2007年度修士論文

センサネットワークにおける SLAMを用いたノード位置推定実験

早稲田大学大学院理工学研究科
情報・ネットワーク専攻

山田　寿夫

学籍番号　3606U110-2
提出　2007年 2月 4日
指導　甲藤 二郎 教授

目次

第1章　序論

1.1. はじめに

近年，無線通信技術や半導体技術の発展に伴い，携帯電話や PDA など無線通信デバイスの小型化，省電力化が進んでいる．また同様に，実世界からの様々な情報を収集するセンサーチップの小型化や廉価化に伴い，「いつでも，どこでも，何にでも，誰でも，ネットワークにつながることができる」というユビキタス社会においてセンサ技術の可能性は非常に有望視されている．そしてそのセンサネットワーク技術において電源問題と並んで大きな問題の一つとされているのが位置推定技術である．

1.2. 無線センサネットワークとは

センサネットワークとは，一般的な意味で，無数のセンサ群によって構成されるネットワークを意味する．有線でネットワークに接続した多数のカメラ群 [1]や，身体に身に着けて生体情報を取得するためのセンサネットワーク[2] [3] まで,「通信機能を持つセンサーデバイスによって構成されるネットワーク」すべてが，センサネットワークと呼ばれる．

このような多様性を持つセンサネットワークのうち，現在，特に強い注目を集めているものが，無線機能を持つ多数のセンサノードから構成される「無線センサネットワーク」である．センサノードとは，一つの筐体(もしくは基板上）に無線通信機能，センサ機能，電源ユニット，そして計算機を詰め込んだ小型のデバイスを指す．このようなセンサノードは，近接するセンサノードと互いに通信しながら，自律的にネットワークを構成する．このようにして構成された無線通信ネットワークを無線センサネットワーク（Wireless Sensor Network， WSN）と呼ぶ．各センサが取得したセンサ情報は，各センサノードのマルチホップ（多段中継）機能を用いて，シンクノードへと中継される．シンクノードとは，インターネットなどセンサノード以外のネットワークへのゲートウェイ機能を持つセンサノードである．このシンクノードを通じて，各センサが取得したセンサ情報は外部ネットワークへ公開される．現在，行われているセンサネットワークに関する研究の多くは，このようなマルチホップの無線センサネットワークを対象とするものである．

設置の自由度や応用範囲が広い無線センサネットワークでは，環境汚染や農作物の生育状況の監視や，物流管理，工場での生産管理，医療健康，防犯・防災技術，災害時の被災状況の把握など，多種多様な分野への応用が考えられている．また，来るべきユビキタス社会を構成するあらゆるサービスの実現のためにもセンサネットワークが注目されている．

1.3. コンテキストアウェア・サービス

来るべきユビキタス社会のなかでは，ネットワーク上を単純ながらも膨大な量の情報が存在していると考えられる．偏在するそれらの情報はあらゆる端末や機能を用い，収集される．そして，それらは最終的にアドレス解決や追跡制御・管理などを実現するネットワーク(NW) ミドルウェアにより，ユーザにとって意味のある，そして価値のある情報に変換される．そうした意味のある情報は，コンテキストと呼ばれ，大きく以下の三つに分類される．

1. 属性：その　モノ・人・空間　は何なのか？
2. 状態：その　モノ・人・空間　はどのような状態か？
3. 位置：どこに　そのモノ・人・空間　はあるのか？

そうした情報を利用して，ユーザに提供されるサービスはコンテキストアウェアなサービスといい，これはユビキタス社会において，非常に重要な概念の一つである．コンテキストアウェアなサービスを実現するためには，ユーザの環境情報(ユーザの位置，行動，体

調など）が非常に重要である．なかでもユーザ，つまり端末の位置情報は非常に有益で，そこから得られた情報からユーザのおおまかな行動が推測できるだけでなく，過去の時間軸との相関関係から，ユーザの行動傾向まで推測することも可能でもある．たとえば，ショッピングの際を考えてみる．位置情報として，ユーザの現在位置が分かる場合，プロファイル情報と連携することで，自分の嗜好にあった最も近い店が推薦されるサービスなどが考えられる．また，そうして得られた目的の店の位置情報が分かる場合，現在のユーザから目的の店までの，最短経路を指示してくれるサービスも考えられる．実際に，GPS を利用した位置取得技術により，目的の店への経路を指示してくれるサービスはあるが，目的の店の位置情報の多くは自動で取得することはできず，ユーザの手動により，そういったコンテキスト情報を取得する必要がある．また，店に到着し，ショッピングの最中，家やオフィスにいる人・動物や家やオフィスにある物・物品の状況などを把握したい場合もあるだろう．そうしたときには，コンテキストとして，家にある人や物の位置情報はコンテキストアウェアなサービスを提供するうえで，大きな情報の一つになるだろう．

このように，位置情報はコンテキストアウェアなサービス実現には欠くことのできない重要な情報であると考えることができる．

1.4. 位置検出技術

センサネットワークでは，情報の発生源となるセンサは必ず実空間に配置される．この時センサが「どこの情報を計測しているのか」という情報が必須となる．つまりセンサネットワークでは，単に温度や湿度といった情報だけでなく，地理的にそのセンサがどこにあるのかを表わす位置情報が必要となる．センサネットワークを構成する各デバイスに位置情報を提供する手段としてまず思いつくのは，GPS(Global Positioning System）の利用であろう．

GPS は高度 2 万 km の衛星軌道上にある NAVSTER(NAVigation Satellite Timing And Ranging) 衛星，通称 GPS 衛星を利用する．GPS 受信端末は，複数の GPS 衛星からの電波を受信し，それぞれの距離を割り出すことにより，現在位置を推測する．この GPS 受信端末を利用すれば，10m 程度の測位は非常に簡単に行える．

しかしながらセンサネットワークはコストや電源事情の問題から，全てのセンサネットワーク用デバイスに GPS 受信端末を搭載することは現実的ではない．加えて，GPS の利用可能な場所は主として屋外に限られる．さらには，都市部など電波伝搬環境が悪い場所では常に期待した精度で測位が行えるとも限らない．また 10m で測位可能であるとはいえ，この測位精度が全てのアプリケーションにとって十分なものであるとも考えにくい．このため，センサネットワークの各デバイスにどのようにして位置情報を与えたらよいかという問題は，ローカライゼーション（位置推定）の問題としてセンサネットワークの研究分野で重要な技術課題となっている．

無線センサネットワークにおいてローカライゼーション問題に対する研究は，センサノードが**受動的**であることを前提とした研究例が多い．受動的な位置情報推定手法として，大きく分けて以下の二つの手法がある．

- Range-based 方式
- Range-free 方式

GPS に利用されている TOA [4]や AOA [5]，TDOA [6]，などの手法に代表される，Range-based 方式は，超音波や磁気などの特殊な情報を利用し，ターゲットまでの距離を推定する手法で，特殊な情報を取得する為のコストがかかるが，少ない位置の既知なビーコンセンサー端末で実現することができる．例えば TDOA（図 1.4.1 参照）では，伝送速度の異なる 2 種類の伝送媒体から Beacon を送信し，到着時間の差と，伝送速度の差から距離を計算する．送受信で時刻の同期は必要ないが，音波は天候に左右されやすく見晴らしの良い環境が必要となる．

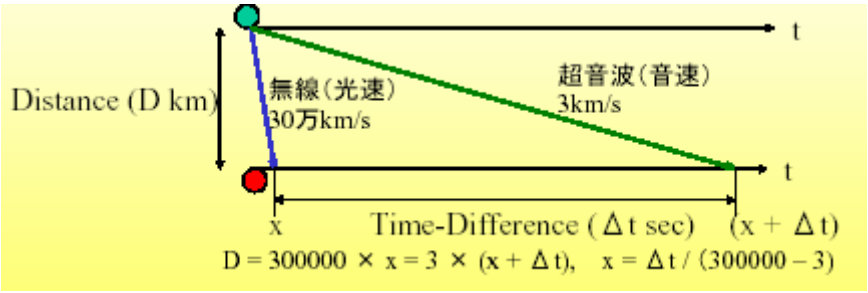


図 1.4.1 TDOA 測定

一方，Centroid [7], DV-HOP [8], APIT [9]などの手法に代表される，Range-free 方式は，特殊なハードウェアを必要としない反面，多数のビーコン端末を必要とする．例えばAPIT 測定では，ランドマークが位置情報を含んだビーコンを定期的に送信する．ランドマークからのビーコンは，広範囲の多くのノードに届く環境を理想としている．各ノードは，受信したビーコンから３台のランドマークの組み合わせで作成可能な三角形を導き出す．この三角形全てに対して，自分が内側にあるか外側にあるかの検証を行い，自分の位置を絞り込んでいく方式である．ランドマークが多ければ多いほど三角形を構成できるため，精度の高い位置推定が可能となる．

しかしながら，これらの位置推定手法は，位置評価を頻繁に更新することにより，移動対応させることは可能ではあるが，位置推定対象の可動性を考慮した十分な設計が行なわれているとは言い難い．そして，特別なインフラ設備が必要となるため，個人ユースでの利用が困難であることが課題として挙げられる．

1.5. ユビキタス・ロボティクス

従来のセンサネットワークにおいては，センサノードを「固定」したものが主流であったが，近年，廉価ロボットの登場やネットワーク・ロボティクス技術の発展によって，センサノードを「移動」させ，能動的な情報収集・処理を行う，ユビキタス・ロボティクス技術に関心が向けられている．移動ロボットを用いることは，モニタリングシステムなどを発展させるだけでなく，コンテキストアウェアなサービスと連携することで，より柔軟でユビキタスなサービスの発展が臨める．

1.6. 本研究の目的

本研究においては，プラットフォームとして市販のセンサノードと移動ロボットを利用し，特別なインフラ設備を必要としない位置推定システムを実験的に検証する．ノード位置推定には，受信電波強度（RSSI 値）と距離との関係に着目し，その関係性をSLAM(Simultaneous localization and mapping)アルゴリズムに利用することで位置推定を実現する．この RSSI 値は無線通信時の副産物として得られるものなので，コストや消費電力を抑えることができる．また，移動ロボットを移動センサノードとして用いることで多数の固定ノードを必要とすることなく位置座標推定を行うことができ，特別なインフラ設備を必要とすることなく，個人ユースで利用を可能にする．

1.7. 本論文の構成

以下に本章以降の構成を示す．

第２章　研究背景として各種使用端末と位置推定アルゴリズムについて説明する．
第３章　提案システムの概要と詳細を述べる．
第４章　提案システムの実装実験による評価と検討を行う．
第５章　本研究の総括と今後の課題について述べる．

第2章　研究背景

2.1. 無線センサ端末 MOTE

2.1.1. MOTE について

センサノードは，マイクロプロセッサ，データストレージ，センサ，AD コンバータ，データトランシーバ，コントローラ，および電源から構成される．近年，このようなセンサノードのうち，小型で安価な Mote （微塵）と呼ばれるデバイスが各社から商品化されている．Mote は，カリフォルニア大学バークレー校の Smart Dust プロジェクトのゴールとして掲げられた，数ミリメートルの大きさのセンサノードを Mote と呼んだことに由来する呼称である．

MOTE とは UC Berkeley の NEST プロジェクト[10] が開発したセンサネットワーク構築用のセンサノードである. Mica MOTE は最も早く市販されたセンサーネットワークプラットフォームである. MOTE には, 無線モジュールに ZigBee (Chipcon 社の CC2420) を使用した Micaz MOTE (図 2.1.1 参照) がある. Micaz MOTE の製品仕様は表 2-1 に示す. これは，Atmel 社の Atmega128 というマイコンを搭載している．また，TinyOS 上で動作し，図 2.1.2 に示す PC インタフェース基板(MIB510) を用いて，プログラミングを自由に書き込むことができる．MIB510 は，システム動作中に基地局としても動作し，Micaz MOTE などの通信ノードと無線でデータをやり取りし，測定データの取り込みや，命令の送信などを行う．さらに，PC とはシリアル通信(RS232C) を介してデータや命令を送受信できる．これらは，それぞれセンサ基盤を接続することでいくつかの環境センシングを行うことができる．現在，センサとして温度，光，音，磁気，加速度などがある．

図 2.1.1　Micaz MOTE

図 2.1.2　MIB510 PC インタフェース基板

表 2-1　Micaz MOTE の製品仕様

製品名		Micaz MOTE
プロセッサ	CPU Speed	7.4MHz(省電力型)
	メモリ (プログラム領域)	128KB (Flash)
	メモリ (データ領域)	512KB (Flash)
	AD Converter	10 bit
無線	無線モジュール	Chipcon CC2420
	無線周波数	2400MHz Zigbee
	データレート	250 Kbaud
電流	電流消費(受信時)	40mA (3Vdcの場合)
	電流消費(Sleep時)	＜30μA (3Vdcの場合)
外形寸法		64×35×27mm
ソフトウェア		TinyOS (言語：NesC) オープンソース開発プラットフォーム
外部電源		2.7-3.3V (DC ：単3乾電池2本相当)

2.1.2. **オペレーティングシステム：**TinyOS

センサネットワーク研究のプログラム開発の多くは，センサノードの API を直接制御するプログラムの作成によって行われていた．このような現状を打開した，最初のオペレーティングシステムが TinyOS[9]である．そこで，ここでは，センサノードのオペレーティングシステムの代表例として，この TinyOS について述べる．TinyOS は，カリフォルニア大学バークレー校で開発された， Mote のハードウェアを制御するためのオペレーティングシステムであり，省電力と物理世界とのインタラクションの 2 点に焦点を絞って設計されている．TinyOS は，デバイスの機能にアクセスするための非常に軽量化されたコンポーネントを提供し，このコンポーネントライブラリはネットワークプロトコル，分配サービス，センサードライバ，そしてデータ収集ツールを含んでおり，TinyOS のイベントドリブン実行モデルは，緻密なパワーマネージメントを可能する．さらにユーザは，これらのコンポーネント群の中から，アプリケーションが必要とする最小限のコンポーネントだけを選択する．そして，それらをイベントとシグナル処理によって結合することによって，目的とするアプリケーション（コンポーネントのスケジューリングとグラフ）を限られた物理メモリ上に構築することができる．コンポーネントは並列実行が可能であり，イベントの待機中に計算資源を消費しない（ Blocking と Polling を行わない）ように設計されている．コンポーネントは「通信」や「センサ読み出し」などに階層化されている．

TinyOS は，Mica Mote などのセンサ端末の動作制御アプリケーションやユーザ・インタフェースアプリケーション開発のためのライブラリとツールが含まれている．また，TinyOS は，オープンソースで公開［15］されており，開発環境としては C 言語の拡張である NesC [16] [17] が提供されている．

2.1.3. **プログラミング言語：**NesC

TinyOS システムは，ライブラリおよびアプリケーションの基本概念を導入する．このシステムは，NesC というプログラム言語で書かれている．nesC 言語は，センサネットワークのような埋め込まれたシステムのために主として意図され，NesC は C＋のシンタックスを持っているが，強健なネットワークに埋め込まれたシステムへソフトウェア・コンポーネントを共に組み立てて，指定し，リンクするためのメカニズムと同様に TinyOS 一致モデルをサポートする．また，潜在的なバグの多くの源を除去し，アプリケーション・デザイナーが容易に設計できるコンポーネントの構築を可能にしてくれる．しかし広範囲なチェックを行なうので，時間(TIME)をコンパイルする．また，TinyOS は，NesC の中で表現される多くの重要な概念を定義している．最初に，NesC アプリケーションは，十分に定義された双方向インタフェースを備えたコンポーネントから構築され，次に，nesC はタスクおよびハードウェア・イベントハンドラーに基づいて，一致モデルを定義し，TIME をコンパイルしてデータ・レースを検知する．NesC はイベント駆動，並列実行，コンポーネント指向という特徴を持ち，TinyOS のコンポーネントとイベント処理に対応している．

2.1.4. Stargate

Stargate は Intel が開発し，Crossbow 社が販売を行うゲートウェイクラスのプラットフォームである. 32bit 400MHz X-scale プロセッサを搭載. OS は組み込み用 Linux kernel を用いる．このプラットフォームには 64MB SDRAM も搭載しているため，処理速度に問題が発生する場合は少ないと考えられるが，内蔵 Flash メモリを 32MB 分しか積んでいないため，インストールするソフトウェアやプログラムには注意をする必要がある．また，USB ポートが付属してあるため, USB-WebCam などを接続することで画像ストリーミングも行うことができる．また，PC カードスロットも搭載してあるため，CF 無線 LAN カードなどを用いて，インターネットとの接続性も確保できる．一方で mica2 などと接続することで mote センサネットワークとインターネットとのゲートウェイとしても機能する．

図 2.1.3　Stargate

表 2-2　Stargate の製品仕様

製品名	stargate
プロセッサ	2bit, 400MHz Intel PXA255 XScale
メモリ	32MBフラッシュメモリ / 64MB SDRAM
通信	802.11b

2.2. 移動ロボット端末

2.2.1. LEGO Mindstorms

Mindstorms とは，ブロックで有名なデンマークの LEGO 社の LEGO ブロックを，プログラミングを用いて作るロボットのキットのことで, 1998年9月にRIS(Robotics Invention System)Ver1.0 と呼ばれる Mindstorms が登場した．そして，1999 年 9 月には，姉妹製品として，RDS(Robotics Discovery Set)や DDK(Droid Development Kit)が発売され，2000 年 7 月には，DDK の姉妹製品で，DSDK(Dark Side Development Kit)が発売された．また, RIS は, 1999 年 10 月に Ver1.0 から Ver1.5 に, 2000 年 11 月に Ver2.0 となった. RIS, RDS，DDK それぞれの頭脳(センサー) ()は，RCX，SCOUT，MICRO SCOUT となっている．LEGO Mindstorms は 12 歳以上の児童向けになっているだけあり，容易にロボットの組み立てからプログラミングまでを行うことができる．

図 2.2.1　LEGO Mindstroms

2.2.2. Mindstorms NXT

今回の実装実験において使用したものは新たな MindStorms として 2006 年 1 月に世界で発表され，日本では 2006 年 10 月から発売になった LEGO MindStorms NXT である．組み立てとプログラムは 30 分程度で可能で，これまで市販バージョンのプログラミングは Windwos のみで可能だったが，今回は初めて Mac にも対応し，National Instruments の labVIEW を使った高度なプログラミングが可能．PC からのプログラム転送は，USB2.0,Bluetooth で可能となり，Bluetooth 搭載の携帯電話や PDA からもコントロールできるようになった．また，インタラクティブなサーボモータが内臓されており，そこに回転センサ (Rotation Sensor)が搭載されているため，Mindstorms を移動ロボットとしての移動制御することが出来る．また超音波センサにより動きに対応した動作が行え，サウンドセンサーでサウンドパターンやトーンを認識できるほか，色や照度に対応するライト

センサー，タッチセンサーも搭載している．LEGO ブロックとしては LEGO TECHNIC の 519 ブロックが用意されている．上記以外のセンサとして，HiTechnic 社では，NXT Gyro，NXT IRLink，NXT IRSeeker，NXT Compass Sensor，NXT Color Sensor，NXT Acceleration / Tilt Sensor などの商品開発がなされている．本研究での提案ではこの LEGO Mindstorms NXT を用いて行う．以下に，LEGO Mindstorms NXT の頭脳である NXT ブロックの技術仕様を示す．

