphi}) \tag{4.10}$$

$$\hat{\varphi} = \tan^{-1}\left(\frac{-2\sigma_{xy}^2}{\sigma_y^2 - \sigma_x^2}\right) \tag{4.11}$$

ここまでの直線生成処理を適用した結果を図 4.10 に示す. 図 4.10 より物体検出により五つの物体に分けられると判別され，直線抽出により計測点から直線を抽出し，環境形状を表現できている.

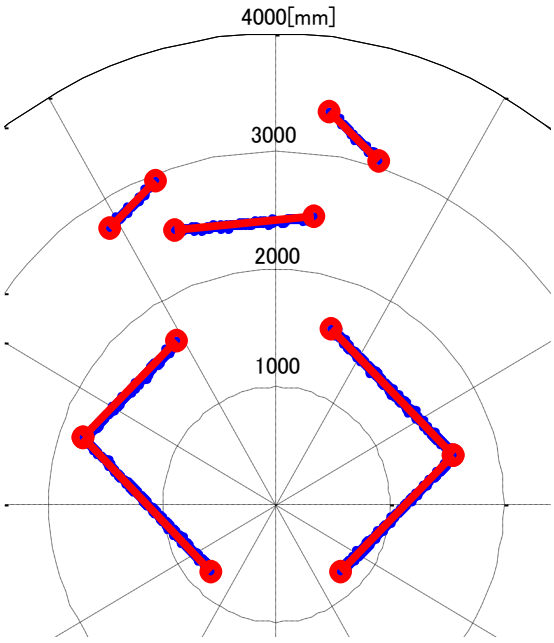


図 4.10　直線抽出結果

4.6　限定アルゴリズム

- 誤差量の指標：品質の定義

各直線に誤差量の指標を設定する．この指標を“品質q”と呼び，(4.12)式で表す．σ_{min}は直線に対する直線を構成する点の垂直距離の標準偏差，nは直線を構成する点の数である．品質の高低を直線と点のイメージで表すと図 4.11 のようになる．つまり同じ点数から構成される直線でもばらつきが小さい方が，ばらつきが同じでも直線を構成する点数が多いほど，品質qは値としては小さくなり品質が高いとする．この品質を用いることにより直線の限定と修正を試みる．

$$q = \frac{\sigma_{min}}{n^2} \tag{4.12}$$

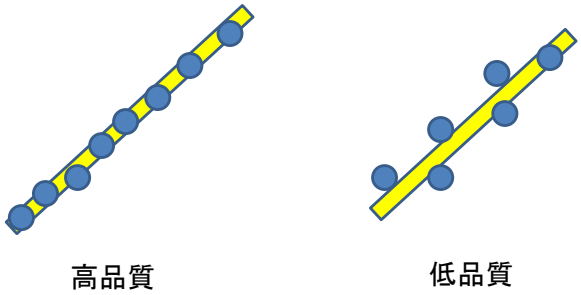


図 4.11　直線の品質

- 限定する直線の判定

抽出した直線を限定するか否かを決定する．その判定は(4.12)式のような場合に限定を行う．q_{min}は，一回の直線抽出において最も高品質(値が最小)な直線の品質値である．そしてq_iは，最も高品質な直線以外の直線の品質値である．ゆえに最も高品質な直線の値に対する他の直線の比率Nを計算する．この比率Nを任意に決定する閾値q_{th}と比較し，閾値より比率Nが大きい場合，全体地図に対する精度が低いと判断し，その直線を削除し地図として採用しない．この判定により環境によって判定基準であるq_{min}が変化するため，環境の複雑さの違いについても対応が可能である．

$$\frac{q_i}{n^2} = N > q_{th} \tag{4.12}$$

限定アルゴリズムを適用した例を示す. 図 4.12 の各直線に対して品質の計算結果を表 4.1 に示す．品質の値は図 4.12 の番号付けされた直線と対応しており，品質の性質上，LRF(原点)に近い直線は品質が高く遠い直線は品質が低く設定される．その中でも最も高品質な直線の品質q_{min}は表の結果のように直線 7 となり，この結果では閾値$q_{th} = 10$で実験したため直線 7 の品質値を 10 倍した値以下の直線に地図が限定される．ゆえに直線 3 と直線 5 が削除され図 4.13 のような結果となり，高品質な直線だけに限定することが可能となる．

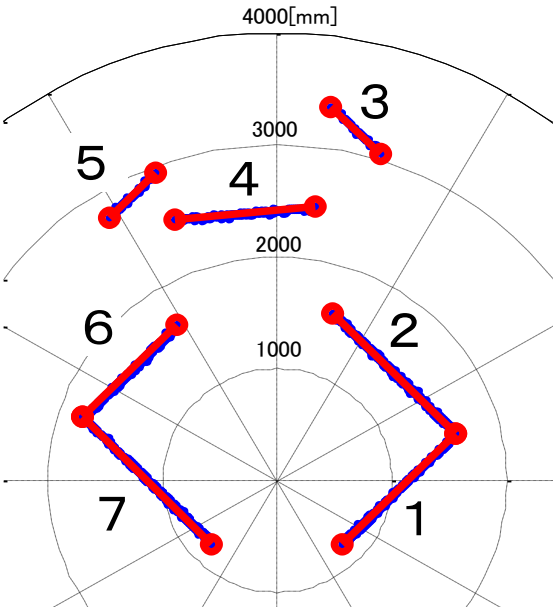


図 4.12　直線の特徴の番号付け

表 4.1　直線の品質

特徴	標準偏差σ_{min}	点の数n	品質$q(\times 10^{-3})$
直線 1	67.1	169	2.4
直線 2	104	159	4.1
直線 3	114.5	27	157
直線 4	82.3	80	12.9
直線 5	107.3	31	111.6
直線 6	102.5	108	8.8
直線 7	70.7	177	2.3

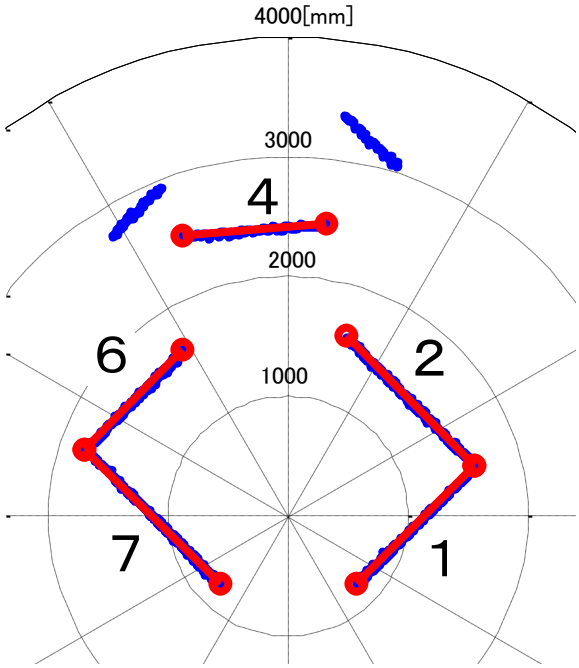


図 4.13　直線限定の結果

4.7　修正アルゴリズム

● 分割点修正アルゴリズム

Split and Merge は直線抽出アルゴリズムとして性能が高いが欠点も存在する．その欠点とは，環境に対する直線の再現性に，ばらつきが発生しやすいことである．その理由は常に直線から一番離れた点を探すというアルゴリズムの性質上，はずれ点に影響されやすいからである．このことは SLAM の推定にも悪影響を及ぼす．よって，その欠点をカバーするように直線の精度を向上させる手法を考える．

最初に行うことは，限定アルゴリズムと同様に抽出した直線の品質(式(4.12))を計算する．Split and Merge によって抽出された直線において品質が低くなる原因として考えられるのは，点集合の適切な分割点が選択されてないことが挙げられる．分割点修正アルゴリズムでは，Split and Merge で選択された分割点の近傍の点を仮の分割点とし，再度直線を引き，品質を計算，前の直線の品質より高かった場合，分割点の修正を行うことで直線の環境の再現性を高める．図を用いて分割点修正アルゴリズムの流れを説明する．図 4.14 に Split and Merge で形成された直線のイメージを示す．点と直線の関係を再度説明すれば，図のように分割点を共有して点集合が分けられ(楕円に囲われた集合)，それぞれの集合に対して最小二乗法を用いて直線が引かれる．加えて，この時点での 2 つの直線の品質q_{a1}, q_{a2}を計算しておく．