- 32 bit ARM7 マイコン@48MHz(メイン)
- 256 KB フラッシュメモリ
- 64 KB RAM
- 8 bit Atmel AVR マイコン@4MHz
- 4 KB フラッシュメモリ
- 512 Bytes RAM
- 64 × 100 pixel LCD ディスプレイ
- Bluetooth 通信(Bluetooth Class II V2.0 compliant)
- USB 2.0 port
- 4 つの入力ポート
- 3 つの出力ポート
- スピーカー(8kHz)
- 6 AA バッテリ

図 2.2.2 NXT ブロック

図 2.2.3 Mindstroms NXT の組み立てサンプル

2.2.3. leJOS

leJOS は Sun Microsystems からリリースされている Java 2 SDK と比較をすると，32k バイトの RAM を積んだコンピュータ(RCX ブロック）で実行できる，Java micro-micro edition といったもの．いわゆる Java 同様 JVM(Java Virtual Machine)を用いて，javac コンパイラーでバイトコード変換する．leJOS には，プログラミング支援のための標準 Java API の縮小バージョンが入っている．leJOS は，浮動少数点数の利用やマルチスレッドへの対応，ループ・配列の利用やイベント・モデルの利用(TimerListener, ButtonListener, SensorListener）・例外処理などおおよそ基本的なロボットプログラミングに必要なあらゆるメソッドを有している．

2.2.4. icommand

leJOS の開発メンバーが Bluetooth 経由で遠隔操作をすることを目的にリリースした Java API であり，Bluetooth から経由で PC から「コマンド」と呼ばれるものを送信することで，LEGO MindStorm NXT を遠隔操作するもの．よって leJOS のようにマイコンにプログラムをダウンロードするという用途には使用することができない．

本研究においてはバージョン 0.5 を使用している．

2.3. 確率ロボティクス

確率ロボティクスの確信は，センサデータから状態推定するという考え方である．状態推定とは，推論できるが直接観測できない値をセンサデータから推定する問題のことである．ほとんどのロボット技術の応用では，何をすべきか決定することは，確かな値さえ分かれば比較的簡単である．例えば，移動ロボットの移動は，そのロボットの正確な位置と，周囲の障害物が分かれば比較的簡単である．しかし，それらの変数は直接推定できない．その代わり，ロボットはセンサを頼りにその情報を収集する．各センサは，それらの変数に対して，部分的な情報しか与えてくれない．そして，センサによる計測は雑音によって乱れてしまう．状態推定はそのような乱れたデータから状態変数を正しく得ることである．確率的状態推定アルゴリズムでは，環境中であり得る状態全体に対して，ロボットの信念確率分布が計算される．確率状態推定の例は，「移動ロボットの位置推定問題」となる．

2.3.1. 確率の概念

確率ロボティクスでは，センサ計測値や制御やロボット環境の状態のような量は，全て確率変数としてモデル化される．確率変数は多値をとることができ，ある特定の確率法則に従った場合，多値をとる．確率的推論は，他の確率変数や観測されたデータから展開される，確率法則を用いた確率変数の計算過程のことである．

X を一つの確率変数，x を，X ではないかと考えられるある特定の値とする．もし X がとれる全ての値の空間が離散的であれば，

$p(X = x)$

というように，確率変数 X が値 x をとる確率を表現する．

本稿では，良く用いられる略記 $p(x)$ を $p(X = x)$ の代わりに用いる．本稿で説明する手法は，連続空間での推定や行動決定についてのものである．連続空間は，連続値を取れるいくつかの確率変数で特徴付けられる．全ての連続的な確率変数には**確率密度関数**(probability density functions, PDF)が定義できると仮定する．平均μ，分散σ^2 の一次元正規分布は，よく知られた密度関数で，正規分布の PDF は，次のような**ガウス関数**：

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\} \quad (2.3.1)$$

で与えられる．本稿では，正規分布を $\mathrm{N}(x;\mu,\sigma^2)$ と略記する．ここで，x，μ，σ^2 はそれぞれ確率変数の値とその平均と分散を表す．

正規分布の式()では x がスカラ値であると仮定されているが，多次元ベクトルの場合は，多変量正規分布と呼ばれる．多変量正規分布は，次式の密度関数：

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\} \quad (2.3.2)$$

で表される．ここでμは平均ベクトル，Σは半正定値対称行列であり，共分散行列と呼ばれる．T は転置を表す．この PDF の指数部は x の二次式であり，μとΣがこの二次関数のパラメータである．式(2.3.1)と式(2.3.3)は，x がスカラで $\Sigma = \sigma^2$ ならば等価である．

2.3.2. ロボットと環境の相互作用

2.3.2.1. 状態

環境は，状態で特徴付けられる．あるいくつかの状態変数は，ロボットの近くにいる人々の位置のように，時間経過とともに変化する性質を持つ．他の状態変数は，建物の壁のように静的である．変化する状態は動的状態と呼ばれ，静的状態や不変状態とは区別される．姿勢，速度，センサが正しく動作しているかどうかなど，ロボット自身に関する変数も状態に含まれる．本稿では，状態は x で表される．x に含まれる状

態変数は問題に応じて変化する．時刻tの状態はx_tと表記される．状態には以下のようなものがある．

・ ロボットの姿勢

これは，グローバル座標系に対するロボットの位置に対するロボットの位置，向きで構成される．剛体の移動ロボットはそのような変数を 6 個持ち，そのうち 3 個はデカルト座標系のもので，残り 3 個は角度方向のもの(ロール，ピッチ，ヨー角)である．平面上を移動する剛体の移動ロボットにとっては,姿勢は通常 3 個の変数で表させる．そのうち 2 個は平面上の位置，残りは向いている(ヨー角)である．

・ マニピュレータの場合

マニピュレータの場合，姿勢はアクチュエータのコンフィギュレーションに関する変数を含む．ロボットアームの各自由度は，ある時刻での一次元コンフィギュレーションで特徴付けられる．コンフィギュレーションの各次元は，ロボットの運動状態を表すための不可欠な要素である．マニピュレータのコンフィギュレーションは，運動学的状態と呼ばれる．

・ ロボットの速度と関節の速度

ロボットの速度と関節の速度は，一般的に力学的状態と言われる．空間を移動する剛体のロボットは，6 個の速度変数で特徴付けられる．それらは，各姿勢の変数と関連している．

・ 環境中の物体の位置と特徴

環境中の物体の位置と特徴も，状態変数である．物体は，樹木であったり，壁であったり，ある壁面，床面に記された点かもしれない．その物体は，それらの外観(色や模様)で特徴付けられる．本稿で扱われる問題では，環境中の物体(センサ)の位置は静的である．ロボティクスでは，これらの対象物はナビゲーションに用いられるため，**ランドマーク**と呼ばれる．ランドマークになり得る物体は，環境中で目立つ不変の特徴を持ち，確実に認識できるものである．

・ 移動物体と人の位置や速度

移動物体と人の位置や速度も，状態変数となり得る．環境中で動くものがロボットだけということはない．ロボットの他の移動体は，それぞれに運動学的，力学的状態を持つ．

・ ロボットの動作への影響

ロボットの動作に影響を与える状態変数は，他にも数多く存在する．例えば，センサが壊れているかどうかも状態変数になり得る．電池で動くロボットにとっては，電池の残量も状態変数になる．

状態x_tが将来を予測するために最善のものであるとき，その状態は**完備**であると言われる．完備性とは，その状態ベクトルに，過去の状態や計測値，制御といった変数を加えても，将来を予測するために有益は情報が増えないことである．次のような条件「将来の状態は確率的に遷移する，しかし，将来に対して影響を与える変数がx_tの他にない，あったとしても状態x_tに従属している」を満たす時間過程は**マルコフ連鎖**として知られている．

状態の完備性の概念は，主に理論上において重要である．実際には，どのようなロボットシステムでも完備状態を特定することは不可能である．現実的な実装では，全ての状態変数から上記で述べたような状態変数を少し抜き出して状態を構成することになる．このような状態は，不完備状態と呼ばれる．

2.3.2.2. 環境との相互作用

ロボットとその周囲の環境との相互作用には，基本的な二種類のタイプがある．一方はロボットがアクチュエータによって環境の状態に影響を与えることであり，もう

一方はセンサを通じて状態に関する情報を収集することである．この両者は同時に起こる．

・ 環境のセンサ計測

認識は，ロボットがセンサを用いて環境の状態に関する情報を獲得する過程のことである．例えば，ロボットは環境に関する情報を得るために，カメラで画像を撮ったり，測域センサを利用したり，触覚センサを利用する場合がある．そのような認識に関する相互作用の結果一つ一つは，計測値と呼ばれる．または，観測や知覚と呼んだりもする．センサの計測値は幾分かの遅れを伴って得られる．したがって，計測値は少し前の時刻の状態に関する情報を与える．

・ 制御動作

制御動作は世界の状態を変化させる．ロボットの環境に能動的に力を加えることで，そのような変化が起こる．制御動作の例には，ロボットの移動や物体のマニピュレーションが含まれる．状態はたいていの場合，ロボットが何を行動しなくても変化する．したがって，ロボットは常に制御動作を実行していると仮定できる．実際，電源の入っているロボットは動かないでいる時も常に制御プログラムを実行しており，それと共に計測を行っている．

ロボットが過去の全てのセンサ計測値や制御動作を記録しておくことが出来ると仮定した場合，環境との相互作用が 2 種類あることから，二つの異なるデータが定義できる．

・ 環境計測データ

環境計測データは，各時刻の環境の状態に関する情報を与える．計測データの例としては，カメラ画像，測域センサの計測結果などが挙げられる．その他，無線機能の備わった端末の電波測位(受信電波強度や電波往復遅延時間)なども含まれる．本稿では小さな時間のずれを無視し，時刻tの計測データをz_tで表す．次式は，時刻t_1からt_2魔で煮えられた全ての計測値の集合を現す($t_1 \leqq t_2$).

$$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, ..., z_{t_2} \quad (2.3.4)$$

・ 制御データ

制御データは，環境の状態の変化に関する情報を与える．移動ロボティクスでは，ロボットの速度は制御データの典型例である．制御は状態の変化に関する情報を伝える．速度の代わりにオドメータも制御データの源となる．オドメータはロボットの車輪の回転量を計測するセンサであり，走行量(オドメトリ)として状態の変化に関する情報を与える．オドメータはセンサであるが，制御動作の影響を計測するので，オドメトリを制御データとして扱う．制御データはu_tで表される．変数u_tは時間$[t-1;t]$での状態の変化に対応する．計測データと同様，制御データは，

$$u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, ..., u_{t_2} \quad (2.3.5)$$

で表される($t_1 \leqq t_2$)．たとえロボットがなにをしなくても環境の状態は変化するので，厳密に言えば，時間が経過したということが制御データとなる．従って制御データは，時間ステップあたりに一つずつ存在し，その中には「なにもしない」という行動が含まれる．

2.3.2.3. 確率的発生法則

状態や計測値の発生は確率の法則に支配される．一般的に，状態x_tは状態x_{t-1}から確率的に発生する．状態x_tの発生という事象は過去の全ての状態や計測値，制御に従属しているように考えられる．ゆえに，状態の発生の背後にある確率法則は，

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t) \quad (2.3.6)$$

で表現される．この等式で表現される性質は，条件付き独立性の例である．条件付き独立性とは，いくつかの変数(条件付け変数)の値が分かっていると，特定のいくつかの変数が他の変数に対して独立となるという性質のことである．

さらに，計測で発生する過程をモデル化することも必要である． x_t が完備ならば，次の重要な条件付き独立性：

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t) \quad (2.3.7)$$

が成り立つ．この式は，(ばらつきのあり得る)計測値 z_t を予測するためには，x_t だけ分かれば良いということを意味している．過去の計測や制御や，過去の状態に関する知識は， x_t が完備であれば必要ない．

式()で得られた確率 $p(x_t \mid x_{t-1}, u_t)$ は，状態遷移確率と呼ばれる．状態遷移確率は，ロボットの制御 u_t によって，どのように環境の状態が時間発展するかを規定するものである．しばしば，状態遷移が時刻 t と独立していることがある．この場合，$p(x_t \mid x_{t-1}, u_t)$ を $p(x' \mid u, x)$ と記述する．ここで x' は事後， x は事前の状態である．

また，式()で得られた確率 $p(z_t \mid x_t)$ は計測確率と呼ばれる．計測確率も時刻 t に依存しないかもしれない．この場合，計測確率は $p(z \mid x)$ と記述できる．計測確率は，環境の状態 x からどの計測値 z が得られるかということに関する確率法則を示す．計測値は，状態を雑音混じりで映し出すものであると考えることが適当である．

状態遷移確率と計測確率は，一対となってロボットとその環境に対する確率力学系を表現できる．図 2.3.1 は，これらの確率で定義された状態と計測値の推移を表している．時刻 t の状態は時刻 $t-1$ と制御 u_t に，確率的に依存している．計測値 z_t は時刻 t の状態に確率的に依存している．このような時間発生モデルは隠れマルコフモデル(HMM)，あるいはダイナミックベイズネットワーク(DBN)と呼ばれる．

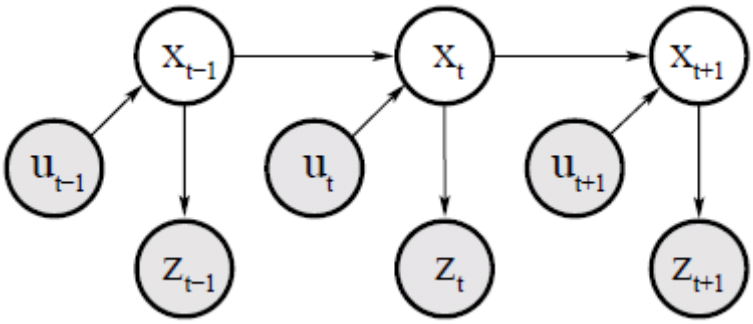


図 2.3.1　ダイナミックベイズネットワーク

2.3.2.4. 信念分布

確率ロボティクスでもう一つ鍵となる概念に，信念がある．信念は，環境の状態に関するロボットの内部知識を反映する．例えば，ロボットの姿勢が，あるグローバル座標系で $x_t = \langle 14.12, 12.7, 45° \rangle$ だったとしても，ロボットはそれを知ることはできない．なぜなら姿勢を直接計測する方法がないからである(GPS でもできない)．その代わり，ロボットは自身の姿勢をデータから推測しなければならない．従って状態に関する内部の信念から，実際の状態を区別しなければならない．信念の類義語として状態の知識や情報状態がある．

確率ロボティクスでは，条件付き確率分布によって信念が表現される．信念確率分布は，実際の状態に関して，あり得る全ての仮説に対して確率(あるいは密度値)を割りつける．信念確率分布は，得られたデータで条件付けられた事後確率分布であり，状態変数で構成される空間上で定義される．以後，ある状態の定義 x_t 上の信念を $bel(x_t)$ で表す．この表現は，

次の事後信念：

$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$

を略記したものである．この事後信念は，過去の全ての計測値 $z_{1:t}$ と制御 $u_{1:t}$ で条件づけられた，時刻 t における状態空間上の確率分布である．

ここで，信念が計測 z_t を反映した**後**のものとして暗に定義されたことに気づく．時折，制御 u_t を実行した直後，z_t を反映する**前**の事後信念を計算する場合がある．このような事後信念は，

$\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t})$ （2.3.8）

で表される．この確率分布は，確率的フィルタリングにおいて**予測**と呼ばれる．これは，時刻 t の計測が入る前に，$\overline{bel}(x_t)$ が以前の事後信念に基づいて時刻 t の状態を予測していることを反映している．$\overline{bel}(x_t)$ から $bel(x_t)$ を計算することは，**修正**，あるいは**計測更新**と呼ばれる．

2.3.3. カルマンフィルタ

2.3.3.1. カルマンフィルタのアルゴリズム

カルマンフィルタ(kalman filter, KF)は，ベイズフィルタ(図 2.3.2)の実装方法として最も多く研究された．カルマンフィルタは，Swerling と Kalman によって，線形ガウス型モデルにおけるフィルタリングや予測の手法として考案された[11] [12]．カルマンフィルタでは，連続状態に対して信念に関する計算が実装される．離散系やハイブリッドな状態空間には適用できない．

1: **Algorithm Bayes_filter**$(bel(x_{t-1}), u_t, z_t)$:
2: *For all* x_t *do*
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1},) bel(x_{t-1}) dx_{t-1}$
4: $bel(x_t) = \eta p(z_t \mid x_t) \overline{bel}(x_t)$
5: *endfor*
6: *return* $bel(x_t)$

図 2.3.2　ベイズフィルタの基本アルゴリズム

カルマンフィルタではベイズフィルタのマルコフ性に加えて，次の三つの条件が満たされると，事後信念はガウス分布となる．

1. 状態遷移確率 $p(x_t \mid u_t, x_{t-1})$ が，ガウス雑音を足した線形関数である必要がある．この条件は，次のような等式：

$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$ （2.3.9）

で表される．ここで x_t と x_{t-1} は時刻 t における状態ベクトルで，u_t は同時刻の制御ベクトルである．これらのベクトルは縦ベクトルで，

$$x_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{n,t} \end{pmatrix} \quad \text{そして} \quad u_t = \begin{pmatrix} u_{1,t} \\ u_{2,t} \\ \vdots \\ u_{m,t} \end{pmatrix} \quad (2.3.10)$$

と表現される． A_t と B_t は行列である． n が状態ベクトル x_t の次元であるとき， A_t は $n \times n$ の正方行列である．B_t は $n \times m$ の行列であり，ここで m は制御ベクトル u_t の次元である．状態と制御をそれぞれ A_t と B_t で乗算して足すことで，この状態遷移関数は線形となる．カルマンフィルタは，このような線形系を想定している．

式(2.3.1)中の確率変数 ε_t は，ガウス分布に従う乱数ベクトルであり，状態遷移の不確かさをモデル化するために足される． ε_t の次元は状態変数の次元に等しい．この乱数ベクトルの平均値はゼロベクトルで，共分散は R_t と記述される．式(2.3.1)で与えられる状態遷移確率は，線形な式にガウス雑音が加わっていることから，線形ガウス型であると呼ばれる．

事後状態の平均値は $A_t x_{t-1} + B_t u_t$, 共分散は R_t で与えられるので, 状態遷移確率は,

$$p(x_t \mid u_t, x_{t-1}) = \det(2\pi R_t)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\} \quad (2.3.11)$$

で得られる．

2. 計測確率 $p(z_t \mid x_t)$ も線形であり，雑音はガウス雑音で無ければならない．つまり，計測確率は,

$$z_t = C_t x_t + \delta_t \quad (2.3.12)$$

で定式化できなければならない．ここで C_t は $k \times n$ 行列である．k は計測ベクトル z_t の次元である．ベクトル δ_t は計測に関する雑音を表している． δ_t の分布は平均ゼロ，共分散 Q_t の多変量正規分布であるとする．計測確率は，次の多変量正規分布：

$$p(z_t \mid x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\right\} \quad (2.3.13)$$

で与えられる．

3. 初期信念 $bel(x_0)$ が正規分布でなければならない．この分布の平均値を μ_0，共分散を $\sum_0$ で表すとすると，この時,

$$bel(x_0) = p(x_0) = \det(2\pi \sum_0)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x_0 - \mu_0)^T \sum_0^{-1}(x_0 - \mu_0)\right\} \quad (2.3.14)$$