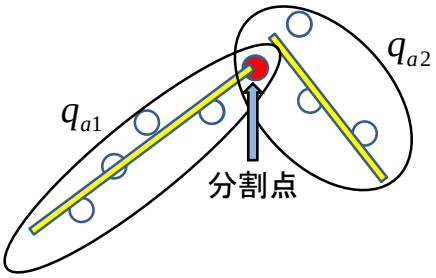


図 4.14　計測点と直線抽出

ここでは説明の簡単化のために分割点の両近傍の点を仮の分割点 1 と仮の分割点 2 とする．そして，図 4.12 のように仮の分割点に従って再度直線を引き，それぞれ分割点 1 の直線の品質q_{b1}, q_{b2}，分割点 2 の直線の品質q_{c1}, q_{c2}を計算し最初の品質との比較を行う．すると，(4.13)式のように分割点 2 の直線の品質q_{c1}, q_{c2}が両直線について最も高品質であると分かる．このように両直線の品質がともに向上したとき，直線の修正を行う．ゆえに今回の例の場合，分割点 2 の直線を採用し分割点修正アルゴリズムを終了する．このアルゴリズムにより結果的に再現性の高い直線を得られる．実際のアルゴリズムでは，最初に設定された分割点から一定距離の点を仮の分割点として修正を試みる．

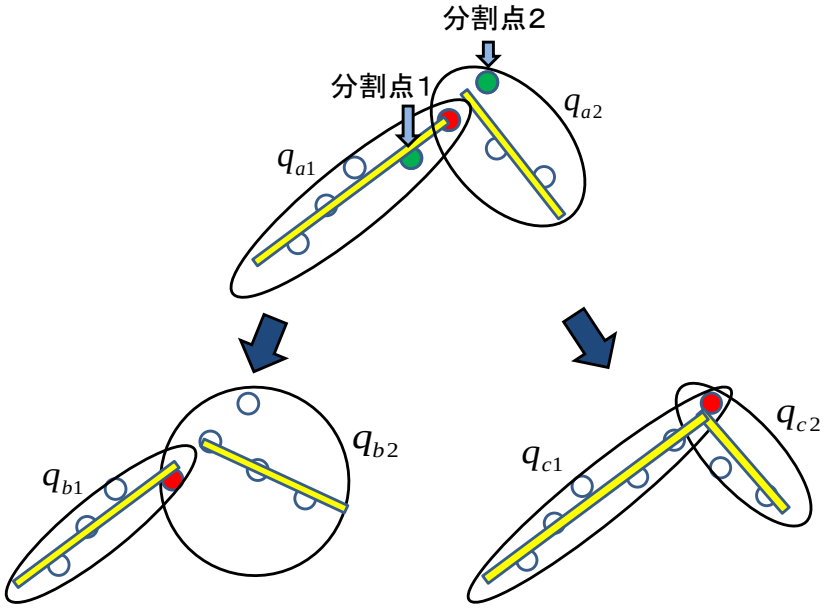


図 4.15　分割点修正アルゴリズムの流れ

$$q_{c1} < q_{a1} < q_{b1} \;\; AND \;\; q_{c2} < q_{a2} < q_{b2} \tag{4.13}$$

分割点修正アルゴリズムを適用した例を図 4.16 と表 4.2 に示す．適用前と適用後の直線 1・2 の品質を比較すると，どちらの直線も品質値が低くなり，高品質な直線が得られている．よって精度向上に寄与している．

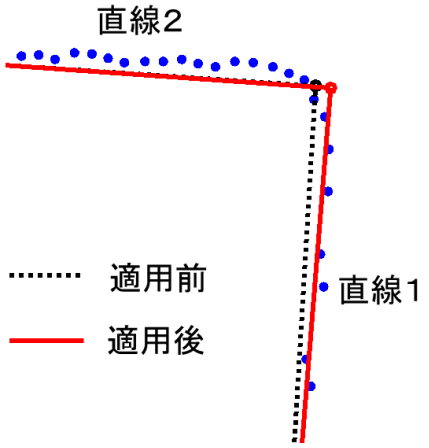


図 4.16　分割点修正アルゴリズム結果

表 4.2　直線の品質

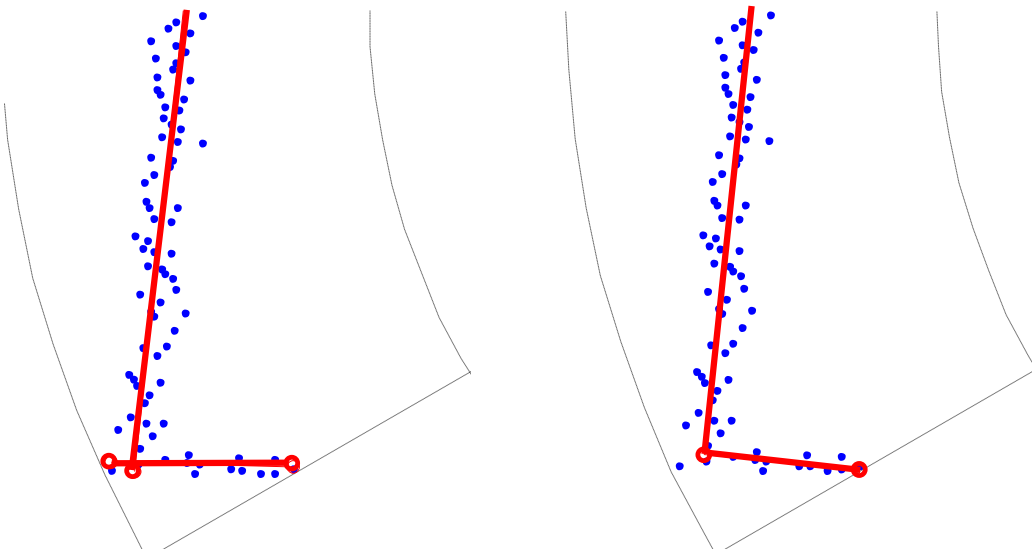
特徴	適用前	適用後
直線 1	26.07	9.54
直線 2	7.2	7

● 角度修正アルゴリズム

角度修正アルゴリズムでは，屋内環境における幾何学的な特性，すなわち部屋の隅が直角であることや向かい合う壁が平行であるという特性を利用し直線を改善する．つまり抽出した直線をある基準となる直線に対する相対的な角度を 0 度・90 度・180 度・270 度と変更する．このアルゴリズムにより，直線の周期間でのばらつきを抑える．このアルゴリズムにおいて重要となるのは，基準となる直線をどのように決めるかである．基準となる直線に実際の環境とのずれが存在すると全体にずれが生じてしまうため，最も環境の再現性の高い直線を基準とする必要がある．よって，基準となる直線には最も高品質な直線を採用する．ゆえに，基準以外の直線の角度φ_mは，(4.14)式のように設定される．φ_{ref} は基準となる直線の角度である．しかし，基準に対して直角や平行関係に程遠い場合は，むやみにアルゴリズムを適用しない．

$$\varphi_m = \varphi_{ref} + \{0,90,180,270\} \tag{4.14}$$

角度修正アルゴリズムを適用した例を図に示す．図 4.17 (a)の適用前の二つの直線が交差する角度は鋭角であるが，図 4.17(b)の適用後の二つの交差する直線の角度は直角となり角の部分を正確に表現できている．また，適用前後で直線の交差点が結合しているのは現実環境としてあり得ないためである．



(a)適応前　　(b)適応後

図 4.17　角度修正アルゴリズム適用前後

4.8　ICL－SLAM の問題点

4.7 節での角度修正アルゴリズムの後に, 直線の端点情報を使用して ICP アルゴリズムを行うものが ICL-SLAM である．ICL-SLAM では直線の端点情報のみを扱っているため，問題が発生する．その問題というのは，ロボットの位置によっては同じ障害物を表す直線でも長さが異なる場合があるというものである．図 4.18 にその例を示す．直線 1 と直線 1’，直線 2 と直線 2’が同じ障害物を表す直線で，直線 3’が右の環境でのみ計測された直線である．直線 2 と直線 2’は同じ長さであるが，直線 1 と直線 1’は同じ障害物を表す直線だが，長さが異なっている．これは他の障害物の影になってしまっているために，LRF で障害物の一部分が計測できないことが原因である．

直線 1 と直線 1’のように同じ障害物でも計測された直線の長さが異なる場合，ICP アルゴリズムで正しい結果が得られない恐れがある．ICL-SLAM では直線の端点情報を使用して ICP アルゴリズムを実行するが，直線の長さが異なる場合は端点の位置も異なる．このような場合において ICP アルゴリズムの(3.1)式の誤差関数を計算すると，正確に合う位置ではなく，わずかにずれた位置が最小となってしまう．わずかにでも位置がずれてしまうと，それ以降の周期で ICP アルゴリズムを実行した場合，そのずれた位置に対してマッチングを行ってしまい精度が低下してしまうことが考えられる．また，このように直線の長さが異なる場合は移動ロボットにおいて頻繁に発生するため，周期を重ねるごとにずれが累積し，精度がますます低下してしまう．

上記のように，直線の端点情報を用いて ICP アルゴリズムを実行すると本来とはずれた位置にマッチングされてしまい，ICL-SLAM の精度が下がってしまう．そこで，直線の端点情報ではなく，直線の方向ベクトルを用いることで問題を解決する．

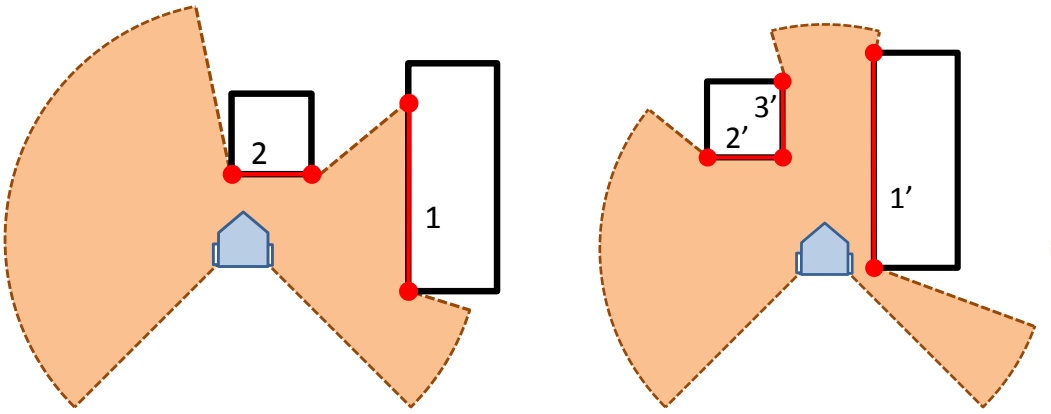


図 4.18　計測によって直線の長さが変化する場合

第 5 章

提案手法

本章では前章の 4.8 節において述べた問題を解決するための提案手法について説明する．5.1 節で提案手法の流れについて述べ，5.2 節で方向ベクトル方向ベクトルについて，5.3 節で方向ベクトルを用いた ICL アルゴリズムについて述べる．

5.1　提案手法の流れ

ICL-SLAM の問題点は ICP アルゴリズムに直線の端点情報を用いているために正確にマッチングを行うことが出来ないことである．そこで，本研究では直線の端点情報ではなく，直線の方向ベクトルを用いて ICP アルゴリズムを実行する ICL アルゴリズムを SLAM に適用することを提案する[12]．ICL アルゴリズムの流れを図 5.1 に示す．

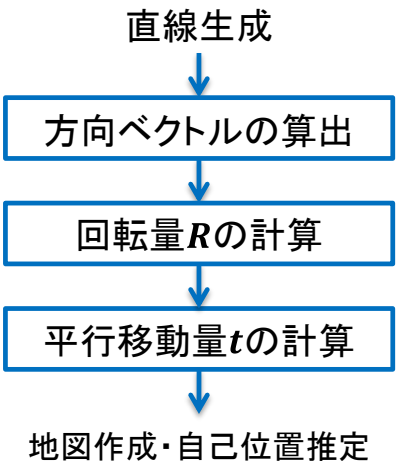


図 5.1　ICL アルゴリズムの流れ

- 方向ベクトルの算出
  直線生成で得られた直線の方向ベクトルを算出
- 回転量$\boldsymbol{R}$の計算
  2 周期間の直線の方向ベクトルから回転量$\boldsymbol{R}$を計算
- 平行移動量$\boldsymbol{t}$の計算
  回転量$\boldsymbol{R}$で直線を回転させたのちに平行移動量$\boldsymbol{T}$を計算

5.2　方向ベクトル

方向ベクトルとは直線が延びている方向を示すベクトルのことである. 図 5.2 のように始点 S と終点 E を端点とする直線 A があった場合，この直線の方向ベクトル$\boldsymbol{A_{dir}}$は(5.1)式と定義される．しかし，(5.1)式では方向が同じであっても直線の長さが異なる場合は方向ベクトルが変わってしまう．そこで，正規化することで直線の長さが異なる場合でも同じ方向ベクトルとなるようにする．正規化した方向ベクトル$\boldsymbol{A_u}$は単位方向ベクトルと呼ばれ，式(5.2)により求まる．本稿では特に記載がない限り，方向ベクトルとある場合は単位方向ベクトルのことを指すこととする.

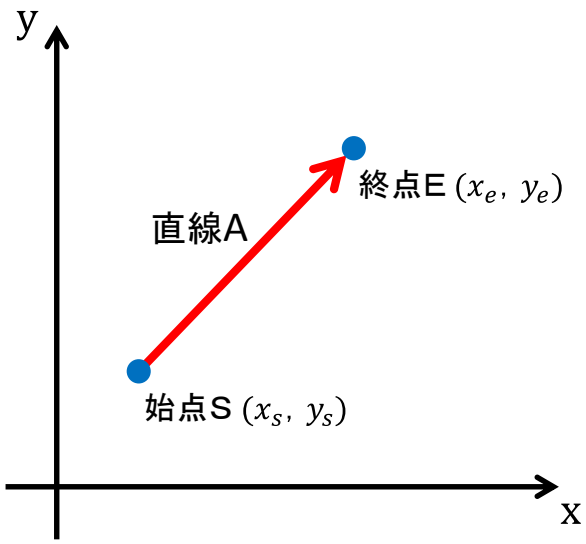


図 5.2　方向ベクトル

$$\boldsymbol{A}_{dir} = [x_s - x_e,\ \ y_s - y_e]^T \tag{5.1}$$

$$\boldsymbol{A}_u = [x_s - x_e,\ \ y_s - y_e]^T / |\boldsymbol{A}_{dir}| \tag{5.2}$$

5.3　方向ベクトルを用いた ICL アルゴリズム

直線の端点情報を用いた ICP アルゴリズムを，直線の方向ベクトルを用いた ICL アルゴリズムへの修正を行うことが本研究の提案部分である．

- 方向ベクトルを用いた回転量$\boldsymbol{R}$の計算

図 5.2 のように 2 つの直線があった場合，この 2 本の直線のなす角θは方向ベクトルにより簡易に求めることが出来る．直線 1 と直線 2 の方向ベクトルをそれぞれA_{u1}，A_{u2}とすると，ベクトルの内積を用いてθは(5.3)式のように表すことができる．また，方向ベクトルは直線の位置が座標内のどこにあっても延びる方向が一緒ならば同じ値となる．これにより，直線の位置によらず 2 本の直線のなす角θは方向ベクトルのみで求めることができる．得られたθから(5.4)式により回転量$\boldsymbol{R}$が求まる．