となる．

図 2.3.3 にカルマンフィルタのアルゴリズムを示す. カルマンフィルタでは, モーメントパラメータ化によって信念が表現される．時刻 t において，信念は平均 μ_t と共分散 $\sum_t$ で表される．カルマンフィルタの入力は時刻 $t-1$ の信念であり，その信念は平均値 μ_{t-1} と共分散 $\sum_{t-1}$ で表される．これらのパラメータを更新するために，入力として制御 u_t と計測 z_t も必要となる．カルマンフィルタの出力は μ_t と $\sum_t$ である．図 2.3.3 の 4 行目で計算される変数 K_t はカルマンゲインと呼ばれている．この変数は，計測を新たな状態推定にどの程度反映させるかを決定する．6 行目で用いられる I はイノベーションの概念を示す．イノベーションとは, 実際の計測値 z_t と, 期待される計測値である 5 行目の $C_t \overline{\mu}$ の差異のことである．

1: **Algorithm Kalman_filter**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:
2: $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
3: $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
4: $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
6: $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
7: *return* μ_t, Σ_t

図 2.3.3　カルマンフィルタのアルゴリズム

2.3.3.2. 拡張カルマンフィルタのアルゴリズム

計測が状態の線形関数で，状態が直前の状態の線形関数であることはカルマンフィルタに不可欠な前提である．ガウス分布に従う関数変数を線形変換するとガウス分布に従う確率変数になることは，カルマンフィルタアルゴリズムの導出において重要な役割を果たす．そして，カルマンフィルタの計算効率が良いのは，ガウス関数のパラメータのみで計算が行えるからである．

しかし，状態遷移や計測が線形であることは，現実には滅多に無いことである．例えば，ロボットが一定の速度，角速度で移動すると円形軌道になるが，この状態遷移は線形変換では表現できない．このことは，ロボティクスに関しては，ごく簡単な問題にしか線形なカルマンフィルタが適用できないことを示している．

拡張カルマンフィルタ(extended kalman filter, EKF)は，線形性の仮定を緩和する．ここで，状態遷移確率と計測確率がそれぞれ非線形関数 g と h に従うと仮定し，

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \quad (2.3.15)$$

$$z_t = h(x_t) + \delta_t \quad (2.3.16)$$

とモデル化する．このモデルは，カルマンフィルタの基礎をなす線形モデルの式(2.3.1)と式(2.3.4)を一般化したものである．式(2.3.1)の行列 A_t と B_t は関数 g に置き換えられ，式(2.3.4)は行列 C_t は h で置き換えられている．しかしながら， g と h は任意なので，信念はガウス分布には従わない．実際，非線形関数 g と h に対して信念を正確に更新することはたいてい不可能であり，ベイズフィルタは数式解を持たなくなる．

EKF では，推定対象の状態に対してガウス分布の近似が計算される．EKF を使用する目的は，複雑な分布形状の事後信念を厳密に計算しようとせず，平均と共分散を計算効率よく推定することである．数式としてきれいに解けない統計を扱うため，EKF では近似が必要になる．

EKF による近似の鍵となる考え方は，線形化である．非線形関数を線形化する手法は数多く存在する．EKF では(一次の)テイラー展開と呼ばれる手法が利用される．テイラー展開は関数 g の値と傾きから， g の線形近似を行う．傾きは，次の変微分

$$g'(u_t, x_{t-1}) := \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}} \quad (2.3.17)$$

で得られる．明らかに g 値と傾きは関数 g の変数煮の値に左右される．この値としては，線形化する時点における最も尤もらしい(最尤な)状態を選ぶことが正しい方法である．ガウス分布では，最尤な状態は事後信念の平均値 μ_{t-1} である．これらの値における勾配から g の

近似である線形外挿：

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{=:G_t}(x_{t-1} - \mu_{t-1}) \qquad (2.3.18)$$
$$= g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$$

が得られる．状態遷移確率は

$$p(x_t \mid u_t, x_{t-1}) \approx \det(2\pi R_t)^{-\frac{1}{2}} \exp\Big\{ -\frac{1}{2}\big[(x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})\big]^T \qquad (2.3.19)$$
$$R_t^{-1}\big[(x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})\big] \Big\}$$

のようにガウス関数で近似される G_t が $n \times n$ の行列であることを注意したい．ここで n は状態の次元を表す．この行列は**ヤコビ行列**と呼ばれる．ヤコビ行列の値は u_t と μ_{t-1} から決定される．そのため，時刻によってヤコビ行列は変化する．

EKF では，計測関数 h も全く同様に線形化される．h のテイラー展開は $\bar{\mu}_t$ 回りで行われる．$\bar{\mu}_t$ は，ロボットがそのときに考えている最尤な状態である．h は，

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{h'(\bar{\mu}_t)}_{=:H_t}(x_t - \bar{\mu}_t) = h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t) \quad (2.3.20)$$

で線形化される．ここで，$h'(x_t) = \dfrac{\partial h(x_t)}{\partial x_t}$ である．ガウス分布を示すと，

$$p(z_t \mid x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\Big\{ -\frac{1}{2}\big[(z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)\big]^T \qquad (2.3.21)$$
$$Q_t^{-1}\big[(z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)\big] \Big\}$$

となる．

図 2.3.4 に EKF のアルゴリズムを示す．

1: **Algorithm Extended_Kalman_filter**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
7: *return* μ_t, Σ_t

図 2.3.4　拡張カルマンフィルタのアルゴリズム

2.3.4. パーティクルフィルタ

パーティクルフィルタは，ノンパラメトリックなベイズフィルタの実装方法の一つである．ノンパラメトリックフィルタでは，例えばガウス分布のように事後信念の形状が固定されている必要が無く，代わりに有限個の数値で事後信念を近似する．これら有限個の数値は，状態空間中の各領域と対応付けられる．パーティクルフィルタは，事後信念から状態をランダムにサンプリングして事後信念 $bel(x_t)$ を表現する．指数関数でパラメトリックにガウス分布を表現する代わりに，パーティクルフィルタはこの分布から選ばれる標本の

集合で分布を表現する．この表現方法はあくまで近似的であるが，ノンパラメトリックなゆえにガウス分布などよりも広範囲の分布を表現できる．標本を用いる表現のもう一つの利点は，確率変数の非線形な変換を表現できることである．

パーティクルフィルタでは，事後確率分布の標本はパーティクルと呼ばれ，

$$X_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]} \quad (2.3.22)$$

と表現される．各パーティクル $x_t^{[m]} (1 \le m \le M)$ は，時刻 t における状態を具体的な事例にしたものである．換言すれば，一つ一つのパーティクルは時刻 t における真の状態に対する一つの仮説である．ここで M はパーティクルの集合 X_t 中のパーティクルの数である．実際にパーティクルフィルタが用いられるとき， M は $M = 1000$ というように大きな数になる．実際の適用例においては，t の関数や，$bel(x_t)$ の特徴量に対する関数などで M が決定されている．

パーティクルフィルタの基礎をなすのは，信念 $bel(x_t)$ をパーティクルの集合 X_t で表現するという考え方である．理想的には，ある状態の仮説 x_t がパーティクルの集合 X_t に含まれる度合いは，ベイズフィルタの事後信念 $bel(x_t)$ に比例すべきである．つまり，

$$x_t^{[m]} \sim p(x_t \mid z_{1:t}, u_{1:t}) \quad (2.3.23)$$

というように選ばれるべきである．式(2.3.2)は，ある領域に標本が多く存在すればするほど，真の状態がその領域に存在する可能性が高くなることを意味している．通常のパーティクルフィルタの場合, 式(2.3.2)のような性質は $M \uparrow \infty$ の場合においてのみ漸近的に成り立つ． M が有限な場合，パーティクルは微妙に異なる分布から選ばれる．しかし実用上，パーティクルの数が少なすぎない限りは(例えば $M \le 100$ など)この違いの影響は無視できる．

パーティクルフィルタアルゴリズムは, 信念 $bel(x_t)$ を 1 ステップ前の信念 $bel(x_{t-1})$ から再起的に構築する．信念はパーティクルの集合で表現されるため，この処理はパーティクルの集合 X_{t-1} からパーティクルの集合 X_t を構築することを意味する．

最も基本的なパーティクルフィルタのアルゴリズムを図 2.3.5 に示す. このアルゴリズムの入力は，最新の制御 u_t，計測 z_t，パーティクルの集合 X_{t-1} である．このアルゴリズムでは，信念 $\overline{bel}(x_t)$ を表現する一時的なパーティクルの集合 $\overline{X}$ が最初に作成される．この過程では，入力された集合 X_{t-1} 中のパーティクル $x_{t-1}^{[m]}$ が順に処理されて集合 $\overline{X}$ が構成される．その後，パーティクルの集合に対してある変換が行われ，事後確率 $bel(x_t)$ を近似するパーティクルの集合 X_t が構成される．

図 2.3.5 の 8〜11 行目のリサンプリングステップは，パーティクルに事後信念 $bel(x_t)$ を表現させる上で重要な機能である．リサンプリングは，ダーウィンの**適者生存**の考え方を確率的に実装したものであり，この作用によって，パーティクルの集合が状態空間中で事後確率の高い領域に集められる．そのようにすることにより，状態空間中で確率の高い領域に対し，フィルタアルゴリズムのための計算資源を集中させることが出来る．

1: **Algorithm Particle_filter**(X_{t-1}, u_t, z_t):
2: $\overline{X}_t = X_t = \emptyset$
3: *for m = 1 to M do*
4: *sample* $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$ //重要度係数 $w_t^{[m]}$ の計算
6: $\overline{X}_t = \overline{X}_t + \left\langle x_t^{[m]}, w_t^{[m]} \right\rangle$
7: *endfor*
8: *for m = 1 to M do* //リサンプリング
9: *draw* i *with probability* $\propto$ $w_t^{[i]}$
10: *add* $x_t^{[i]}$ *to* X_t
11: *endfor*
12: *return* X_t

図 2.3.5　パーティクルフィルタのアルゴリズム

2.4. 移動ロボットの動作

ここでは，フィルタリングアルゴリズムを実装するために必要である動作モデルについて述べる．動作モデルは，確率密度関数：

$p(x_t \mid u_t, x_{t-1})$　（2.4.1）

を定義づけるモデルである．この状態遷移モデルは，上記のフィルタ手法の予測ステップにおいて主要な役割を果たす．ここで x_t と x_{t-1} はどちらもロボットの姿勢であり（x 座標のことではない），u_t は動作コマンドである．このモデルは，x_{t-1} において動作コマンド u_t が実行させたときの，ロボットの運動額的状態に対する事後確率分布を表現するものである．

本稿では実装する事も考え，二輪移動ロボットの動作モデルについて説明する．移動ロボットの旋回角度(回転速度)を ω，移動ロボットの中心速度(並進速度)を v とすると，制御は，

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} \quad (2.4.2)$$

となる．また，旋回半径を ρ とすると，移動ロボットの中心速度は，

$v = \rho\omega$　（2.4.3）

で表される．一方，移動ロボットの中心から車輪までの距離を d とすると，各車輪の旋回速度は，

$$\begin{aligned} v_R &= (\rho + d)\omega \\ v_L &= (\rho - d)\omega \end{aligned} \quad (2.4.4)$$

となる．

つまり，移動ロボットの旋回角度 ω，中心速度 v，旋回半径 ρ はそれぞれ，

$$\omega = \frac{v_R - v_L}{2d}$$
$$v = \frac{v_R + v_L}{2} \qquad (2.4.5)$$
$$\rho = \frac{d(v_R + v_L)}{v_R - v_L}$$

で得られる．一方，移動ロボットを 2 次元座標系で見ると，

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta \quad (2.4.6)$$
$$\dot{\theta} = \omega$$

となる．

次に移動制御を時間 Δt において考える．Δt の間に，車輪および移動ロボットの中心がそれぞれ ΔL_L，ΔL_R，ΔL だけ移動し，旋回中心周りに $\Delta\rho$ だけ回転したとすると，

$$\Delta L_R = (\rho + d)\Delta\theta$$
$$\Delta L_L = (\rho - d)\Delta\theta \quad (2.4.7)$$
$$\Delta L = \rho\Delta\theta$$

なので，

$$\Delta L = \frac{\Delta L_R + \Delta L_L}{2}$$
$$\Delta\theta = \frac{\Delta L_R - \Delta L_L}{2d} \quad (2.4.8)$$
$$\rho = \frac{\Delta L}{\Delta\theta}$$

となる．

次に座標で考え，ロボットの姿勢の算出方法を示す．まず，ある時間における角度は，

$$\theta_{i+1} = \theta_i + \Delta\theta \quad (2.4.9)$$

となるので，姿勢に関して，Δが微小な場合は，

$$x_{i+1} = x_i + \Delta L\cos(\theta_i + \frac{\Delta\theta}{2})$$
$$y_{i+1} = y_i + \Delta L\sin(\theta_i + \frac{\Delta\theta}{2}) \qquad (2.4.10)$$

で表せ，Δが大きい場合は，図 2.4.4 のように L' を考え，

$$L' = 2\rho\sin\left(\frac{\Delta\theta}{2}\right) \quad (2.4.11)$$

を繰り返し計算することでロボットの姿勢を知ることができる．

このように，ロボットの車輪の回転角度を観測することでロボットを制御できる．この手法は，デッドレコニング(DR)と呼ばれている．

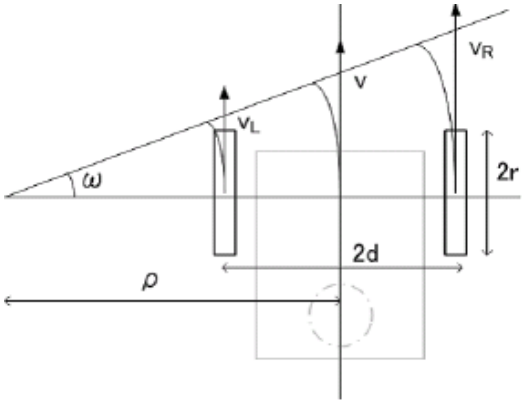


図 2.4.1　車輪の旋回

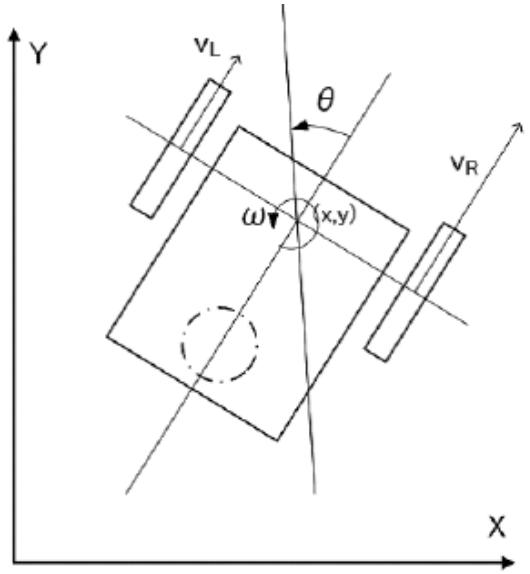


図 2.4.2　座標系で見た移動ロボット

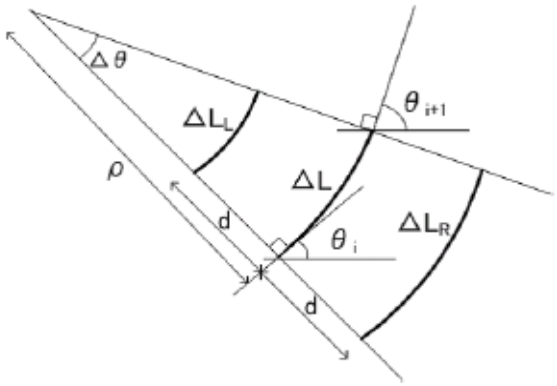


図 2.4.3　円弧に近似

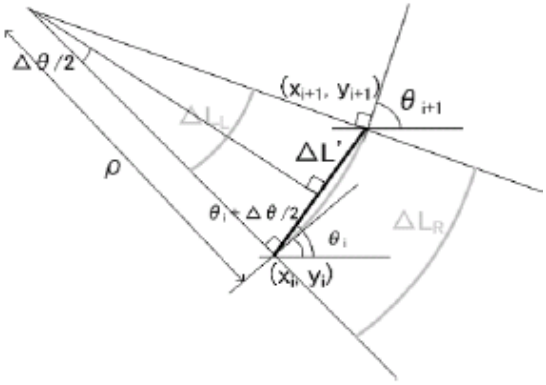


図 2.4.4　移動による座標の移動

2.5. ロボットの知覚

2.5.1. 環境計測モデル

環境計測モデルは，ロボットの動作モデルと共に，確率ロボティクスで用いられるもう一つのモデルである．計測モデルは，現実の世界で生成されたセンサ計測値の生成過程を記述するものである．確率ロボティクスではセンサ計測値の雑音が明示的にモデルに組み込まれる．そのようなモデルは，ロボットのセンサに内在する特有の不確かさを説明するものになる．形式的には，計測モデルは条件付確率分布 $p(z_t \mid x_t, m)$ で定義される．ここで x_t はロボットの姿勢，z_t は時刻 t における計測，そして m は環境の地図である．本稿では，z_t を得るためのセンサとして，距離計測について説明する．

本稿では，移動ロボットとランドマークの距離を，距離の増加に伴って減少する通信時の電界強度(Receive Signal Strength Indication，RSSI)の大小関係を，距離の大小関係に対応付けて用いる．もちろんソナーレンジスキャンなどを用いて距離の絶対値を知ることができればそれに越したことはない．しかし，本稿で最低限要求される情報は，距離の大小関係という相対的な情報である．

ここでは，自由空間における電界強度の特性を示す．

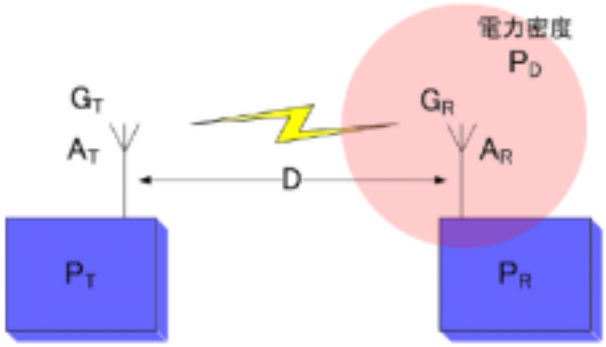


図 2.5.1　自由空間における 2 つのノードの通信

自由空間とは一般に，通信を行う端末間に電波を反射したり遮断する物体がない，極めて電波的に理想的な環境のことをいう(図 2.5.1 参照)．その自由空間においては，送受信される電波は，直接波のみである．つまり，回折波や透過波や反射波を考慮に入れないモデルである．その電界強度はフリスの伝達公式(式 2.5.3) によって求められる．フリスの伝達公式は，送信電力と D[m] 遠方の地点におけるアンテナ受信電力の関係を表わす．受信地点の電力密度 P_D（式 2.5.1）と受信アンテナの実効面積 A_R（式 2.5.2）から受信電力 P_R を求める事ができる．$P_R = P_D \times A_R$ であるので，受信電力 P_R は式 2.5.3 のようになる．つまり，フリスの伝達公式より，自由空間における電界強度は単純に，通信距離の 2 乗に反比例して減衰する．

$$P_D = \frac{G_T P_T}{4\pi D^2} \quad [W/m^2] \quad （2.5.1）$$

$$A_R = \frac{\lambda^2}{4\pi} G_R \quad [m^2] \quad （2.5.2）$$

$$P_R = P_D A_R = \left(\frac{\lambda}{4\pi D}\right)^2 G_T G_R P_T \quad [W] \quad （2.5.3）$$