以上のことから，2 つの周期間の直線の方向ベクトルを用いることで，移動ロボットの回転量$\boldsymbol{R}$を求めることができる．

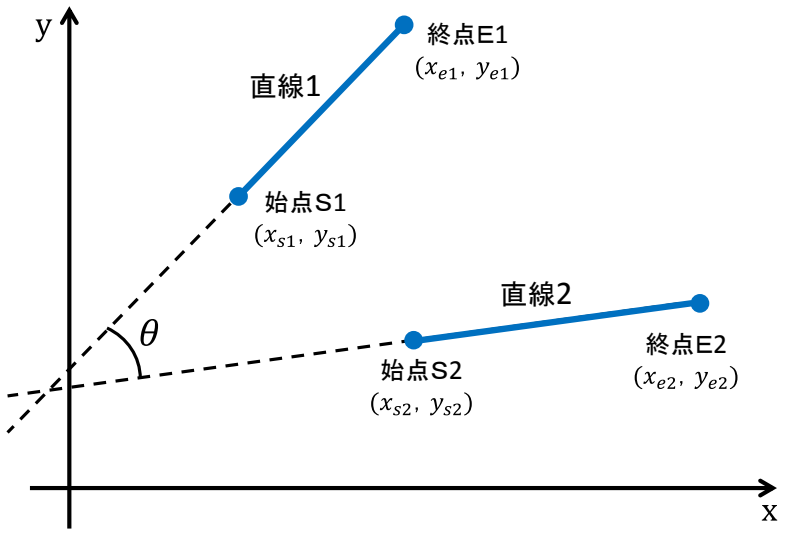


図 5.3　2 本の直線となす角θ

$$\theta = \cos^{-1}\frac{A_{u1}\cdot A_{u2}}{|A_{u1}||A_{u2}|} \quad (-1 \le \theta \le 1) \tag{5.3}$$

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{5.4}$$

ただ，回転量$\boldsymbol{R}$の計算は直線 1 組に対して 1 回行っており，直線の組み合わせが複数あればその組み合わせの数だけ$\boldsymbol{R}$が計算される．直線の組み合わせが複数ある場合の例を図 5.4 に示す．異なる周期に LRF で計測した結果から直線生成を行った結果が(a)と(b)であったとする．(a)の直線 A と(b)の直線 A'が直線の組である．ほかの直線も同様に，直線 B と直線 B'，直線 C と直線 C'，直線 D と直線 D'という組み合わせになっている．まず，直線 A と直線 A'の回転量$\boldsymbol{R}$を求め，次に直線 B と直線 B'の回転量$\boldsymbol{R}$を求める．同様に直線 C と直線 C'，直線 D と直線 D' の回転量$\boldsymbol{R}$を求める．図 5.4 の場合では回転量$\boldsymbol{R}$は 4 つ求まる．このため，導出した複数の$\boldsymbol{R}$から最適なものを選択する必要がある．

そこで，(5.5)式の誤差関数$E(\boldsymbol{R})$を新たに定義する．ここで，A_{mj}はこれまでに得られた直線の方向ベクトル，A_{dj}は今周期で得られた直線の方向ベクトルを示す．計算したすべての回転量$\boldsymbol{R}$に対してこの誤差関数$E(\boldsymbol{R})$を求める．この誤差関数$E(\boldsymbol{R})$が最小となる場合の回転量$\boldsymbol{R}$が最も正確な値であると判断し，回転量$\boldsymbol{R}$を決定する．

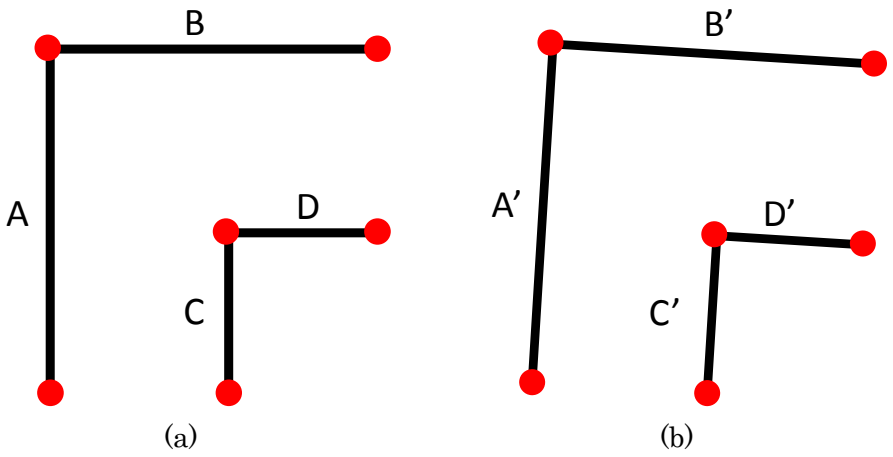


図 5.4　直線が複数ある場合

$$E(\boldsymbol{R}) = \sum_{j=1}^{N_2} \sum_{i=1}^{N_1} \left\| A_{mj} - \boldsymbol{R} \times A_{di} \right\|^2 \tag{5.5}$$

しかし、同じ障害物を表す直線どうしで計算しないと正しい回転量を求めることができない．このため，同じ障害物を表す直線を対応付ける必要がある．直線の対応付けには図 4.4 の直線パラメータのうち，垂直距離ρと垂直線までの角度φを用いる．同じ障害物を表す直線であればρとφの値が近い値になるため，閾値を設定して 2 本の直線パラメータρ，φの差が閾値以内であれば同じ障害物を表す直線として対応付ける．しかし，これだけでは異なる障害物を表す直線であっても対応付けてしまう恐れがある．図 5.5 において直線 1 と直線 2 はそれぞれ異なる障害物を表す直線とする．この 2 本の直線は図のようにρとφの値は同じであり，方向ベクトルも同じとなる．このため，これらの値によって異なる障害物を表す直線かどうかを判断することができない．

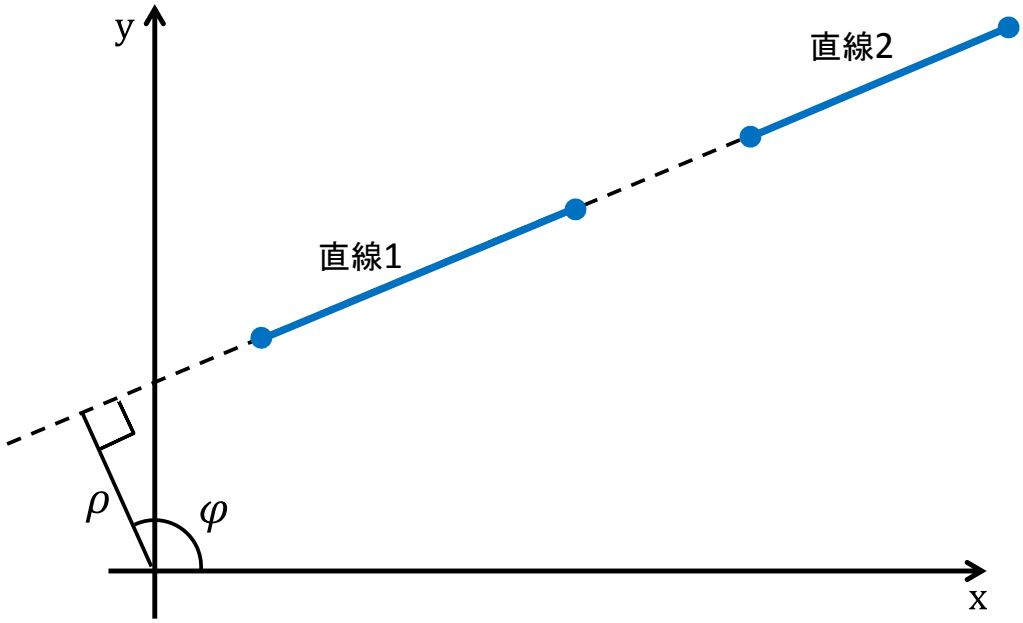
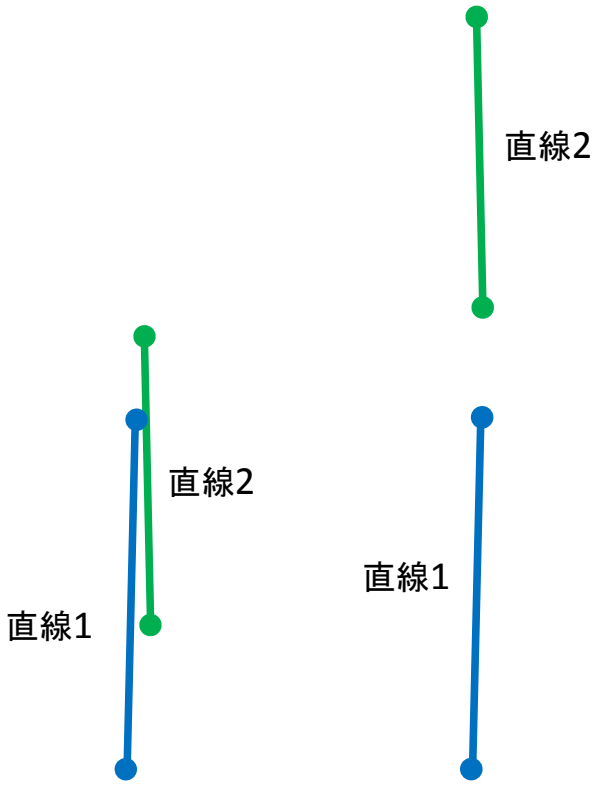