P_D　：電力密度$[W/m^2]$
P_R・P_T　：受信・送信電力$[W]$
A_R　：受信側のアンテナ実行面積$[m^2]$
G_R・G_T　：受信側・送信側のアンテナ絶対利得$[dBi]$
D　：通信距離$[m]$

このとき，受信電力$[W]$を[dBm]単位に直すと，

$$RSSI[dBm] = 10(-2\log x + \log 2P_t + 2\log\frac{G\lambda}{4\pi}) \quad （2.5.4）$$

となる．ここでλは電波の波長$[m]$であり，電波の周波数を $f[Hz]$，真空中の電波の速度を $C = 3\times10^8$ とした場合，

$$\lambda[m] = \frac{C[m]}{f[Hz]} \quad （2.5.5）$$

となる．

前述のセンサノード MOTE の自由空間におけるノード間距離に対する RSSI 値を求めると下図のような関係性を得られる．MOTE のアンテナ特性は表 2-3 のようになっている．

表 2-3　MOTE のアンテナ仕様とパラメータ設定

製品名	Micaz MOTE
アンテナ名称	λ/4接地型垂直アンテナ
仕様周波数	2.4 GHz
送信電力 (mW)	1
アンテナゲイン	2.15 [dBi] (1.64[倍])

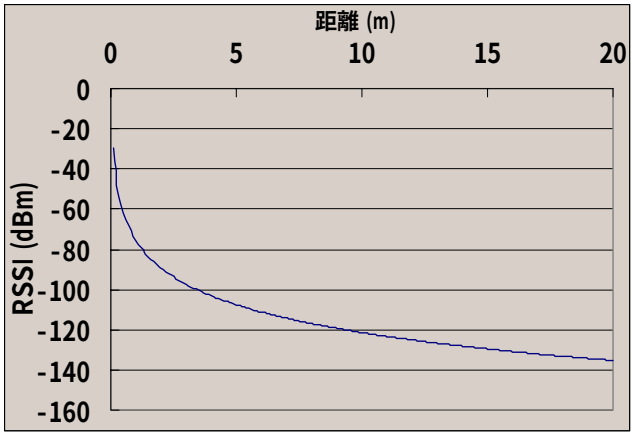


図 2.5.2　RSSI と距離の関係性

2.5.2. ランドマーク計測

ランドマークを扱うためのセンサモデルをここで構築しておく．地図作成において，地図 m を環境中の物体及びそれらの特性のリストとして定義される．

$$m = \{m_1, m_2, \ldots m_N\} \quad (2.5.6)$$

ここで，N は環境中の物体の個数を表し，各 $m_n (1 \le n \le N)$ は各物体の特性を表している．通常，地図のインデックスは**特徴ベース**と**位置ベース**に分類される．特徴ベースの地図では，n は特長に対するインデックスになる．m_n の値には，その特徴に関する情報と共に，デカルト座標系におけるその特徴の位置が含まれる．位置ベースの地図では，インデックス n はある特定の位置と対応付けられる．平面の地図では，世界座標（x y）を直接表現するために，地図の要素を m_n の代わりに $m_{x,y}$ で表現される．

ランドマーク計測モデルは，前者の特徴ベースの地図においてのみ定義できる．各特長は特徴量と位置座標を有する．位置座標は，単に地図のグローバル座標中における位置のことであり，$m_{i,x}$ $m_{i,y}$ で表記される．

ランドマーク知覚における雑音を，距離，方向，特徴量に対するそれぞれ独立したガウス雑音としてモデル化する．結果として得られる計測モデルは，時刻 t における i 番目の特徴が，地図にある j 番目のランドマークに対応する場合に対して定式化される．例によってロボットの姿勢を $x_t = (x\ y\ \theta)^T$ とすると，

$$\begin{pmatrix} r_t^i \\ \phi_t^i \\ s_t^i \end{pmatrix} = \begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \mathrm{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \\ s_j \end{pmatrix} + \begin{pmatrix} \varepsilon_{\sigma_r^2} \\ \varepsilon_{\sigma_\phi^2} \\ \varepsilon_{\sigma_s^2} \end{pmatrix} \quad (2.5.7)$$

となる．ここで $\varepsilon_{\sigma_r}, \varepsilon_{\sigma_\phi}, \varepsilon_{\sigma_s}$ は，それぞれ平均ゼロ，標準偏差 $\varepsilon_r, \varepsilon_\phi, \varepsilon_s$ のガウス分布に従う誤差である．

2.5.3. データの対応付け問題

距離計測の一番の問題は，データの対応付け問題として知られる．この問題は，ランドマークが一意に特定できない場合に起こる．そのようなとき，ランドマークの識別に関して幾分かの不確かさが残る．この場合，距離・方向計測モデルを作成するためには，特徴 f_t^i と地図中のランドマーク m_j 間の対応変数を導入することが有用である．この変数は $c_t^i (c_t^i \in \{1,\ldots,N+1\})$ と表される．N は地図 m 中のランドマークの数である．$c_t^i = N+1$ のときのみは例外であり，この時特徴の観測は地図中のどの特長とも対応付けられない．これは，偽のランドマークを扱うために重要である．また，ロボットによる地図作成に対しても大変有意義である．それは，ロボットがまだ観測されていないランドマークに出会うことがあるからである．この問題は，式(2.5.7)で実装できる．

2.6. SLAM

ここでは，SLAM (simultaneous localization and mapping)について述べる．また，CML (Concurrent Mapping and Localization)としても知られている．SLAM はロボットが環境の地図を利用できず，さらに自身の姿勢も分からないときに生じる問題である．代わりに与えられるのは，計測 $z_{1:t}$ と制御 $u_{1:t}$ のみである．“ simultaneous localization and mapping ”という用語は，そのような結果生じる問題を表現している．つまり，SLAM では，ロボットは環境の地図を，地図上での自己位置を推定しながら生成しなければならない．SLAM は非常に難しい問題である．SLAM では地図が未知であり，さらに地図も推定しなければならないという点で位置推定よりも困難である．また，姿勢が既知の場合の地図生成よりも困難である．姿勢が分からず，さらに姿勢も推定しなければならないからである．

確率論の観点からは，SLAM には主に 2 種類の形式が存在する．そのどちらも，実用上，同等に重要である．一方は**オンライン SLAM 問題**（図 2.6.1 参照）として知られており，各時刻において地図と姿勢の事後確率：

$$p(x_t, m \mid z_{1:t}, u_{1:t}) \quad (2.6.1)$$

が推定される．ここで x_t は時刻 t における姿勢，m は地図，$z_{1:t}$ は計測，$u_{1:t}$ は制御である．この問題がオンライン SLAM と呼ばれるのは，各時刻 t で変数の推定が行われるからである．多くのオンラインＳＬＡＭアルゴリズムは逐次的である．つまり，過去の計測値や制御は，それらが推定に反映された時点で破棄される．

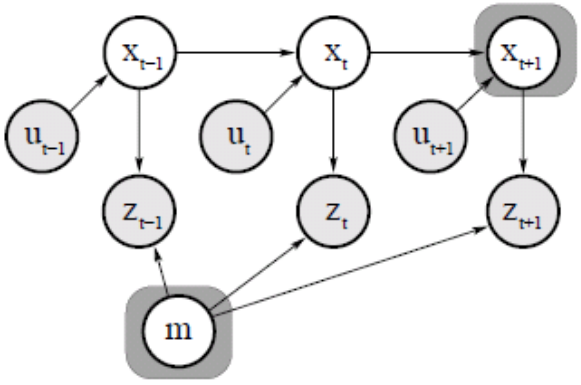


図 2.6.1　オンライン SLAM 問題のモデル

もう一方の SLAM 問題は，**完全 SLAM 問題**（図 2.6.2 参照）と呼ばれている．完全 SLAM 問題では最新の姿勢 x_t だけに対してではなく，ロボットの軌跡全体 $x_{1:t}$ に対して事後確率：

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \quad (2.6.2)$$

が計算される．

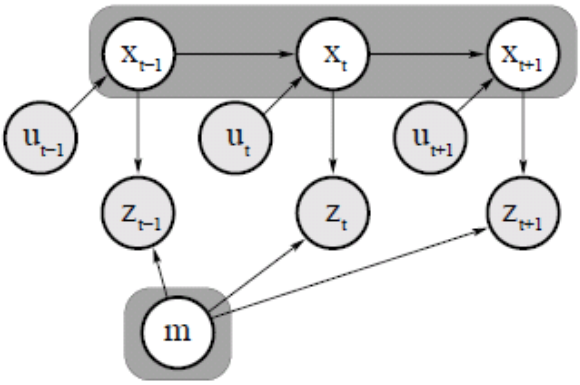


図 2.6.2　完全 SLAM 問題のモデル

オンライン SLAM と完全 SLAM は微妙に異なるため, 問題を解くためのアルゴリズムは違ってくる．具体的には，オンライン SLAM 問題は，完全 SLAM 問題から過去の姿勢に関して積分した結果：

$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \iint \cdots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) dx_1 dx_2 \ldots dx_{t-1} \quad (2.6.3)$$

で定義される．通常，オンライン SLAM では上式の積分は各時刻に 1 回ずつ処理される．

SLAM 問題には，推定問題としての性質として，連続な要素と離散的な要素が混在した問題となる特徴がある．連続的な推定問題は地図中での物体の位置やロボット自身の姿勢の変数と関係している．物体は，特徴ベースの表現におけるランドマークであったり，レンジファインダで検出された物体の一部であったりする．離散的な推定は，対応問題で行わなければならない．ある物体が検出されたとき，SLAM アルゴリズムはその物体と以前に検出した物体の関係を推論しなければならない．この推論は，通常は離散的である．つまり，検出された物体が以前のものと同じか，あるいは違うかということが推論される．

また，前述の対応変数を明示的に定義することが有効なことがある．このとき，オンライン SLAM の事後確率は,

$$p(x_t, m, c_t \mid z_{1:t}, u_{1:t}) \quad (2.6.4)$$

で与えられる．一方，完全 SLAM 問題の事後確率は,

$$p(x_{1:t}, m, c_{1:t} \mid z_{1:t}, u_{1:t}) \quad (2.6.5)$$

となる．オンライン SLAM の事後確率は，完全 SLAM の事後確率から，過去のロボットの姿勢に関して積分し，過去の対応に関して和をとることで,

$$p(x_t, m, c_t \mid z_{1:t}, u_{1:t}) = \iint \cdots \int \sum_{c_1} \sum_{c_2} \cdots \sum_{c_{t-1}} p(x_{1:t}, m, c_{1:t} \mid z_{1:t}, u_{1:t}) dx_1 dx_2 \ldots dx_{t-1} \quad (2.6.6)$$

のように計算できる．この両方の SLAM 問題では，式(2.6.4) と式(2.6.5)の事後確率分布を完全に推定することが基本となる．完全は事後確率分布は，地図と姿勢，あるいは地図と軌跡について分かる情報を全て反映したものとなる．

実際には，完全な事後確率分布を計算することは実行不可能である．その原因は,

(1) 連続パラメータ空間の次元が高いこと

(2) 離散的な対応変数の数が多いこと

である．最新鋭の SLAM アルゴリズムでは，数万あるいはそれ以上の特徴から地図が作成される．対応が既知であったとしても，地図に対する事後確率に対してだけでも，10^5 以上の次元の空間に対する確率分布の計算が必要になる．このことは，三次元の位置推定問題とは著しく異なる．さらに，SLAM の応用例の多くでは，対応関係は未知である．全ての対応変数のベクトル $c_{1:t}$ に対して可能性のある場合の数は，時刻 t と共に指数乗的に増加する．したがって，対応問題を解決できる実用的な SLAM アルゴリズムは，近似に頼らざる

を得ない.

以下に二種類の SLAM アルゴリズムについて論述する．本稿では,

- ランドマークは MOTE（ビーコン端末）として扱われる
- 各ビーコン端末には，ビーコン ID を任意に設定できる

という点から，各 SLAM 手法において対応関係が**既知**の場合の手法について述べる.

2.6.1. EKF SLAM

歴史的に最も初期の SLAM アルゴリズムは，拡張カルマンフィルタ（EKF）を基にしたものである．一言で説明すると，EKF SLAM アルゴリズムは，最尤データ対応付け手法を用いて EKF をオンライン SLAM に適用したものである．そのことから，EKF SLAM は次のような数多くの近似や制限的な過程をも免れない.

EKF SLAM では，地図は特徴ベースであり，点ランドマークで構成される．計算上の問題から，点ランドマークの数は少ないことが普通である（1000 以下）．さらに，EKF を用いるときは，ランドマークに曖昧さが無いほどよく機能する傾向がある．この理由から，EKF SLAM では，特徴抽出にかなりのテクニックが必要となり，よく人工ビーコンが特徴として用いられる.

EKF SLAM ではガウス雑音の仮定がロボットの動作や知覚に必要となる．そして，事後確率の不確かさは比較的小さくなければならない．そうでないと，EKF の線形近似で許容範囲を超えた誤差が生じがちになる．EKF SLAM アルゴリズムは，**ポジティブ**なランドマークの発見情報しか処理できない．センサ計測でランドマークが検知されないことで生じるネガティブ情報に対応していない．これは，ガウス分布による信念の表現が直接の原因である.

対応関係が既知の場合の SLAM アルゴリズムでは，SLAM 問題の連続的な部分だけを扱えばよい．この場合の SLAM アルゴリズムの開発は，ロボットの位置 x_t を推定することにくわえて，途中で出会った全てのランドマークの座標を推定するということである．そのため，状態ベクトルにランドマークの座標を加えなければならない.

便宜上, ロボットの姿勢と地図の状態を含む状態ベクトルを複合状態ベクトルと呼び, y_t で表す．複合状態ベクトルは,

$$y_t = \begin{pmatrix} x_t \\ m \end{pmatrix} = \begin{pmatrix} x & y & \theta & m_{1,x} & m_{1,y} & s_{1,y} & m_{2,x} & m_{2,y} & s_2 & \dots & m_{N,x} & m_{N,y} & s_N \end{pmatrix}^T \quad (2.6.7)$$

で与えられる．ここで，x, y, θ は時刻 t におけるロボットの座標を表す(状態変数 x_t, y_t とは違う)．また，$m_{i,x}, m_{i,y}$ $(i = 1, \dots, N)$ は i 番目のランドマークの座標を表し，s_i はその特徴量である．この状態ベクトルの次元は $3N+3$ である．ここで N は地図中のランドマークの数を表す．妥当な N の値を考えた場合，明らかに，このベクトルの次元は著しく大きくなる．EKF SLAM では，オンライン事後確率 $p(y_t \mid z_{1:t}, u_{1:t})$ が計算される.

ここで，EKF SLAM アルゴリズムの導出を行う．SLAM では，ロボットの初期姿勢が座標系の原点として扱われる．他の座標系への変換が出来るため，この定義にはいくらかの任意性が残されている．最初，ランドマークの位置は全て**未知**である．次の平均と共分散：

$$\mu_0 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \end{pmatrix}^T \quad (2.6.8)$$

$$\sum_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \infty & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \infty \end{pmatrix} \quad (2.6.9)$$

は，上記の場合の信念を表す．この共分散行列のサイズは$(3N+3)\times(3N+3)$である．この行列には，ロボットの姿勢の変数に関する3×3のゼロ行列が含まれている．他の部分の対角成分は全て無限になる．

ロボットが動くと，状態ベクトルは雑音のない速度モデルに従って変化する．SLAM では，この動作モデルは複合状態ベクトルのために，

$$y_t = y_{t-1} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega_t\Delta t + \gamma_t\Delta t \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad (2.6.10)$$

のように拡張される．変数x, y, θはy_{t-1}中でロボットの姿勢を表す．ロボットの動作は姿勢のみに影響し，ランドマークの位置はそのままになるため，最初の 3 個の要素だけが非ゼロである．これにより，同じ式をよりコンパクトに，

$$y_t = y_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega_t\Delta t + \gamma_t\Delta t \end{pmatrix} \quad (2.6.11)$$

と記述できるようになる．ここで，

$$F_x = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0 \quad 0 \quad \cdots \quad 0}_{3N \text{ columns}} \\ 0 & 1 & 0 & \\ 0 & 0 & 1 & \end{pmatrix} \quad (2.6.12)$$

は，三次元状態ベクトルを$3N+3$次元のベクトルに写像するための行列である．雑音を考慮した動作モデルは，

$$y_t = y_{t-1} + F_x^T \underbrace{\begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega\Delta t \end{pmatrix}}_{g(u_t, y_{t-1})} + N(0, F_x^T\ R_t\ F_x) \quad (2.6.13)$$

のようになる．ここで $F_x^T\ R_t\ F_x$ は共分散行列を状態ベクトルの次元まで拡張したものである．

EKF と同じように，動作関数 g は一次テイラー展開で近似され，

$$g(u_t, y_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(y_{t-1} - \mu_{t-1}) \quad (2.6.14)$$

となる．ここでヤコビ行列 $G_t = g'(u_t, \mu_t)$ は，u_t において g を微分したものである．

明らかに，式(2.6.13)の加法的な形式によって，このヤコビ行列を，

$$G_t = I + F_x^T\ g_t\ F_x \quad (2.6.15)$$

$$\text{ここで，}\ g_t = \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} \quad (2.6.16)$$

というように，一つの $(3N+3)\times(3N+3)$ 単位行列（y_{t-1} の微分）と，ロボットの姿勢変化を特徴付ける低次元のヤコビ行列 g_t の和に分解できる．これらの近似を一般的な EKF アルゴリズムに当てはめると，図 2.6.3 で示す EKF SLAM アルゴリズム 2〜5 行目が得られる．明らかに，5 行目で乗算されているいくつかの行列は疎である．このアルゴリズムを実装するときには，この性質を活用すべきである．この更新の結果，制御 u_t でフィルタを更新した後の，時刻 t における推定の平均 $\overline{\mu}_t$ と共分散 $\overline{\sum}_t$ が得られる．計測 z_t はこの時点では反映されていない．

計測更新の導出を以下に示す．具体的には，計測モデルは，

$$z_t^i = \underbrace{\begin{pmatrix} \sqrt{(m_{j,x}-x)^2 + (m_{j,y}-y)^2} \\ \text{atan2}(m_{j,y}-y, m_{j,x}-x) - \theta \\ m_{j,s} \end{pmatrix}}_{h(y_t, j)} + N\left(0, \underbrace{\begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}}_{Q_t}\right) \quad (2.6.17)$$

で与えられる．ここで，x, y, θ はロボットの姿勢，i は z_t 中のここのランドマーク観測に与えられるインデックス，そして $j = c_t^i$ は時刻 t に観測されたランドマークのインデックスである．変数 r, ϕ, s はそれぞれランドマークとの距離, 方向, 特徴量である．そして，$\sigma_r, \sigma_\phi, \sigma_s$ はそれらの計測雑音の分散である．

この式は，線形関数

$$h(y_t, j) \approx h(\overline{\mu}_t, j) + H_t^i(y_t - \overline{\mu}_t) \quad (2.6.18)$$

で近似される．ここでH_t^iは，状態ベクトルy_tに対するhの微分である．hはこの状態ベクトルの二つの要素：ロボットの姿勢x_tとj番目のランドマークの位置m_jにしか依存していないため，この微分は低次元のヤコビ行列h_t^iと行列$F_{x,j}$に，

$$H_t^i = h_t^i F_{x,j} \quad (2.6.19)$$

のように分解できる．$F_{x,j}$はh_t^iを状態ベクトルの次元に写像する行列である．h_t^iは状態変数x_t, m_jに対して計算された，関数$h(y_t, j)$の$\bar{\mu}_t$におけるヤコビ行列：

$$h_t^i = \begin{pmatrix} \frac{\bar{\mu}_{t,x} - \bar{\mu}_{j,x}}{\sqrt{q_t}} & \frac{\bar{\mu}_{t,y} - \bar{\mu}_{j,y}}{\sqrt{q_t}} & 0 & \frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q_t}} & \frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q_t}} & 0 \\ \frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{q_t} & \frac{\bar{\mu}_{t,x} - \bar{\mu}_{j,x}}{q_t} & -1 & \frac{\bar{\mu}_{t,y} - \bar{\mu}_{j,y}}{q_t} & \frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{q_t} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.6.20)$$

である．$q_t = (m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2$であり，また以前と同様，$j = c_t^i$は計測$z_t^i$と対応するランドマーク(のインデックス)である．行列$F_{x,j}$の次元は$6 \times (3N+3)$であり，

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & \underbrace{0 \cdots 0}_{3j-3} & 0 & 0 & 1 & \underbrace{0 \cdots 0}_{3N-3j} \end{pmatrix} \quad (2.6.21)$$

となる．これにより低次元の行列h_t^iを$3 \times (3N+3)$行列にできる．以上の数式によって，図2.6.3に示すEKF SLAMアルゴリズムにおける，8〜17行目でのカルマンゲインの計算が導出される．ただし，この導出の他に，もう一つの重要な拡張が必要である．というのも，あるランドマークが始めて観測されたとき，式(2.6.8)で設定された初期の推定姿勢で線形化すると誤差が大きくなるという問題が存在するからである．それは，hが線形化される点が$(\hat{\mu}_{j,x}, \hat{\mu}_{j,y}, \hat{\mu}_{j,s})^T = (0\ 0\ 0)^T$になるからであり，実際のランドマークの位置と大きく異なるからである．より適切な初期推定位置は，図2.6.3の10行目で与えられる．ここでは，ランドマークの推定$(\hat{\mu}_{j,x}, \hat{\mu}_{j,y}, \hat{\mu}_{j,s})^T$が，観測からの期待値で初期化されている．この期待位置は，ロボットの姿勢の期待値と，ランドマークの計測値から，

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix} \quad (2.6.22)$$

のように計算される．この初期化は，計測の関数hが全単射である場合のみ可能であるということに注意したい．計測値は二次元であり，ランドマークの位置も二次元である．計測値の次元がランドマークの座標よりも小さいときは，hが射影になり，一つの計測だけでラ

ンドマークの位置の期待値$(\overline{\mu}_{j,x}, \overline{\mu}_{j,y}, \overline{\mu}_{j,s})^T$を計算することは不可能である．例えば，コンピュータビジョンを用いて SLAM を実装した場合，そのようなことが起きる．というのも，カメラではランドマークの方向は分かっても距離が分からないことが多いからである．そのような場合，SLAM では，複数の観測が三角測量で統合されてランドマークの初期位置が決定されることが多い．SLAM では，このような問題は **bearing only SLAM** と呼ばれている．

最後に，EKF アルゴリズムに必要なメモリは，地図中のランドマーク数 N の二次関数になることを述べておく．計算時間も同様である．このような二次の計算複雑性は，EKF の様々なところで行われる行列の計算に起因する．

1: ***Algorithm EKF_SLAM***$(\mu_{t-1}, \sum_{t-1}, u_t, z_t, c_t)$：

2: $$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \underbrace{0 \quad \cdots \quad 0}_{3N} & & \end{pmatrix}$$

3: $$\overline{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega\Delta t \end{pmatrix}$$

4: $$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

5: $$\overline{\sum}_t = G_t \sum_{t-1} G_t^T + F_x^T R_t F_x$$

6: *for all observed features* $z_t^i = (r_t^i \ \phi_t^i \ r_t^i)^T$ *do*

7: $$j = c_t^i$$

8: *if landmark* j *never seen before*

9: $$\begin{pmatrix} \overline{\mu}_{j,x} \\ \overline{\mu}_{j,y} \\ \overline{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \overline{\mu}_{t,x} \\ \overline{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \overline{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \overline{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

10: *endif*

11: $$\delta = (\delta_x \quad \delta_y)^T = (\overline{\mu}_{j,x} - \overline{\mu}_{t,x} \quad \overline{\mu}_{j,y} - \overline{\mu}_{t,y})^T$$

12: $$q = \delta^T \delta$$

13: $$\hat{z}_t^i = (\sqrt{q} \quad \text{atan2}(\delta_y, \delta_x) - \overline{\mu}_{t,\theta} \quad \overline{\mu}_{j,s})^T$$

14: $$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0\cdots0}_{} & 0 & 0 & 0 & \underbrace{0\cdots0}_{} \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & \underbrace{0\cdots0}_{3j-3} & 0 & 0 & 1 & \underbrace{0\cdots0}_{3N-3j} \end{pmatrix}$$

15: $$H_t^i = \frac{1}{q}\begin{pmatrix} \sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & -\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & \delta_x & -1 & -\delta_y & -\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,j}$$

16: $$K_t^i = \overline{\sum}_t H_t^{iT}(H_t^i \overline{\sum}_t H_t^{iT} + Q_t)^{-1}$$

17: $$\overline{\mu}_t = \overline{\mu}_t + K_t^i(z_t^i - \hat{z}_t^i)$$

18: $$\overline{\sum}_t = (I - K_t^i H_t^i)\overline{\sum}_t$$

19: *endfor*

20: $\mu_t = \overline{\mu}_t$

21: $\sum_t = \overline{\sum}_t$

22: *return* $\mu_t, \sum_t$

図 2.6.3　EKF SLAM のアルゴリズム

2.6.2. FastSLAM

ここでは，パーティクルフィルタを SLAM に導入することを考える．特徴の位置推定に関する従属性は，ロボットの姿勢が不確かな時だけに生じるものであり，この構造によって，Rao-Blackwellized パーティクルフィルタとして知られるパーティクルフィルタを SLAM に適用できるようになる．Rao-Blackwellized パーティクルフィルタでは，いくつかの変数に対する事後確率分布だけがパーティクルフィルタで表現され，他の変数の表現にはガウス分布が用いられる．

FastSLAM では，ロボットの軌跡の推定にパーティクルフィルタが用いられる．各パーティクルの姿勢の下で，地図に関する誤差は互いに条件付き独立となる．したがって，地図作成の問題が多数の別々な問題に分解される．各問題は，地図中の各特長に対するものである．FastSLAM は，これらの特徴の位置を EKF で推定する．この EKF は，各特長に対する低い次元でのものである．このような方法は，EKF SLAM のアルゴリズムと根本的に異なるものである．これまでのアルゴリズムでは，全特徴に関する位置情報を含んだ一つの大きなガウス分布が演算の対象であった．

基本的は FastSLAM アルゴリズムは，特徴の数に対して，対数時間で演算可能なように実装できる．したがって，FastSLAM は，EKF よりも計算量の点で有利である．しかしながら，FastSLAM には，もっと重要な利点がある．それは，データ対応付けの判断がパーティクル一つ一つに対して行われることに由来する．その結果，フィルタでは一つではなく，複数のデータ対応付けに対する事後確率が記録され続ける．これは，EKF SLAM とは全く対照的な性質である．EKF SLAM では，各時刻において，一つのデータ対応付けしか追従していなかった．データ対応付けに対するサンプリングを行うことで，FastSLAM は最尤な対応付けを計算するだけではなく，データ対応付けの事後確率分布を近似できる．複数の対応付けを同時に追跡できる能力があるため，FastSLAM には琢次的は最尤データ

対応付けに基づいたアルゴリズムよりも，顕著なロバスト性がある．

EKF SLAM アルゴリズムでは，非線形名ロボットの動作モデルを線形化する必要があった．一方，パーティクルフィルタは非線形名ロボットの動作モデルに対応している．このことは，ロボットの運動の非線形性が大きいときや，不確かさが比較的大きいときに効果を発揮する．

また，パーティクルフィルタを用いることによって，FastSLAM は完全 SLAM 問題とオンライン問題を両方解くことが出来る．FastSLAM は，特徴の位置が条件付き独立性を満たすために，軌跡全体に対する事後確率分布を計算するように定式化される．しかし，パーティクルフィルタでは各時刻で一回の姿勢推定を行うため，FastSLAM はオンラインアルゴリズムになる．それゆえに，FastSLAM はオンライン SLAM 問題も解くことになる．FastSLAM が両方の問題に対応できる唯一のアルゴリズムである．

FastSLAM アルゴリズムにおけるパーティクルは，図 2.6.4 のような構造を持つ．各パーティクルには，ロボットの軌跡の推定値 $x_{1:t}^{[k]}$ と，地図中の各特長 m_j に対するカルマンフィルタが含まれている．カルマンフィルタは，平均 $\mu_{j,t[k]}$ と共分散 $\sum_{j,t[k]}$ で表される．ここで $[k]$ はパーティクルのインデックスである．また，パーティクルの数は M で表される．

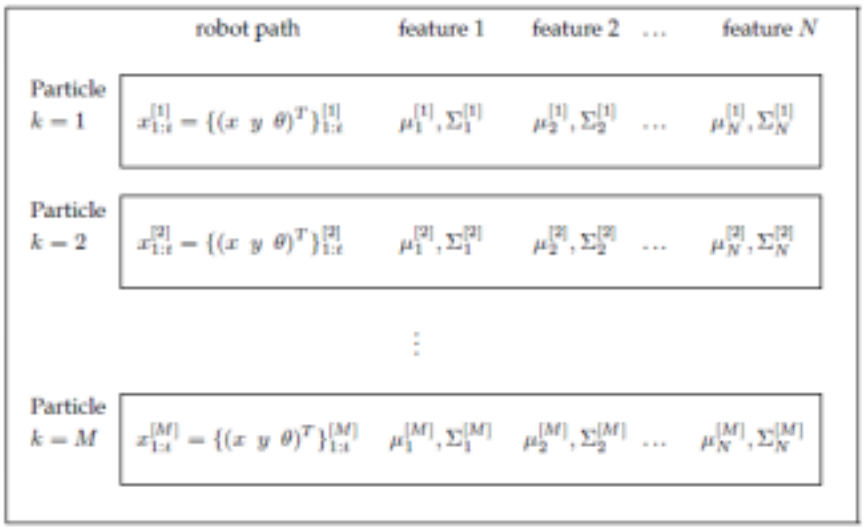


図 2.6.4　各パーティクルの構造

ここでは，SLAM 問題の基本的な数学的性質から各ステップを導出する．ここでの導出においては，FastSLAM がオンライン SLAM ではなく，完全 SLAM 問題を解くことを想定する．完全 SLAM 問題のため，各パーティクルは全時刻の軌跡の標本となっている．しかし一方で，更新には最新時刻の姿勢しか必要としないので，FastSLAM はフィルタのようにオンラインで実行できる．

式(2.6.2)にある完全 SLAM の事後確率分布 $p(y_{1:t} \mid z_{1:t}, u_{1:t})$ が，

$$p(y_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \prod_{n=1}^{N} p(m_n \mid x_{1:t}, z_{1:t}, c_{1:t}) \quad (2.6.23)$$

と積の形式に変形できることは, FastSLAM で重要な数学的洞察である. この**因子分解**は, 軌跡と地図に対する事後確率の計算が，$N+1$ 個の確率の計算に分解できることを表している．FastSLAM では，地図中の各特長については，位置に対して別々の推定 $p(m_n \mid x_{1:t}, z_{1:t}, c_{1:t})$ （$n = 1, \ldots, N$）が行われる. このように事後推定の確率分布が因子分解できるという性質は,「構造化されていない」事後分布を推定するための SLAM アルゴリズ

ムに対し, 計算量の面で多大な利点をもたらす. FastSLAM は, $MN+1$個のフィルタを別々に実行することで，因子分解された形式を利用する．このフィルタの次元は，どれも低いものとなる．また，因子分解から分かるように，FastSLAM は各特長に対して別々の EKF を計算する．つまり，EKF の個数は全体で NM 個になり，各特長，各パーティクルに対して一つずつ EKF が実行される．これにより，EKF SLAM よりもずっと効率の良い更新が出来るようになる．

本稿では対応関係が既知であると前述した．この場合の FastSLAM のパーティクルは，

$$Y_t^{[k]} = \left\langle x_t^{[k]}, \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, \ldots, \mu_{N,t}^{[k]}, \Sigma_{N,t}^{[k]} \right\rangle \quad (2.6.24)$$

と表記される．これまでのように，$[k]$という記述はパーティクルのインデックスを表す．$x_t^{[k]}$はロボットの軌跡を推定するための変数であり，$\mu_{n,t}^{[k]}, \Sigma_{n,t}^{[k]}$は，それぞれ$n$番目の特徴の位置に対する推定の平均値と共分散である. これらの変数が一緒になってk番目のパーティクル$Y_t^{[k]}$を構成する．FastSLAM の事後推定は，そのようなパーティクルM個で表現される．

時刻tにおける事後推定を，時刻$t-1$のものからフィルタリング，あるいは計算するときには，時刻$t-1$におけるパーティクルセットY_{t-1}から新たなセットY_tが生成される．生成される新たなセットには，制御u_tと計測z_t，既知の対応c_tが反映される．この更新は，次のようなステップで行われる．

1. **新たな姿勢のサンプリングによる推定軌跡の延長**

   FastSLAM では，制御u_tが，各パーティクルY_{t-1}からロボットの姿勢x_tをサンプリングするために用いられる．もっと具体的に説明すると，k番目のパーティクルにおいて，動作後の推定姿勢が，

   $$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t) \quad (2.6.25)$$

   で抽出される．ここで$x_{t-1}^{[k]}$は，k番目のパーティクルに記録されている時刻$t-1$出のロボットの姿勢の事後推定である．得られた標本$x_t^{[k]}$は，過去の軌跡$x_{1:t-1}^{[k]}$と共に仮のパーティクルセットに加えられる．サンプリングステップを図 2.6.5 に示す．この図では，一つの初期姿勢から，姿勢のパーティクルが生成されている．

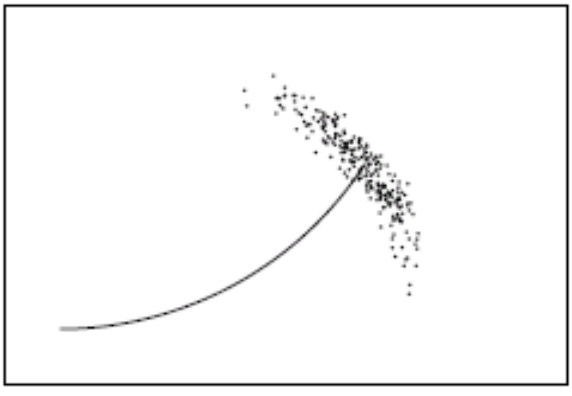


図 2.6.5　確率的動作モデルから得られたパーティクル

2. **観測された特長に対する推定の更新**

   次に FastSLAM では平均$\mu_{n,t-1}^{[k]}$と共分散$\Sigma_{n,t-1}^{[k]}$で現された特徴の事後推定が更新される．更新された値は，仮のパーティクルセットに加えられる．

   更新式は，特徴m_nが時刻tで観測されたかどうかで変化する．特徴nが時刻tに観測

されなかった場合，つまり $n \neq c_t$ のときについては，

$$p(m_n \mid x_{1:t}, c_{1:t}, z_{1:t}) = p(m_n \mid x_{1:t-1}, c_{1:t-1}, z_{1:t-1}) \quad (2.6.26)$$

が適用される．つまり，推定は何も変更されない．これは，

$$\left\langle \mu_{n,t}^{[k]}, \Sigma_{n,t}^{[k]} \right\rangle = \left\langle \mu_{n,t-1}^{[k]}, \Sigma_{n,t-1}^{[k]} \right\rangle \quad (2.6.27)$$

という，単純な更新が行われることを意味する．観測された特徴 $n = c_t$ にたいしては，

$$\begin{aligned} p(m_{ct} \mid x_{1:t}, c_{1:t}, z_{1:t}) &= \frac{p(z_t \mid m_{ct}, x_{1:t}, c_{1:t}, z_{1:t-1}) p(m_{ct} \mid x_{1:t}, c_{1:t}, z_{1:t-1})}{p(z_t \mid x_{1:t}, c_{1:t}, z_{1:t-1})} \\ &= \frac{p(z_t \mid x_t, m_{ct}, c_t) p(m_{ct} \mid x_{1:t-1}, c_{1:t-1}, z_{1:t-1})}{p(z_t \mid x_{1:t}, c_{1:t}, z_{1:t-1})} \end{aligned} \quad (2.6.28)$$

が適用される．この式を正規化の記号 η を用いて表現すると，

$$p(m_{ct} \mid x_{1:t}, c_{1:t}, z_{1:t}) = \eta p(z_t \mid x_t, m_{ct}, c_t) p(m_{ct} \mid x_{1:t-1}, c_{1:t-1}, z_{1:t-1}) \quad (2.6.29)$$

となる．時刻 $t-1$ における $p(m_{ct} \mid x_{1:t-1}, c_{1:t-1}, z_{1:t-1})$ の確率分布は，平均 $\mu_{n,t-1}^{[k]}$，共分散 $\sum_{n,t-1}^{[k]}$ のガウス分布で表現される．時刻 t の推定も同様にガウス分布で表すため，FastSLAM は，計測モデル $p(z_t \mid x_t, m_{ct}, c_t)$ を EKF SLAM と同じように線形化する．例によって，計測関数 h をテイラー展開で，

$$\begin{aligned} h(m_{ct}, x_t^{[k]}) &\approx \underbrace{h(\mu_{ct,t-1}^{[k]}, x_t^{[k]})}_{=:\hat{z}_t^{[k]}} + \underbrace{h'(x_t^{[k]}, \mu_{ct,t-1}^{[k]})}_{=:H_t^{[k]}} (m_{ct} - \mu_{ct,t-1}^{[k]}) \\ &= \hat{z}_t^{[k]} + H_t^{[k]} (m_{ct} - \mu_{ct,t-1}^{[k]}) \end{aligned} \quad (2.6.30)$$

のように近似する．ここで，微分係数 h' は特徴の座標 m_{ct} において求められる．この線形近似は，$x_t^{[k]}, \mu_{ct,t-1}^{[k]}$ において h の接線となっている．この近似の下で，特徴 c_t の位置に対する事後推定はガウス分布で表現できる．新たな平均と共分散は，普通の EKF の計測更新：

$$K_t^{[k]} = \overline{\sum}_{ct,t-1}^{[k]} H_t^{[k]T} (H_t^{[k]} \overline{\sum}_{ct,t-1}^{[k]} H_t^{[k]T} + Q_t)^{-1} \quad (2.6.31)$$

$$\mu_{ct,t}^{[k]} = \mu_{ct,t-1}^{[k]} + K_t^{[k]} (z_t - \hat{z}_t^{[k]}) \quad (2.6.32)$$

$$\sum_{ct,t}^{[k]} = (I - K_t^{[k]} H_t^{[k]}) \sum_{ct,t-1}^{[k]} \quad (2.6.33)$$