図 5.5　ρとφが同じ値の 2 本の直線

2 本の直線が同一の障害物を表すものであるかどうかを判断するために直線の端点情報を利用する．図 5.6(a)が同一の障害物であると判断する場合で，図 5.6(b)が異なる障害物と判断する場合である．この図のように，2 本の直線が一部でも重なっていれば同一の障害物判断して対応付けを行い，重なっている部分がなければ異なる障害物であると判断して対応付けを行わない．これを実現するためには 2 本の直線の端点を見る．1 本の直線の端点の間にもう 1 本の直線の端点の少なくともどちらかが入っていればいい．

以上により，同じ障害物を示す直線の対応付けができる．



(a)同一の障害物　　(b)異なる障害物

図 5.6　2 本の直線が同一の障害物を示すかどうかの判断

平行移動量$\boldsymbol{t}$の計算

平行移動量$\boldsymbol{t}$の計算はこれまで通り ICP アルゴリズムを使用する．方向ベクトルや直線パラメータρ，φでは直線が座標内のどの位置にあるのかを示すことができないため，直線の端点情報を使用する．$\boldsymbol{t}$の計算は従来とほぼ同じように(3.1)式の誤差関数を最小とするものを求める．この際に回転量$\boldsymbol{R}$は先述のように方向ベクトルから求めたものを使用する．

第 6 章　検証実験

6.1　実験内容と条件

実験内容は，LRF を搭載した独立操舵二輪移動ロボット MIUBOT が，ある環境において初期位置から目標地点まで移動する．その際に得られた車輪エンコーダ情報と LRF 計測情報を基に, 第 5 章で提案した手法である ICL アルゴリズムを用いた ICL-SLAM を実行し，地図作成と自己位置推定を行う．また，有効性を確認するためにオドメトリの計算結果から作成した地図と自己位置推定の結果, それに ICP アルゴリズムを用いた ICL-SLAM も従来手法として行い比較する．

オフライン実機実験とシミュレーションによる実験を行った．オフライン実機実験環境は, 図 6.1 の俯瞰写真のように周囲を囲われた 4[m]×4[m]の環境中央に障害物を配置した．MIUBOT の走行経路は図 6.2 に示すように左手前を初期位置とし，最初に直進して左奥へ進み障害物を回り込むように右折するように経路を設定した．実験パラメータ条件は表 6.1 に示す．