で得られる．ステップ 1 と 2 はパーティクルの数 M だけ繰り返され，M 個のパーティクルを含んだ仮のセットが生成される．

3. **リサンプリング**

最後ステップとして，FsatSLAM はこのパーティクルセットからリサンプリングを行う．リサンプリングについて，FastSLAM では，仮のパーティクルセットから(置き換えによって)，新たなパーティクルセットを作成する．この時，すでに定義した重みの大きさに従って，仮のセットからパーティクルがサンプリングされる．得られた M 個のパーティクルセットは，正式なセット Y_t となる．リサンプリングが必要なのは，仮のセット中のパーティクルが，求めたい事後推定に従わずに分布しているからである．従っていない理由は，ステップ 1 では最近の制御 u_t のみに従って姿勢 x_t が生成されており，計測 z_t が反映されていないからである．パーティクルフィルタにおいて，リサンプリングはこのようなミスマッチを修正するための共通の方法である．

このようなミスマッチについて，一次元の例を図 2.6.6 に示す．ここでは破線が**提**

案分布を表している．パーティクルは提案分布か生成されている．実際は目標分布を表している．FastSLAM では，z_t は提案分布とは無関係で，目標分布に反映される．図 2.6.6 の下に描かれているようにパーティクルに重み付けをして，この重みに基づいてリサンプリングすることで，新たなパーティクルセットが目標分布を近似するようになる．

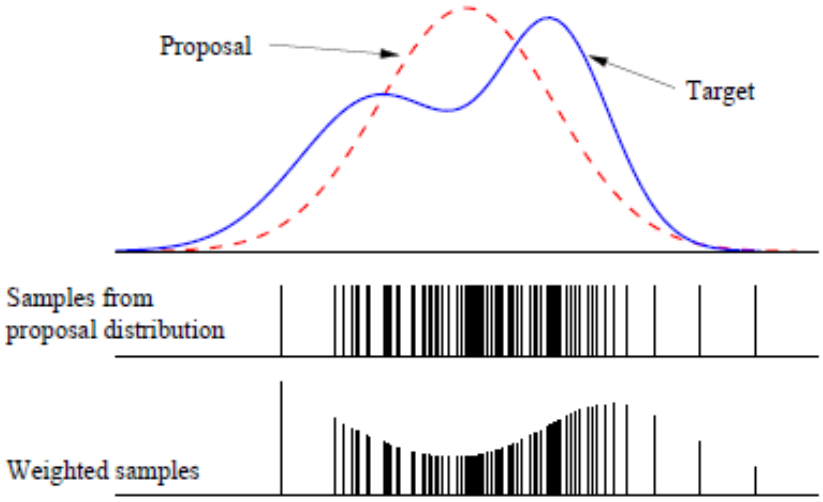


図 2.6.6　リサンプリングのミスマッチ

重みの決定は，仮のセットが示すロボットの奇跡の提案分布を計算すると行える．Y_{t-1} 中で軌跡を表すパーティクルセットが $p(x_{1:t-1} \mid z_{1:t-1}, u_{1:t-1}, c_{1:t-1})$ に従って分布していると仮定するとき，仮のセット中で軌跡を表すパーティクルは，

$$p(x_{1:t}^{[k]} \mid z_{1:t-1}, u_{1:t-1}, c_{1:t-1}) = p(x_t^{[k]} \mid x_{t-1}^{[k]}, u_t)\, p(x_{1:t-1}^{[k]} \mid z_{1:t-1}, u_{1:t-1}, c_{1:t-1}) \quad (2.6.34)$$

に従って分布する．因子 $p(x_t^{[k]} \mid x_{t-1}^{[k]}, u_t)$ は，式(2.6.25)で用いられてるサンプリングのための分布である．

目標分布は計測 z_t，対応 c_t を考慮したものである．すなわち，

$$p(x_{1:t}^{[k]} \mid z_{1:t}, u_{1:t}, c_{1:t}) \quad (2.6.35)$$

と表される．リサンプリングの過程では，目標分布と提案分布の違いが計算される．前述のように，リサンプリングのための**重み**は目標分布を提案分布で割って，

$$w_t^{[k]} = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{p(x_{1:t}^{[k]} \mid z_{1:t}, u_{1:t}, c_{1:t})}{p(x_{1:t}^{[k]} \mid z_{1:t-1}, u_{1:t-1}, c_{1:t-1})} \quad (2.6.36)$$

$$= \eta p(z_t \mid x_t^{[k]}, c_t)$$

で得られる．最後の変形式は，次の変換：

$$w_t^{[k]} = \eta \int p(z_t \mid m_{ct}, x_t^{[k]}, c_t)\, p(m_{ct} \mid x_t^{[k]}, c_t)\, dm_{ct}$$

$$= \eta \int p(z_t \mid m_{ct}, x_t^{[k]}, c_t) \underbrace{p(m_{ct} \mid x_{1:t-1}^{[k]}, z_{1:t-1}, c_{1:t-1})}_{\sim N(\mu_{ct,t-1}^{[k]}, \Sigma_{ct,t-1}^{[k]})} dm_{ct} \quad (2.6.37)$$

と等価である．ここではセンサ計測値の予測に無関係な変数は省略されている．$N(x; \mu, \Sigma)$ は変数 x に対する平均 μ，共分散 Σ のガウス分布を表す．

式(2.6.37)の積分は，時刻 t において観測された特徴の位置の推定値，そして計測モデルが含まれている．式(2.6.37)を計算するために，FastSLAM ではステップ 2 の計測更新で行われたものと同じ線形近似が用いられる．

$$w_t^{[k]} \approx \eta \,|\, 2\pi Q_t^{[k]} \,|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_t^{[k]})Q_t^{[k]-1}(z_t - \hat{z}_t^{[k]})\right\} \qquad (2.6.38)$$

で与えられる．ここで $Q_t^{[k]}$ は共分散で，

$Q_t^{[k]} = H_t^{[k]T} \sum_{n,t-1}^{[k]} H_t^{[k]} + Q_t$ （2.6.39）

と計算される．この式は，実測値 z_t がガウス分布に基づいてばらつくときの確率分布を表す．この分布は，式(2.6.37)の積分に，h の線形近似を反映して得られたものである．このようにして得られる重みは，仮のパーティクルセットから新たに M 個のパーティクルをサンプリングするために用いられる．このリサンプリング過程によって，パーティクルは計測の確率に比例して生き残るようになる．

対応関係が既知の SLAM 問題に対する FastSLAM アルゴリズムの更新則は，これらの 3 ステップによって構成される．この更新の実行時間は，軌跡の長さに影響を受けない．時刻 t のパーティクルを新たに生成するとき，最後の姿勢 $x_{t-1}^{[k]}$ しか用いられないからである．つまり，過去の姿勢は問題なく捨てることが出来る．したがって計算時間もメモリ量も，FastSLAM では経過時間の長さに依存しないという魅力的な結論が得られる．

図 2.6.7 に対応関係が既知の場合の FastSLAM アルゴリズムをまとめたものも示す．単純化のため，この実装は各時刻において一つだけ特徴が計測されることを仮定している．この実装は比較的素直なものであり，前述の各ステップが実装されている．実際，この FastSLAM は最も実装が簡単な SLAM アルゴリズムの一つで，これを **FastSLAM 1.0** と呼ぶ．

FastSLAM 1.0 では姿勢が制御 u_t だけからサンプリングされ，重みの計算のために計測 z_t が用いられた．このことは，制御の正確さが，ロボットのセンサの正確さよりも比較的よくないときに問題となる．その例を図 2.6.8 に示す．この例では，提案分布によって (a) のように広い領域にパーティクルが生成されている．しかし，楕円で示すように，計測によって，最尤が高いままであるパーティクルの割合が小さくなっている．リサンプリングの後では，楕円の中のパーティクルしか「生き残れない」．**FastSLAM 2.0** では，この問題を回避するために，u_t だけでなく z_t も用いてサンプリングを行う．その結果，FastSLAM 2.0 は FastSLAM 1.0 よりも効率が良くなる．そのかわり，FastSLAM 2.0 の実装は FastSLAM 1.0 の実装よりも難しくなり，導出も複雑なものとなる．

1: **Algorithm Fast_SLAM** (z_t, c_t, u_t, Y_{t-1}):
2: for $k = 1$ to M do
3: retrieve $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \sum_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \sum_{N,t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}
4: $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$
5: $j = c_t$
6: if feature j never seen before
7: $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$
8: $H = h'(x_t^{[k]}, \mu_{j,t}^{[k]}$
9: $\sum_{j,t}^{[k]} = H^{-1} Q_t (H^{-1})^T$
10: $w^{[k]} = p_0$
11: else
12: $\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$
13: $H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$
14: $Q = H \sum_{j,t-1}^{[k]} H^T + Q_t$
15: $K = \sum_{j,t-1}^{[k]} H^T Q^{-1}$
16: $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$
17: $\sum_{j,t}^{[k]} = (I - KH) \sum_{j,t-1}^{[k]}$
18: $w^{[k]} = \mid 2\pi Q \mid^{\frac{1}{2}} \exp\left\{ -\frac{1}{2} (z_t - \hat{z}_n)^T Q^{-1} (z_t - \hat{z}_n) \right\}$
19: endif
20: for all other features $j' \neq j$ do
21: $\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$
22: $\sum_{j',t}^{[k]} = \sum_{j',t-1}^{[k]}$
23: endfor
24: endfor
25: $Y_t = \emptyset$
26: do M times
27: draw random k **with probability** $\propto$ $w^{[k]}$
28: add $\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \sum_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_N^{[k]}, \sum_N^{[k]} \right\rangle \right\rangle$ to Y_t
29: endfor
30: return Y_t

図 2.6.7　FastSLAM のアルゴリズム

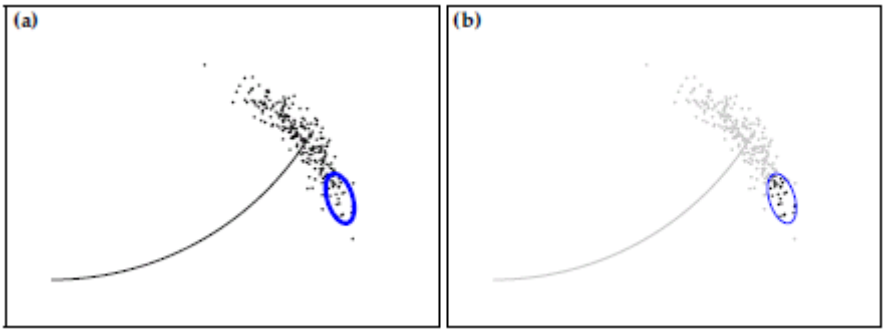


図 2.6.8　提案分布と目標分布のミスマッチ

第3章　提案システム

3.1. 提案システム

3.1.1. 提案システムの概要

既存の位置情報推定技術は，高密度にセンサ端末を配置するなど，特別なインフラ設備が必要であり，また，多くの位置が既知であるビーコン端末を必要とするため，位置推定システムの構築のために，高いコストや労力が必要となるという問題がある．こうした問題が，個人ユースでの利用を難しくさせてしまっている．そこで本研究は，特別なインフラ設備を必要とせず，プラットフォームとして市販のセンサノードと移動ロボットを用いた位置情報管理システムの検討を行う．

提案システムは，位置の既知なビーコン端末を必要とせず，自動または遠隔操作可能な廉価ホームロボットのみで構築可能なシステムであり，対象となるセンサは簡易な通信機構があればよいものとなっている．移動ロボットが，自動または遠隔操作で移動し，能動的に対象となるセンサの位置を推定し，それらをデータベースとして管理しておくことで，位置の変わらない静的な物体から，人や動物など，動的に位置が変化するものまで，その存在の有無や数量，実際の位置までを把握することができる．

このシステムは用途として，人や動物や物品の監視・保護・管理，また，人が立ち入ることが出来ない，災害現場の環境把握，洞窟や海底の調査などが考えられる．

3.1.2. 提案システムの構成

提案システムは以下の三つのシステムにより構成されており，図 3.1.1 に簡単な構成図を示す．

- ロボットセンサネットワーク
- ロケーション管理サーバ
- ユーザ

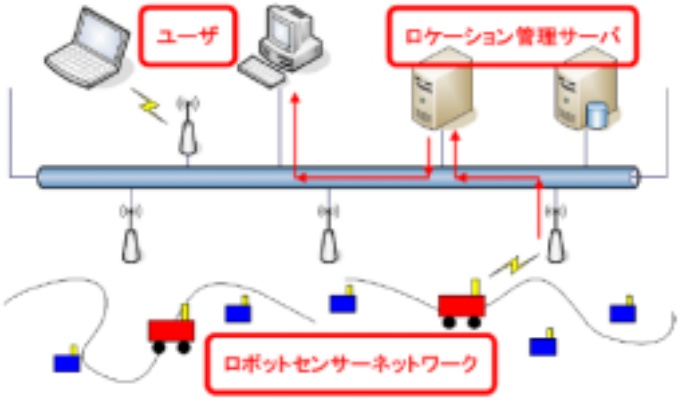


図 3.1.1　提案システムの構成

1．ロボットセンサネットワークの役割は以下のように分けられる．
   - 移動ロボットによるセンシング
   - 移動ロボットによるロケーション管理サーバへの取得データを送信
   - ロボットの自己位置推定

2．ロケーション管理サーバは以下の役割に分けられる．
   - 位置推定

・位置推定情報の保持・更新
3．ユーザは以下のサービスが利用できる．
・位置情報取得
・コンテキストアウェアサービスの利用

3.2. 実装環境への適用

実装するシステムの概要を表したものを図 3.2.1 に示す．実装するシステムは，ランドマーク（ターゲットセンサ）と，移動ロボットと，ロケーション管理サーバによって構成されている．前述で提案したシステムのうち，ロボットセンサネットワークはターゲットセンサと移動ロボットにより構成されている．以下に実装環境の詳細を述べる．

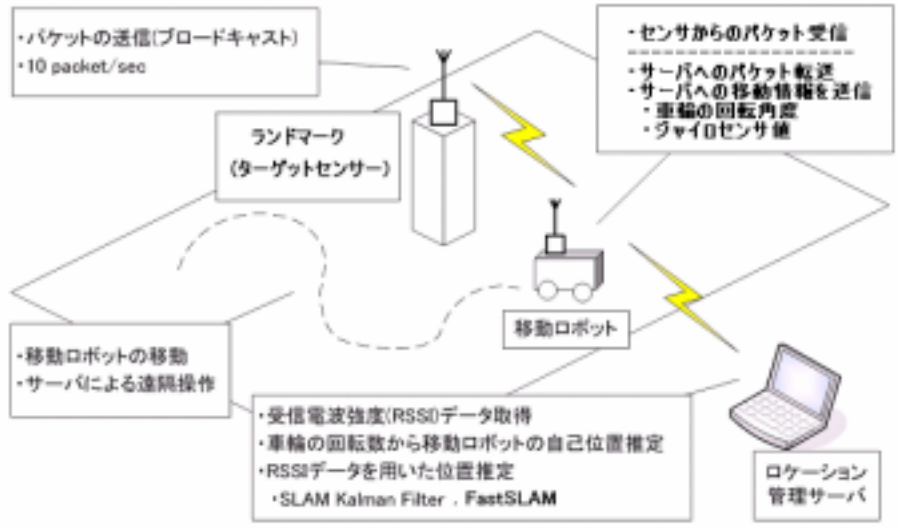


図 3.2.1　実装環境の構成

3.2.1. ランドマーク

2.5 節で述べたように，人工ビーコンもランドマークとして扱う事ができる．ここでのランドマークとは 2.1 節で示した無線センサ端末 Micaz MOTE の事である．以後，このランドマークをターゲットセンサと呼ぶ．このターゲットセンサの動作は以下のようなものである．

・ターゲットセンサの TinyOS パケットのブロードキャスト(微弱無線)
・送信間隔は 10packet/sec

つまり，ターゲットセンサの動作は，単純に，連続したパケット送信を行い続けるものである．このパケットには，MOTE が環境をセンシングした情報を含むことも出来る．

3.2.2. 移動ロボット

移動ロボットの設計として, 対向二輪駆動型(1 ボールキャスタ)を採用した. この構成は，二つの車輪で，それぞれがモーター駆動しており，加えて，安定させるためにキャスタとして，ボールキャスタを用いるものである．なので，当然，移動ロボットの移動制御は二つのモーターのみでの制御となる．今回，このような構成を採用したのは，まず作ることが容易であることや，自由度が高いこと，ロボット制御の数学も容易であることからである(2.4 節参照)．しかし，ボールキャスタは車輪キャスタよりは地面に対する摩擦が強いため，その取り付け方などは工夫しなければならない．本実験においては，実験器具の取り付け場所，ボールキャスタの高さを調節するなど，なるべくボールキャスタに摩擦抵抗が加わらないに重心の設計をしてある. このように設計された移動ロボットを図 3.2.2 に示す.

この移動ロボットには，Lego Mindstorms NXT 上に，ターゲットセンサと同等の MOTE センサ, 移動基地局としての stargate, さらには既存の位置推定手法との比較のために GPS も積載させている．

この移動ロボットのシステム上の機能は，ターゲットセンサとロケーション管理サーバの移動ゲートウェイサーバとなるものである．この移動ロボットの動作は以下のようなものである．

・ 遠隔操作による移動制御(Bluetooth Class II V2.0)

・ MOTE センサによる，ターゲットセンサからの TinyOS(TOS) パケット受信(ZigBee 2.4GHz)

・ stargate(移動ロボット）による，ロケーション管理サーバへの TOS パケット転送(無線 LAN 802.11b)

・ Lego Mindstorms NXT(移動ロボット）によるロケーション管理サーバへの以下 2 つのセンサから得られる移動動作情報の送信(Bluetooth Class II V2.0)

- Rotation Sensor：両左右車輪の回転角度
- NXT Gyro Sensor：両車輪中央に取り付けられたジャイロセンサ値

この際に, Lego Mindstorms NXT からロケーション管理サーバへ送信される動作情報は, ロケーション管理サーバが, 移動ロボットの自己位置推定（デッドレコニング）を行う際に用いるデータである．2.4 節のように，この移動ロボットの自己位置を推定する際には，移動ロボットの車輪の直径 2r と車輪間の長さ 2d が必要であるため，その設定を表 3-1 に示す．

図 3.2.2 移動ロボット

表 3-1 移動ロボットの車体設定

	長さ (cm)
車輪間の距離	16
車輪の直径	8.3

3.2.3. ロケーション管理サーバ

今回の実験において用いたロケーション管理サーバは，図 3.2.3 に示す，ノート PC を用いた．また，表 3-2 に，おおまかな仕様を示す．このロケーション管理サーバの動作は以下のようなものである．

・ 遠隔操作による移動ロボットの移動動作制御

・ 移動ロボットからの，移動動作情報（上記の二つのセンサ値）の受信(Bluetooth Class II V2.0)

- 移動ロボットの自己位置推定(デッドレコニング)

・ stargate(移動ロボット）からの TOS パケットパケットの受信(無線 LAN 802.11b)

- TOS パケットから得られた受信電波強度(RSSI 値) データを用いてノード位置推定

- EKF-SLAM によるノード位置推定
- FastSLAM によるノード位置推定

つまり，ロケーション管理サーバは，移動ロボットを構成する小型基地局の stargate との無線 LAN 通信と，同じく移動ロボットを構成する Lego Mindstorms NXT との Bluetooth 通信とを同時に行う．将来的には，移動ロボットの自己位置推定と，ターゲットセンサの位置推定をも並行して行う．

図 3.2.3　ロケーション管理サーバ

表 3-2　ロケーション管理サーバの仕様

製品名	NEC LaVie LJ700/E
OS	Windows XP service pack 2
プロセッサ	PentiumR M 597MHz
通信	802.11b, Bluetooth Class II V2.0 準拠

3.2.4. GPS

先ほど既存の位置推定手法との精度比較のために GPS を移動ロボットに搭載させたことは述べた．ここでの既存の位置推定手法とは，前述のように TOA 測定を指す．また，現在最も位置情報取得サービスが利用されている GPS と精度比較することは，本手法をコンテキスト・アウェアなサービスに利用するためのイメージがしやすい．

本研究においては，移動ロボットの負荷にならないように，Wintec 社が開発した小型の GPS ロガーを採用した（図 3.2.4 参照）．この GPS ロガーの製品仕様を表 3-3，特徴を以下に示す．

・同クラス最小，防水
  - サイズ: 60×38×16mm　重さ: 41g

・Google Maps, Google Earth などにログデータ表示可能
・ログ記録可能 Waypoint 数 130,000 点以上
・USB，Bluetooth でリアルタイムナビゲーション可能

図 3.2.4　GPS ロガー

表 3-3　GPS ロガーの製品仕様

製品名	WBT-201
GPSチップ	超高感度Atmel-ublox ATR 0625 (-158dBm) 採用
位置精度	2.5m CEP (単独測位)
NMEA	NMEA0183 2.3 GGA,RMC, GSA, GSVに対応

また, 前述のように移動ロボットである Mindstorms NXT の Rotation Sensor と NXT Gyro Sensor の値からデッドレコニングを求められる事と同様に，GPS も緯度・経度からデッドレコニングを求めることができる．以下に緯度・経度から方位と距離を求める計算式を示す.

経度をλ，緯度をφとすると，距離を求める 2 点の座標はそれぞれ，$(\lambda_1, \phi_1), (\lambda_2, \phi_2)$ と示す事ができる.

$\Delta\lambda = \lambda_2 - \lambda_1$　（3.2.1）

$\Delta\phi = \phi_2 - \phi_1$　（3.2.2）

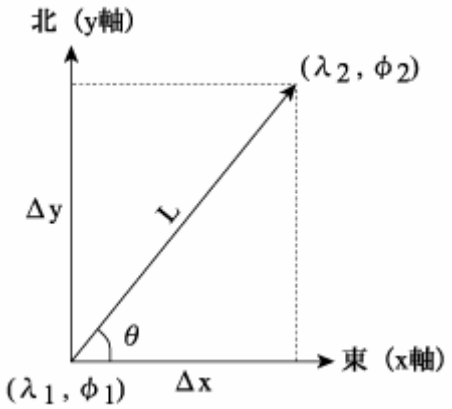


図 3.2.5　緯度・経度から距離・方位の求め方

まず，緯度方向の変位Δy については地球を完全な球体と仮定した場合，緯度の変化に対する円弧の長さで計算できる．ここで，地球の半径として赤道半径を使用すると，

地球の赤道半径　A=6378137 [m]　（3.2.3）

となり，y 軸方位の変位は,

$\Delta y = A\Delta\phi$　（3.2.4）

となる.

一方, 経度方向の変化Δx についても同様に経度の変化に対する円弧の長さで計算できる

が，この場合，緯度によって円の半径が変化することを考えなければならない．緯線に沿って地球をスライスしたとすると，その切り口の半径は緯度の余弦に比例するので，

$$\Delta x = A\Delta\lambda\cos\phi_1 \qquad (3.2.5)$$

となる．ここで代表点としてϕ_1を用いたが，ϕ_2としても実用上は同じことである．

２点間の距離 L は三平方の定理により，

$$L = \sqrt{\Delta x^2 + \Delta y^2} \qquad (3.2.6)$$

また，方位角 θ は，

$$\theta = \tan^{-1}(\Delta y / \Delta x) \qquad (3.2.7)$$

となる．このとき方位角の基準は真東の方角を 0 度とする．

実際には地球は完全な球体ではないため，広範囲間（沖縄～北海道間など）の距離と方位を求める場合などでは適用できないが，GPS ログのポイント間の距離など，ごく小さい範囲での移動では問題ない．もし，広範囲間の距離と方位を求める場合には，球面三角法を用いて計算する必要がある．

第4章　実装実験

4.1. 実験概要

ここでは，第 3 章で説明したシステムを実環境へ適用し，実装実験を行う．本実験では，屋外と屋内において，それぞれ二つのシナリオを用意した．それぞれの実験で得られた各結果を評価し，最終的にはノード位置推定の結果についてエラーメトリックを用いて解析する．以下に，その実験内容の詳細と評価の手順・方法を説明する．

4.1.1. 実験場所

まず始めに，実験場所と提案システムへの影響について簡単に説明する．

１．屋外実験

屋外での実験場所（図 4.1.1 参照）は，建物が隣接している広い空間である．四方をコンクリート材質の建物に囲まれているとはいえ，見通しがよく，特に目立つ障害物はない．地面の状況は，図 4.1.2 に示す通り，石材質の硬いアスファルトであり，造りが凸凹している．この材質は摩擦が大きく，図 3.2.2 に示したロボットの移動にとっていい環境ではない事が想像つく．また，移動ロボットの行路は必ずしも平面というわけではない．

図 4.1.1　屋外空間

図 4.1.2　屋外の地面

２．屋内実験

屋内での実験場所（図 4.1.3 参照）は，コンクリートの建物内部の広い空間である．コンクリート材質柱や壁，椅子など障害物などがあり，あまり見通しは良くない．地面の状況は，図 4.1.4 に示す通り，土砂材質の滑らかなタイルである．タイルとタイルのつなぎ目が多少凸凹している．屋外に比べて，地面の摩擦が小さく，移動ロボットにとって，スムーズな移動制御ができる．また，移動ロボットの行路は，ほぼ平面となっている．実験では，移動ロボットが柱や椅子などの障害物を気にならないように行路を設定した．

図 4.1.3　屋内空間

図 4.1.4　屋内の地面

4.1.2. **実験シナリオ**

次に，実験シナリオについて説明する．一般的に受信する電波(RSSI 値)は，互いの距離が近いときは安定（分散が小さい）しており，離れているときは不安定（分散が大きい）になる，という特性を持っている．この特性を利用し，本実験では以下の二つのシナリオを用意し，ノード位置推定の精度にどのように影響を与えるのかを評価する．

１．実験シナリオ 1 (図 4.1.5)：
　　移動ノードと固定ノードの受信電波の強度を，**極端**に強弱の差をつけた場合

２．実験シナリオ 2 (図 4.1.6)：
　　移動ノードと固定ノードの受信電波の強度を，**均等**に強弱の差をつけた場合

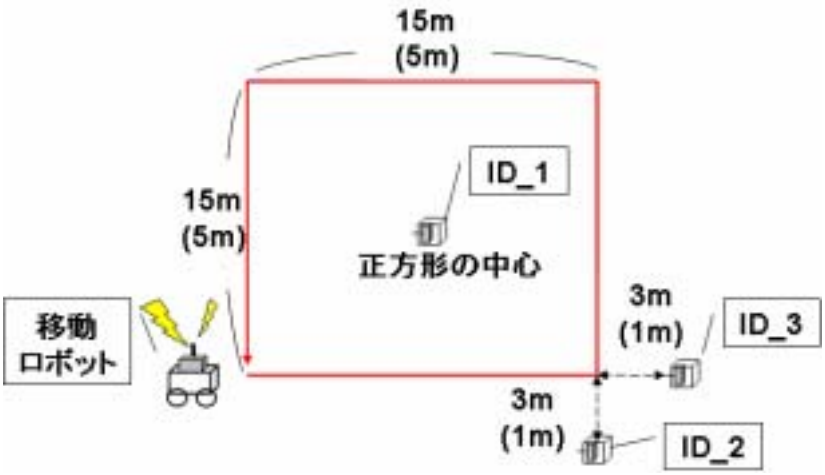


図 4.1.5　実験シナリオ 1　(括弧内数値：屋内実験環境)

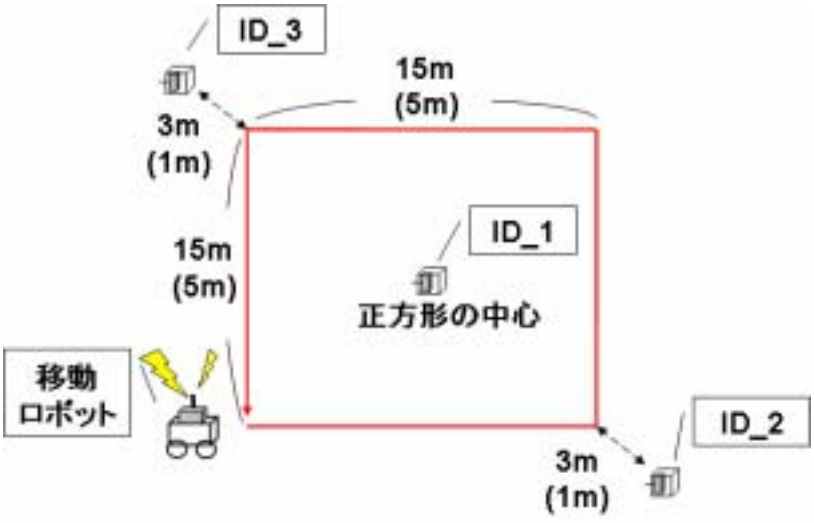


図 4.1.6　実験シナリオ 2　(括弧内数値：屋内実験環境)

ここで，図 4.1.5 と図 4.1.6 に示した各シナリオにおいて，移動ノードの移動範囲と固定ノードの設定範囲の括弧内の数値は屋内実験についての設定値である．この範囲の違いは，屋外と屋内において，それぞれ実用思考で設定した数値となっている．移動ノード(移動ロボット)には，遠隔操作で正方形の移動動作をするように行い，出発した場所と到着する場所が同じとなるようにしている．また，固定ノード（ターゲットセンサ）にはノード ID 番号を自由に定めることができ，どの ID 番号からパケットを受信してきたのかが分かる仕組

みとなっている．本研究では固定ノードを三つ用意し，それぞれ，ID_1，ID_2 ，ID_3 と設定した．

4.1.3. 実験評価の流れ

以下に，実装評価への流れを説明する．

① 屋内外において，それぞれ二つのシナリオ環境で実験を行う．詳細は上記の通りである．

② 各固定ノードから得られた受信電波強度（RSSI 値）の結果についてモデル化を行う．

③ 以下三つのセンサから得られた移動情報から移動精度（デッドレコニング）を比較
・車輪(ローテーション)，ジャイロ，GPS(屋外のみ有効)

④ 以下の SLAM アルゴリズムを用いてノード位置推定
・EKF-SLAM，FastSLAM

⑤ 次節で説明する以下の三つエラーメトリックで解析
・Cross-Track Error，　Along-Track Error，　Node Error

4.1.4. エラーメトリックについて

上記のエラーメトリックの説明を以下に示す．

1．Cross-Track Error
設定した行路からの移動ノードの**左右横**のずれとなる．

2．Along-Track Error
設定した行路からの移動ノードの**前後縦**のずれとなる．

3．Node Error
設定したターゲットセンサの位置（既知）と最終的に得られた位置（推定）とのずれとなる．

4.2. 屋外実験

上記の屋外場所で，図 4.1.5 と図 4.1.6 の二つのシナリオについて実験し，得られた実験結果について，4.1.4 項で示した手順で，以下に評価・考察を示す．

4.2.1. RSSI 値の評価

ここでは，2.5 節で示したように，移動ロボットの知覚に当たる環境計測モデルについて示す．

2.5 節で距離と受信電波強度（RSSI 値）の関係性を示したように，RSSI 観測値は，移動ロボットとターゲットセンサの距離の関係性を導き出すことができる．ただし，図 2.5.2 で示した関係性は自由空間の場合のみのモデルに限定される．実装では，場所や天候，人通り，その他ターゲットセンサの調子までもが環境計測に影響を与えるので，図 2.5.2 のモデルを実装に適用することはできない．つまり，RSSI 観測値と実際のターゲットセンサとの距離の関係性を結び付けるには，実験毎回において，各ノード間（移動ノード‐固定ノード間）のキャリブレーション作業という手間が必要となる．その労力はノード数を増やすごとに大きくなってしまうため，将来的には，無数の環境計測モデルサンプルから適切な環境計測モデルを選ばせるようなシステムにしたい．

シナリオ 1，シナリオ 2 で得られた観測結果を，それぞれ図 4.2.1，図 4.2.2 に示す．横軸は実験経過時間を示し，縦軸はその時間に得られた RSSI 値を示している．

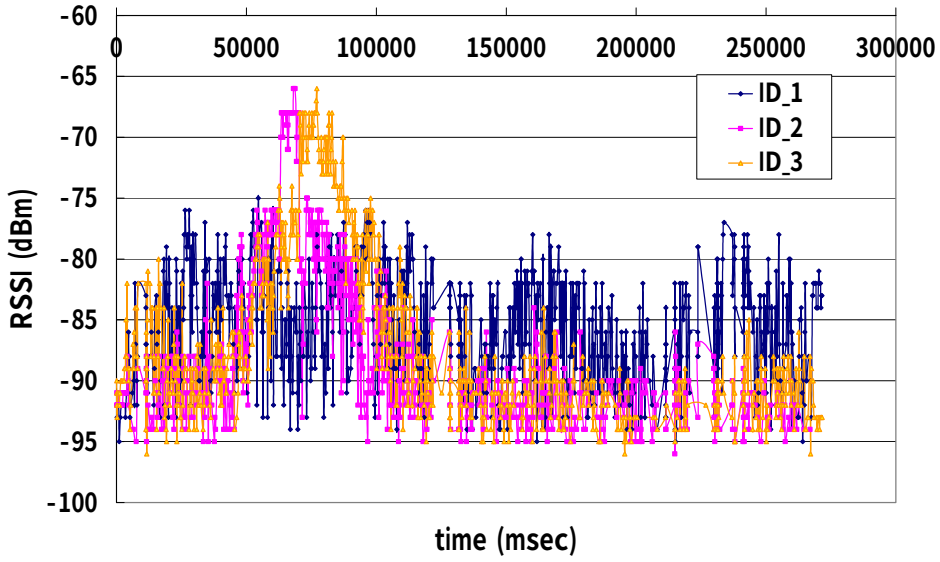


図 4.2.1　屋外での環境計測（シナリオ 1）

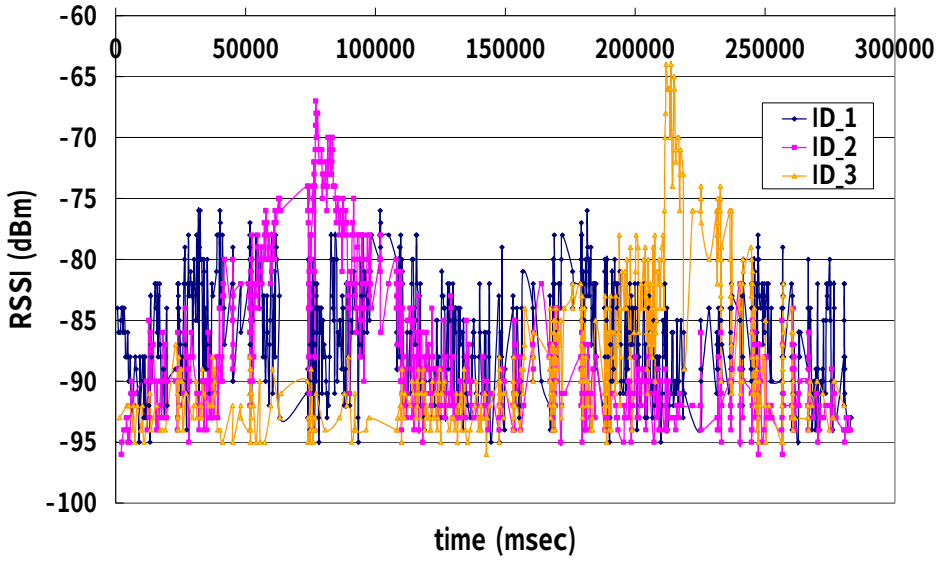


図 4.2.2　屋外での環境計測（シナリオ 2）

図 4.2.1 において，角固定ノード ID_1 では時間 60 [sec] あたりで RSSI 値が一番ピークとなり，ID_2 では 75 [sec] あたりでピークとなっている．このピークの時間は，移動ロボットが最初の角を曲がる時間に相当する．ID_3 においては他のノードより目立ったピークはないが，波打つように変化が表れている．図 4.2.2 も同様にして，実験の経過を見ることが出来る．本研究において，環境計測で重要なことは，，移動ノードと各固定ノードとの正確な距離を決定することではなく，距離との傾向（分散）を示すことにある．

次に，屋外実験におけるターゲットセンサと移動ロボットの電波到達距離とパケットの到達率（Packet Receive Rate, PRR）の関係性を図 4.2.3 に示す．

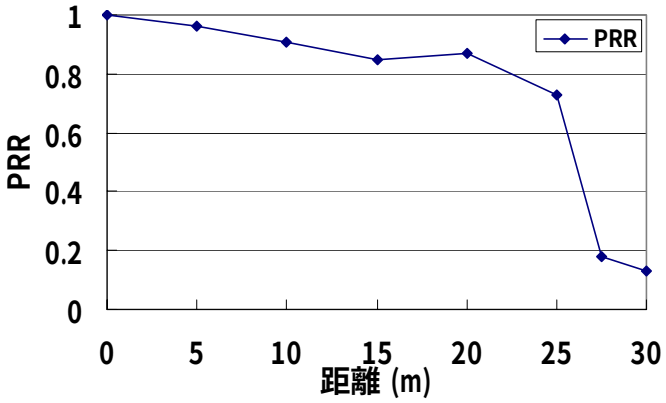


図 4.2.3　距離と PRR の関係性（屋外実験）

以上のことから，移動ロボットとターゲットセンサの距離が近い場合には，RSSI 値が強くなり，電波環境は安定する傾向がある，一方，距離が離れた場合は RSSI 値が弱くなり，電波環境は不安定となる傾向がある，ということが分かる．

4.2.2. 移動精度の比較

ここでは，2.4 節で示したような移動ロボットの動作ついて示し，移動精度(デッドレコニング)比較を行う．

シナリオ 1，シナリオ 2 で得られた移動動作の結果を，二次元座標を用いて，それぞれ図 4.2.4，図 4.2.5 に示す．

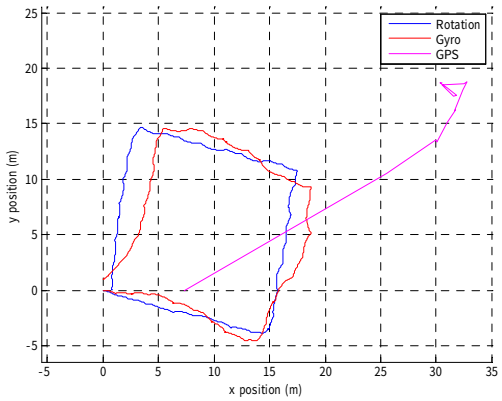


図 4.2.4　屋外の移動精度比較(シナリオ 1)