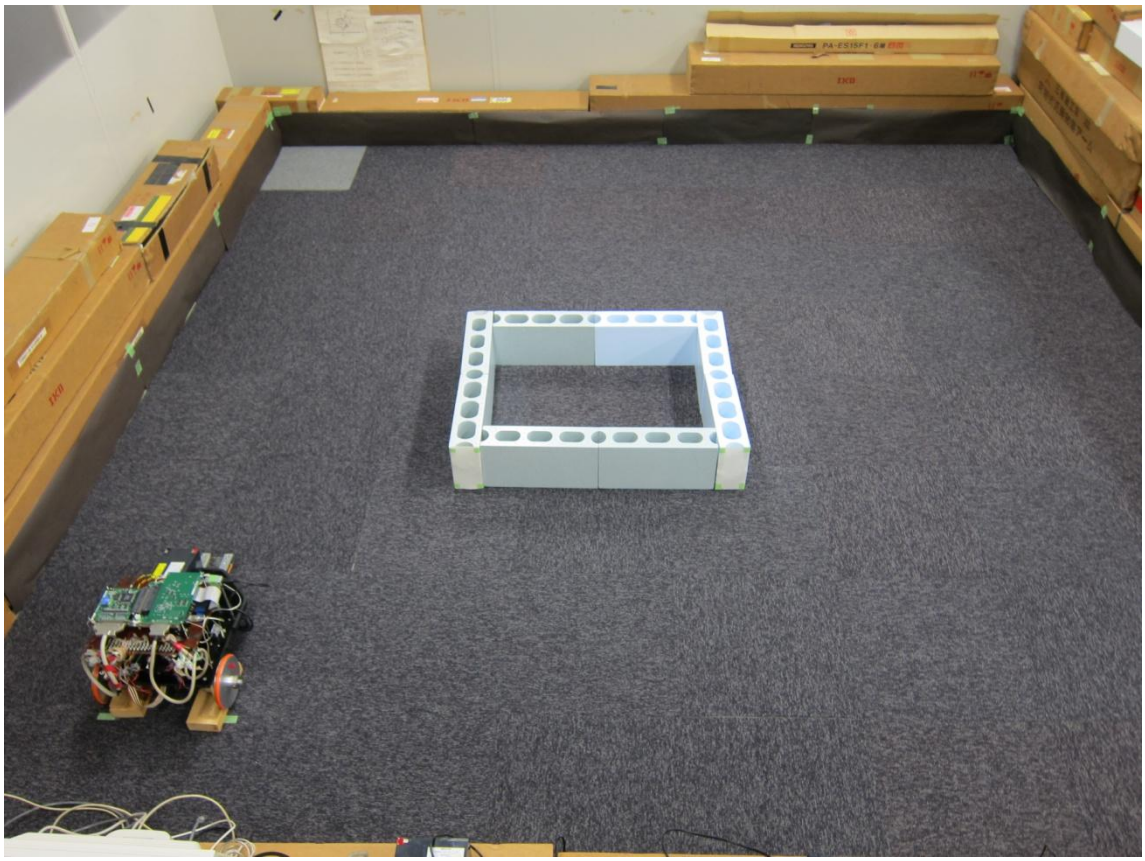


図 6.1　実験環境

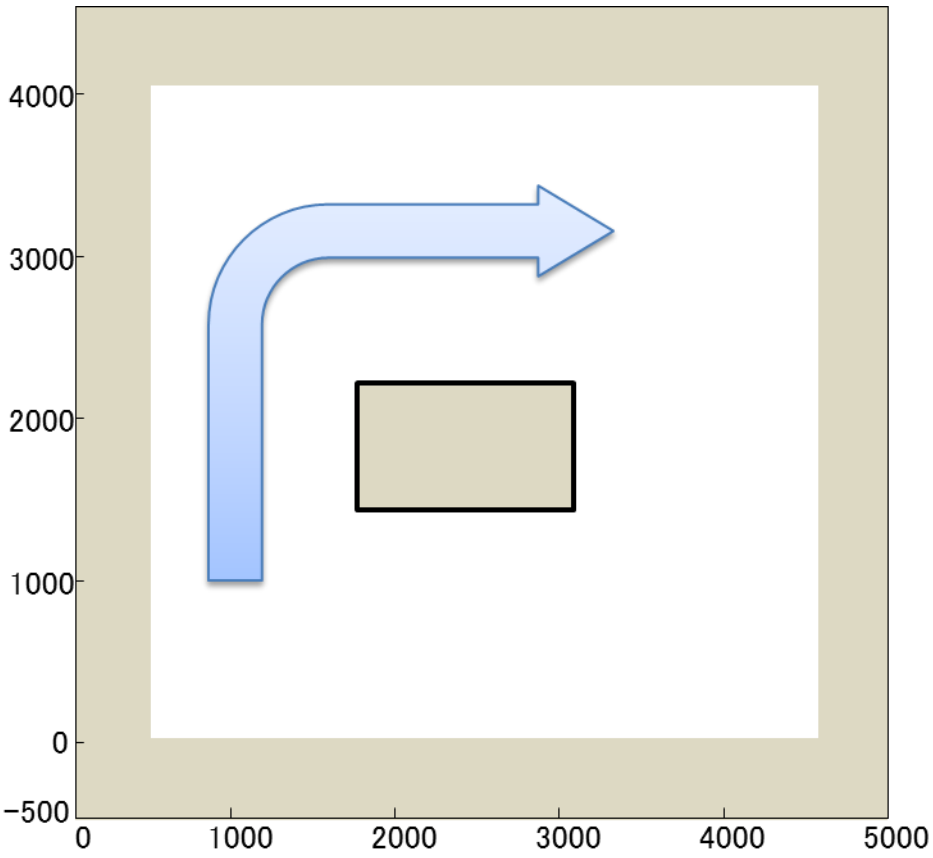


図 6.2　経路イメージ

表 6.1　実験パラメータ条件

車体速度	$v = 0.1$ m/s
初期ロボット変数行列	$R_0 = [0 \quad 0 \quad 0 \quad 0 \quad 0]^T$
Split and Merge の閾値	$d_{max} = 0.15$ m
可変閾値の偏差角	$\lambda = 10$ deg
限定アルゴリズムの閾値	$q_{th} = 10$
直線の対応付けのρの閾値	$d_\rho = 0.15$ m
直線の対応付けのφの閾値	$d_\varphi = 10$ deg

シミュレーション実験環境は，図 6.3 のような 10[m]×10[m]の環境に障害物を複数配置した．走行経路は図中の点の位置を初期位置とし，破線の経路を反時計回りに走行し初期位置へと戻ってくる経路を設定した．実験パラメータ条件は表 6.2 に示す．

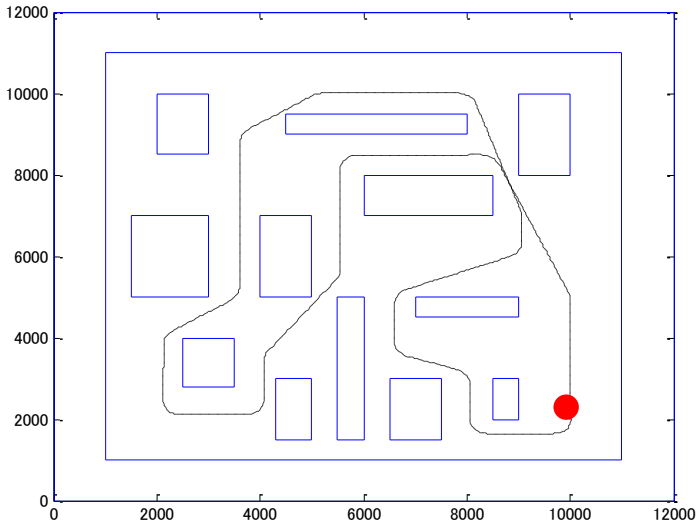


図 6.3　シミュレーション環境と走行経路

表 6.2　実験パラメータ条件

車体速度	$v = 0.5$ m/s
初期ロボット変数行列	$R_0 = [0 \quad 0 \quad 0 \quad 0 \quad 0]^T$
Split and Merge の閾値	$d_{max} = 0.15$ m
可変閾値の偏差角	$\lambda = 10$ deg
限定アルゴリズムの閾値	$q_{th} = 10$
直線の対応付けのρの閾値	$d_\rho = 0.15$ m
直線の対応付けのφの閾値	$d_\varphi = 10$ deg

6.2　実験結果

6.2.1　実機実験結果

- 地図生成結果

まずはオフライン実機実験の結果を示す．オドメトリの計算結果からの地図と従来法，提案手法を行った結果，生成された地図をそれぞれ図 6.4 と図 6.5，図 6.6 に示す．SLAM に用いる LRF のデータはどれも 750 回分のスキャンデータを使用した．

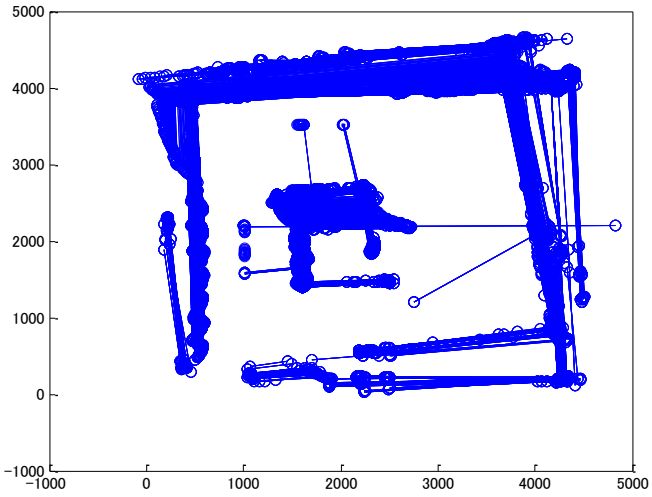


図 6.4　オドメトリの結果からの生成地図

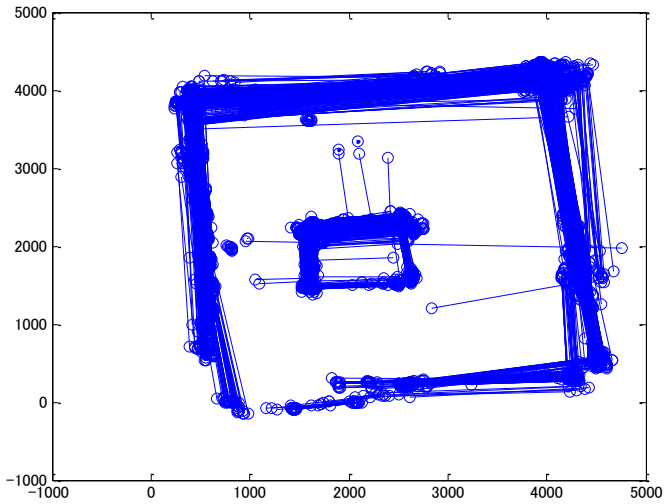


図 6.5　従来の ICL-SLAM による生成地図

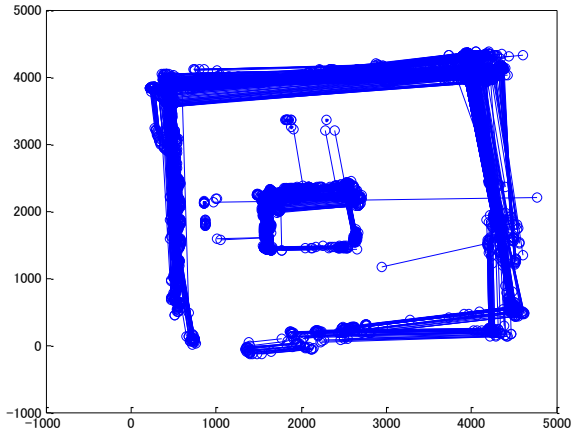


図 6.6　提案手法の ICL-SLAM による生成地図

図 6.4，図 6.5，図 6.6 より生成された地図の精度は，従来法と提案法ともにオドメトリの結果のものよりも向上している．従来法と提案法では，ほぼ同じ精度であるが，左下部分や上下の壁を表す直線が従来法に比べて一致している部分が多く精度の向上が確認できた．また，ロボットの移動経路上に突発的に計測される小さなノイズがあり，間違った地図の生成に影響を及ぼしている．

● 自己位置推定結果

従来法と提案手法による自己位置推定の結果をそれぞれ図 6.5，図 6.6 に示す．図中の緑の線がオドメトリの計算結果，赤色の線が ICL-SLAM による自己位置推定結果である．ロボットの走行軌道を黒色の破線で表しているが，オドメトリの計算結果と重なっているため表示されていない．

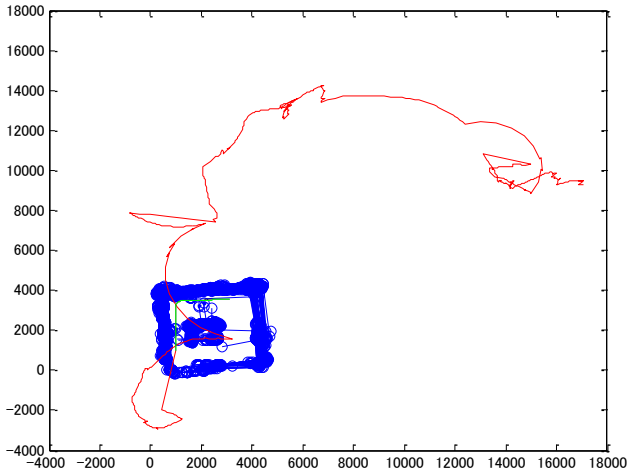


図 6.5　従来の ICL-SLAM による自己位置推定結果

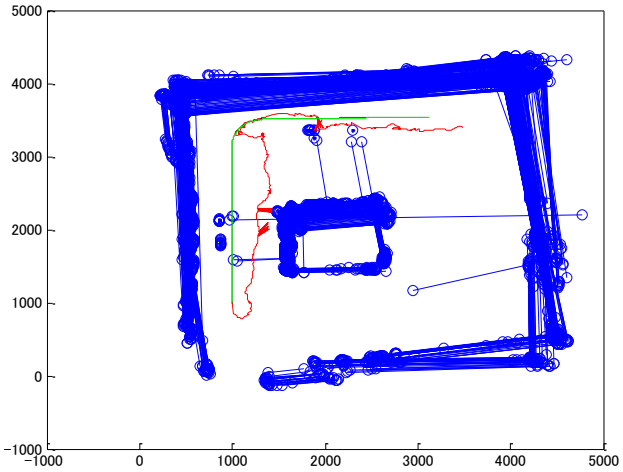


図 6.6　提案法による ICL-SLAM の自己位置推定結果

図 6.5 の従来法での自己位置推定結果は大きくずれて環境の外にまで行ってしまっている．一方，図 6.6 の提案法では若干のずれはあるものの比較的精度よく自己位置の推定ができている．

6.2.2　シミュレーション実験結果

- 地図作成結果

シミュレーション実験の結果を示す．オドメトリの計算結果からの地図と従来法，提案手法を行った結果，生成された地図をそれぞれ図 6.7 と図 6.8，図 6.9 に示す．SLAM に用いる LRF のデータはどれも 870 回分のスキャンデータを使用した．

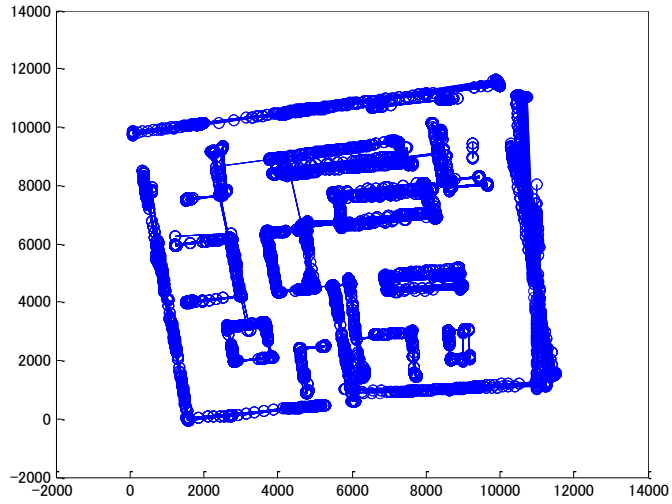


図 6.7　オドメトリの結果からの生成地図

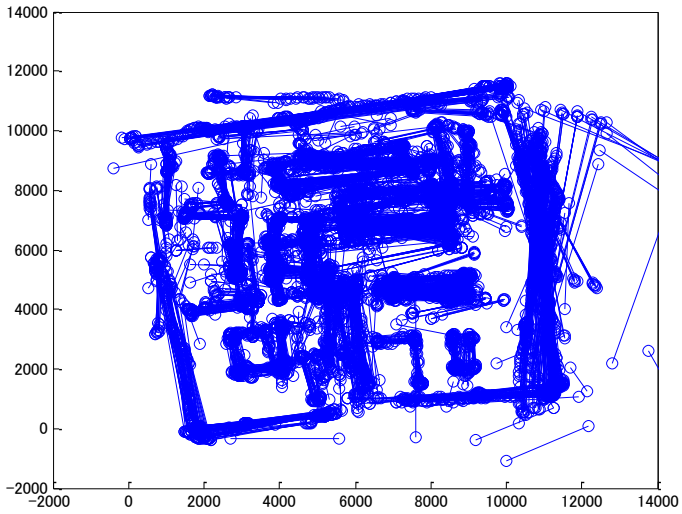


図 6.8　従来の ICL-SLAM による生成地図

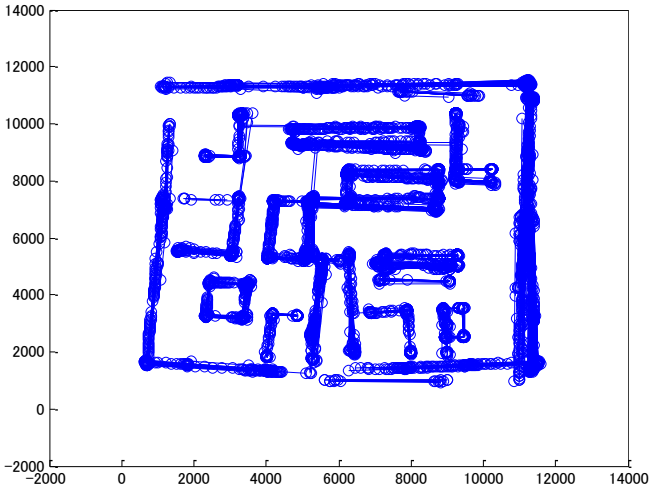


図 6.9　提案手法の ICL-SLAM による自己位置推定結果

図 6.7，図 6.8，図 6.9 の結果から，従来法では環境形状の再現ができておらず正確に ICL-SLAM を行うことができておらず，オドメトリから生成した地図よりも精度が低下している．一方，提案法ではオドメトリから生成したものから角度が正しく修正されており，精度が向上している．ただ，左下部分が若干歪んでしまっている．

● 自己位置推定結果

従来法と提案手法による自己位置推定の結果をそれぞれ図 6.10，図 6.11 に示す．実機実験の結果と同様に図中の緑の線がオドメトリの計算結果，赤色の線が ICL-SLAM による自己位置推定結果である．ロボットの走行軌道を黒色の破線で表しているが，こちらもオドメトリの計算結果と重なっているため表示されていない．

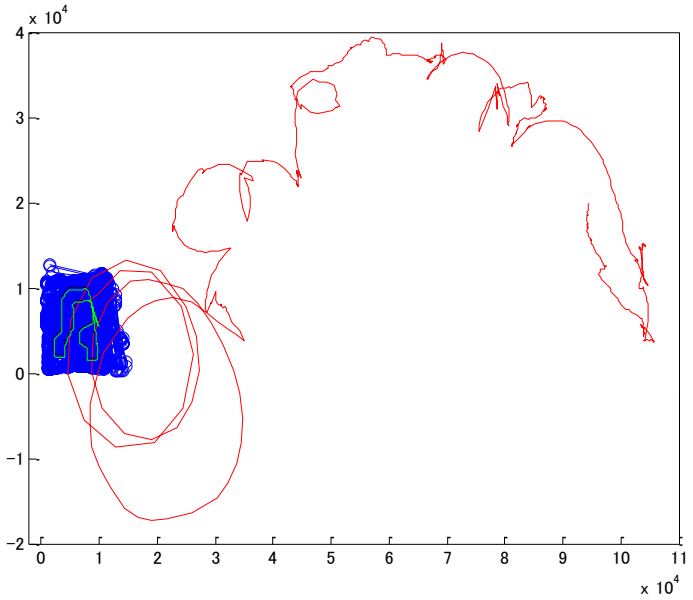


図 6.10　従来の ICL-SLAM による自己位置推定結果

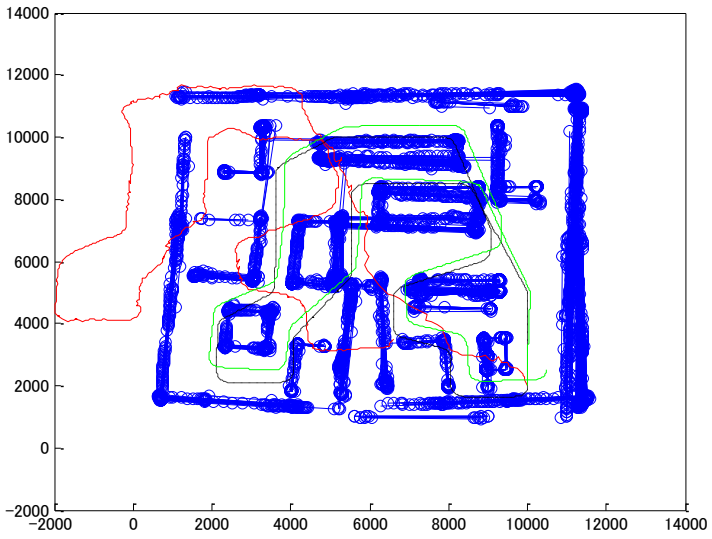


図 6.11　提案手法の ICL-SLAM による自己位置推定結果

図 6.10 のように，従来法ではシミュレーションでも自己位置推定の結果が大きくずれている．図 6.11 の提案法では最初に大きくずれてしまっているが，その後は走行軌道の形と合うように自己位置推定ができている．

以上の結果から，提案手法による ICL-SLAM では地図作成，自己位置推定のどちらも従来の ICL-SLAM に比べ精度が向上したことが確認できた．

第 7 章

結　言

本研究は ICL-SLAM の地図作成と自己位置推定の精度を向上させるために，これまで直線の端点情報で行っていた ICP アルゴリズムを直線の方向ベクトルを用いて行うように修正し，これを ICL アルゴリズムとした．SLAM とは自律移動ロボットの自己位置推定と環境地図生成の同時処理問題を解決する手法である．今回，対象とした ICL-SLAM の基となった ICP-SLAM の利点はロボットのモデル化を必要としないために計算が複雑ではないことと，モデル化されていない想定外の運動に対しても脆弱ではないことである．しかし，LRF から得られる膨大な点群により計算に時間がかかるということと，センサのノイズの影響を受けやすいという欠点がある．これを点ではなく直線を用いることで欠点を解消したものが ICL-SLAM である．しかし，ICL-SLAM では ICP-SLAM で用いられている ICP アルゴリズムをそのまま使用している，本来は点群に対して使用するべき ICP アルゴリズムを直線の端点情報という少ない点数で実行していたために，同じ障害物を表す直線でも異なる長さとなる場合に対応できずに精度が低下してしまっていた．そこで，直線の長さの関係ない方向ベクトルを利用するように ICP アルゴリズム修正することでこの問題を解決することを提案した．この提案手法を用いることで ICL-SLAM の地図作成及び自己位置推定の精度向上を目的としていた．ゆえに，提案手法を用いたオフライン実機実験とシミュレーション実験では，方向ベクトルを用いることで正しく ICL アルゴリズムを実行することで地図作成及び自己位置推定の精度の向上を確認した．

ただ，自己位置推定の精度はまだ低く，現在の状態で使用することは難しいと考えられる．よって今後の課題としては，自己位置推定のさらなる向上が必要と考えられる．ICL アルゴリズムでは移動パラメータ$(\boldsymbol{R},\boldsymbol{t})$のうち，回転量$\boldsymbol{R}$の導出では方向ベクトルを利用することで長さの異なる直線に対応したが，平行移動量$\boldsymbol{t}$の導出では従来のまま直線の端点情報を使用している．このため，同じ障害物を表す直線でも長さが異なる場合について完全に対応できているとは言えず，この部分により精度が低下してしまっている点があると考えられる．よって，平行移動量$\boldsymbol{t}$の導出でも直線の長さの影響を受けない方法に変更する必要があるといえる．また，現在の ICL-SLAM において，抽出した直線をそのまま前の地図に書き足しているが，このままでは周期が増えるごとにデータ数が増加してしまう．よって，データ数を減らすために前回までの周期で作成した地図の直線とその周期で抽出された直線が同じ直線と判断された場合はどちらかに統一するか，間を取った線を生成するなどした方が良い．また，現在は直線の品質を直線の限定・修正のみに使用しているが，品質の高い線に合わせてマッチングを行うといった工夫も可能である．

参考文献

[1] Gregor Michalicek, Prof. Dr. Jianwei Zhang; Dr. Werner Hansmann, “Development of a 3D Simultaneous Localization and Mapping Exploration System”, Diplomarbeit, Oktober, (2010)

[2] 大原, 駒田, 平井 “室内用自律移動ロボットのための ICL-SLAM” 三重大学修士論文, 2012

[3] Garulli,A.; Giannitrapani,A.; Rossi, A.; Vicino, A.; “Mobile robot SLAM for line-based environment representation” Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC’05. 44th IEE Conference p2041-2046, (2005)

[4] 倉爪 亮 :「協調ポジショニングシステムに関する研究」東京工業大学学位論文, (1998)

[5] 高田 健吾 :「2 平面の距離計測に基づく 3 次元環境モデル生成」大阪大学修士学位論文, (2008)

[6] Chen, Y. and Medioni, G.: ”Object modeling by registration of multiple range images”, Proceedings of IEEE International Conference on Robotics and Automation, pp. 2724-2729, (1991)

[7] Besl, P. J. and McKay, N. D.: “A Method for Registration of 3-D Shapes”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, pp. 239-256, (1992)

[8] Rusinkiewicz, S. and Levoy, M.: “Efficient Variants of the ICP Algorithm”, Proceedings of International Conference on 3-D Digital Imaging and Modeling, pp. 145-152, (2001)

[9] G.A.Borgers.; J.Aldon.; “Line Extraction in 2D Range Images for Mobile Robotics” Journal of Intelligent and Robotic Systems 40:267-297, (2004)

[10] Viet,N.; Stefan,G.; Agostino,M.; Nicola,T.; Roland,S.; ”A comparison of line extraction algorithms using 2D range data for indoor mobile robotics” Auton Robot 23:97-111, (2007)

[11] D. Sack; W. Burgard “A comparison of methods for line extraction from range data”, In Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV), (2004)

[12] Alshawa M. “ ICL: iterative closest line - a novel point cloud registrationalgorithm based on linear features”, Ekscentar, no. 10, pp. 53-59, (2007)

謝辞

本研究の遂行および本論文の作成にあたり，熱心なご指導とご意見を賜りました三重大学教授　平井淳之先生，同大学准教授　駒田諭先生，同大学准教授　弓場井一裕先生に深く感謝致します．ならびに研究の技術的な支援として大変お世話になりました同大学技術職員　中村勝氏に深く感謝致します．

研究活動を切磋琢磨した電機システム研究室の皆様には深く感謝しています．研究活動としては，自律移動ロボットグループの山口敦由氏，大原一真氏，黒田優希君には非常にお世話になりました．また，ロボットグループの中で特に近い位置にいた橋本賢人君には研究中に多くの意見を頂き，感謝しています．ならびに共に大学院を卒業する石﨑将崇君，荻田拓君，柏木宏康君，西口佳孝君，宮嶋啓伍君，山本和幸君には，学生生活を有意義なものにしてくれたことを深く感謝しています．

最後に，不自由なく学生生活を送らせてくれた家族に感謝します．

論文目録

[1] M.Terada, S.Komada, D.Yashiro, J.Hirai: “Research for improvement of map accuracy for ICL-SLAM using the line quality” The 2nd International Symposium for Sustainability by Engineering at MIU, AO-13

[2] 寺田，駒田，矢代，平井：「ICL-SLAM における直線の品質を用いた地図精度の向上に関する研究」平成 24 年度　三重地区計測自動制御研究講演会，RC-6（2012.12）