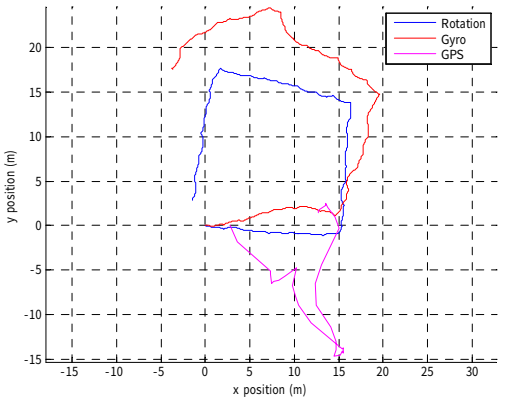


図 4.2.5　屋外の移動精度比較(シナリオ 2)

ここで，以下の各センサ値から 2.4 節で導出された式から算出してプロットしたもの．

- 青い線：移動ロボットの車輪に相当するローテーションセンサの値
- 赤い線：移動ロボットの両車輪の中央に取り付けられたジャイロセンサの値
- ピンクの線：移動ロボットの両車輪の中央に取り付けられた GPS の値(3.2.4 項参照)

前述の通り屋外での実験は，移動ロボットの行路は凸凹が目立ち，摩擦抵抗が大きい．そのため，予想では，車輪の回転角度値を用いることは好ましくなく，ジャイロセンサなどの道の良し悪しに関係なく動作を出力できるツールを用いることが好ましいと考えられた．しかし，上図を見て分かるとおり，両シナリオ共に，移動ロボットのローテーションセンサ(回転角度値)を用いた場合の移動精度が高いことが分かる．これは，3.2.2 項で示した移動ロボットの設計が良かったためだと考えられる．TOA 測定で知られる GPS に関して

は，15[m]×15[m]の移動範囲では移動測定ができていないことがわかる．
　以上のことから，本研究では，移動ロボットの車輪の回転角度値を用いることによって，動作制御の精度を高められることが分かる．

4.2.3. EKF SLAM の結果

　ここでは，上記の環境観測モデルと移動動作モデルを用いて2.6.1項で説明したEKF SLAMを導入する．本研究におけるEKF SLAMの実装コードはMATLABプログラムを用いて作成したものである．各シナリオにおいて，移動動作モデルをローテーションセンサの値とジャイロセンサの値を採用したものそれぞれについて以下にEKF SLAMの実行結果を示す．座標は二次元座標である．各図の見方を以下に示す．

- 青い線：移動ロボットの設定行路
- 緑の線：移動ロボットの移動動作モデルを示したもの(4.2.2 項と同様)
- 赤い線：EKF SLAM の自己位置推定結果
- 黒の＊：ターゲットセンサが実際に設置された場所
- 赤い▽：EKF SLAM で推定されたターゲットセンサの位置

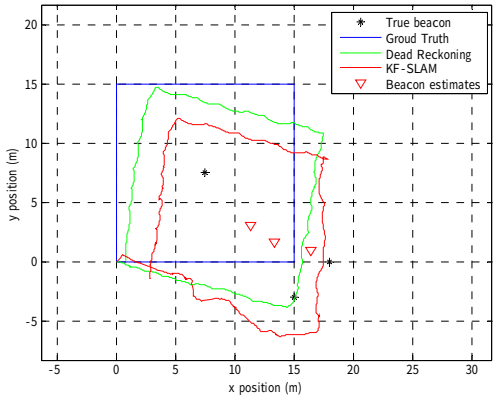


図 4.2.6　屋外シナリオ1のEKF SLAM
(移動動作モデル：Rotation Sensor)

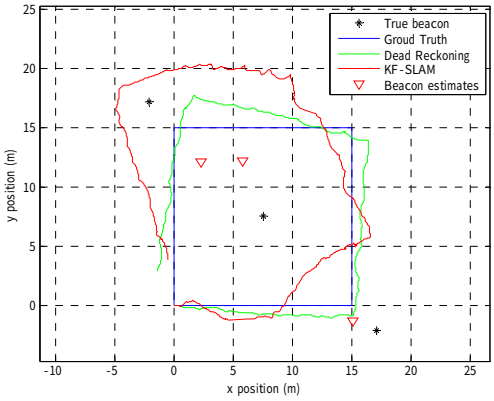


図 4.2.8　屋外シナリオ2のEKF SLAM
(移動動作モデル：Rotation Sensor)

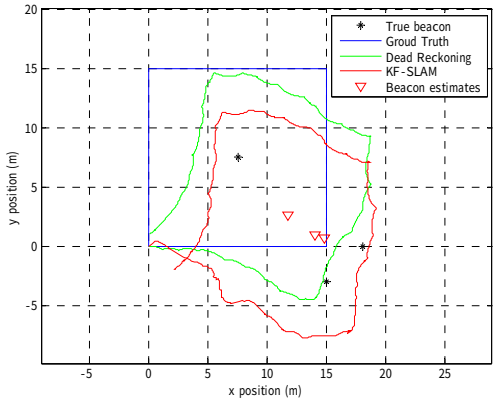


図 4.2.7　屋外シナリオ1のEKF SLAM
(移動動作モデル：Gyro Sensor)

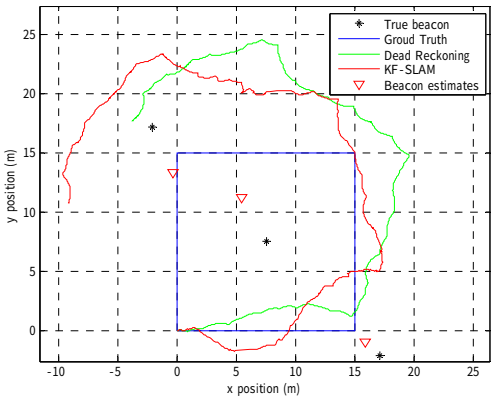


図 4.2.9　屋外シナリオ2のEKF SLAM
(移動動作モデル：Gyro Sensor)

4.2.4. FastSLAM の結果

　EKF SLAM と同様に，以下に FastSLAM の実行結果を示す．本研究では，FastSLAM2.0 (2.6.2 項参照) を用いて実装した．また、図 2.6.4 に示したように各パーティクルには，ロ

ボットの軌跡の推定値 $x_{1:t}^{[k]}$ と，地図中の各特長 m_j に対するカルマンフィルタが含まれている．そのため、ここでは全パーティクル数 M の平均を各時間でプロットしている。

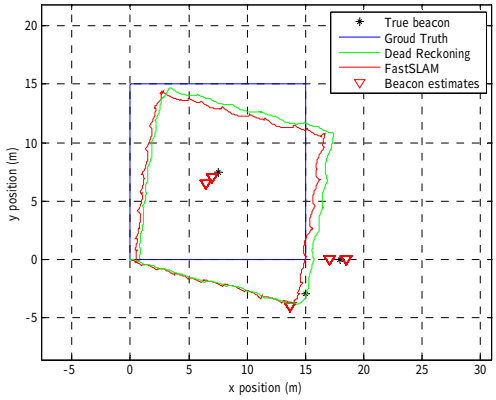


図 4.2.10　屋外シナリオ1のFastSLAM
(移動動作モデル：Rotation Sensor)

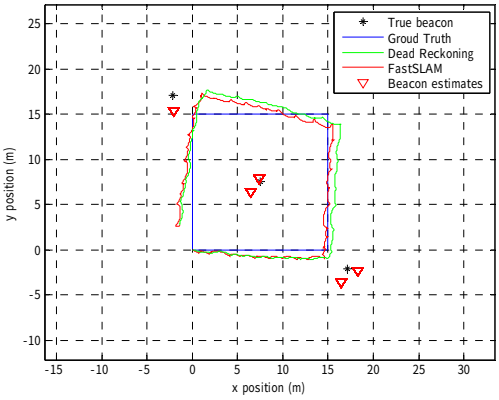


図 4.2.12　屋外シナリオ2のFastSLAM
(移動動作モデル：Rotation Sensor)

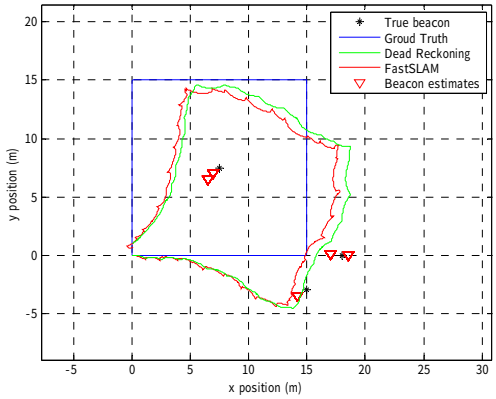


図 4.2.11　屋外シナリオ1のFastSLAM
(移動動作モデル：Gyro Sensor)

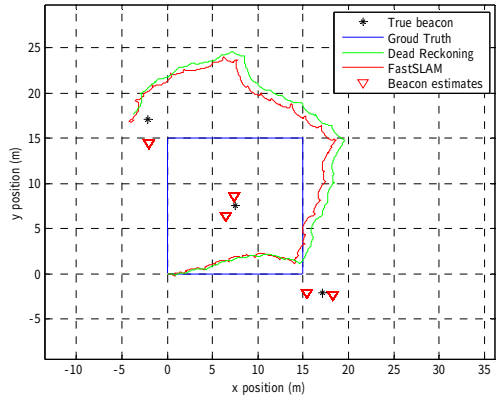


図 4.2.13　屋外シナリオ 2 のFastSLAM
(移動動作モデル：Gyro Sensor)

4.2.5. **エラーメトリック**

ここでは，屋外実験において得られた上記の動作モデル，EKF SLAM，FastSLAM の結果を 4.1.4 項で説明したエラーメトリックを用いて解析し，考察を行う．

1．Cross-Track Error

シナリオ 1 とシナリオ 2 において Cross-Track Error の平均値と最大値をそれぞれ図 4.2.14，図 4.2.15 に示す．また，標準偏差値を用いて解析した結果を図 4.2.16 に示す．

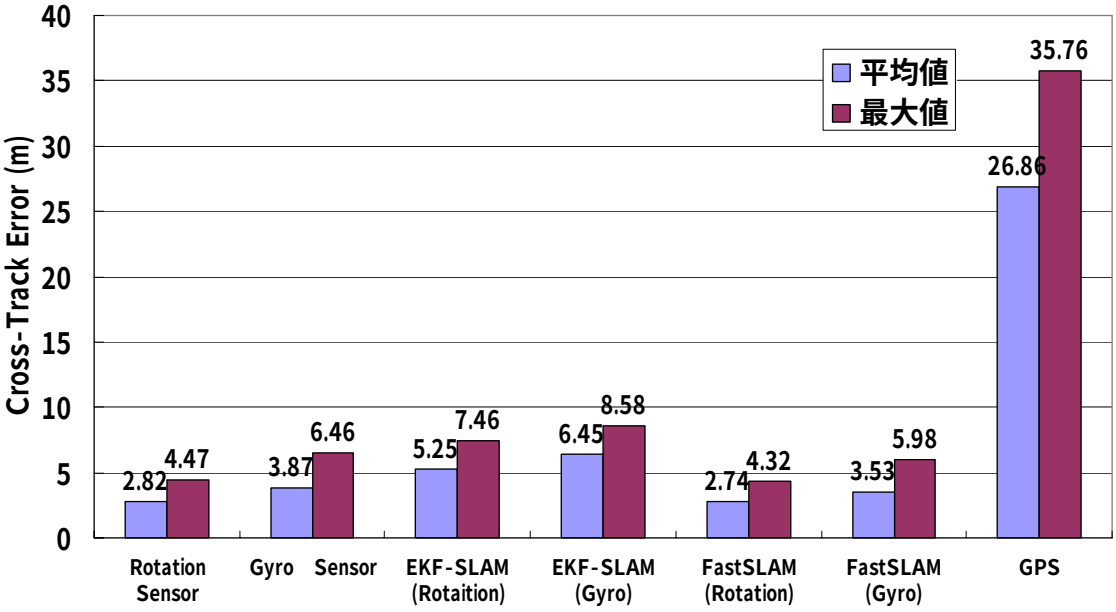


図 4.2.14　屋外シナリオ 1 の Cross-Track Error

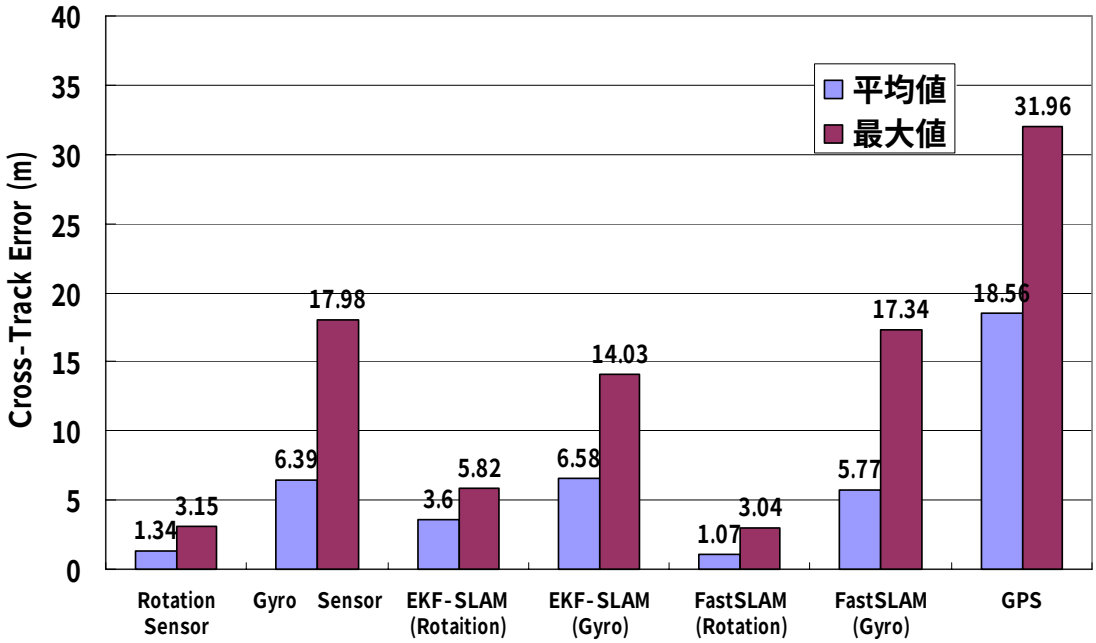


図 4.2.15　屋外シナリオ 2 の Cross-Track Error

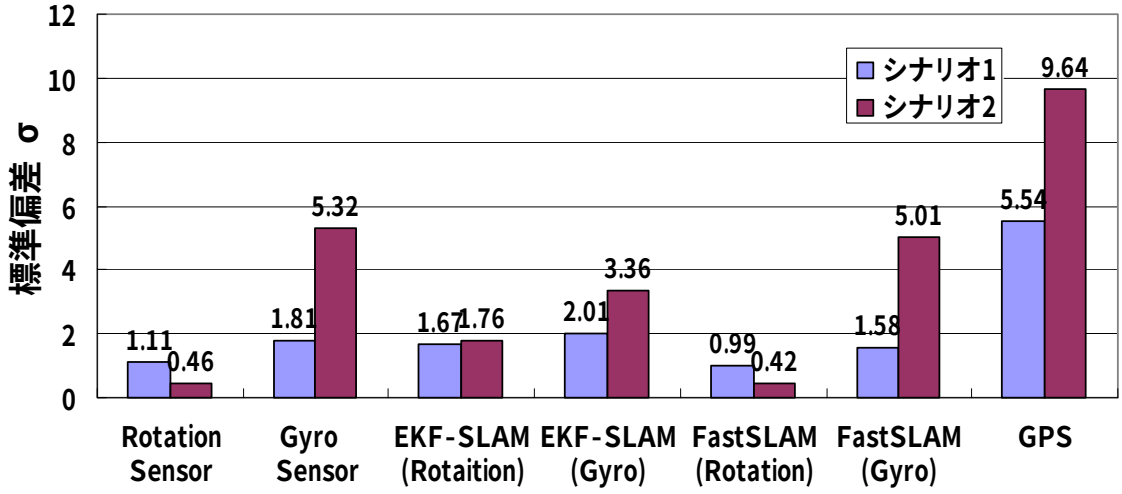


図 4.2.16　Cross-Track Error の標準偏差（屋外実験）

2．Along-Track Error

Cross-Track Error 同様に，Along-Track Error を図 4.2.17，図 4.2.18 に示す．また，標準偏差値を用いて解析した結果を図 4.2.19 に示す．

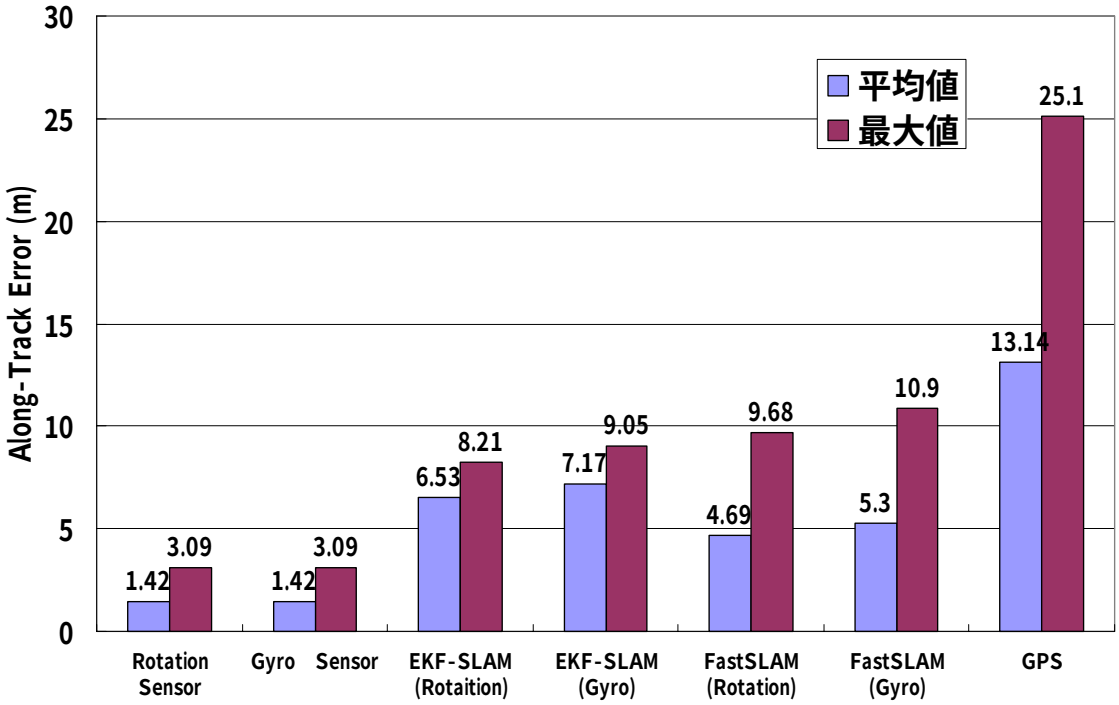


図 4.2.17　屋外シナリオ 1 の Along-Track Error

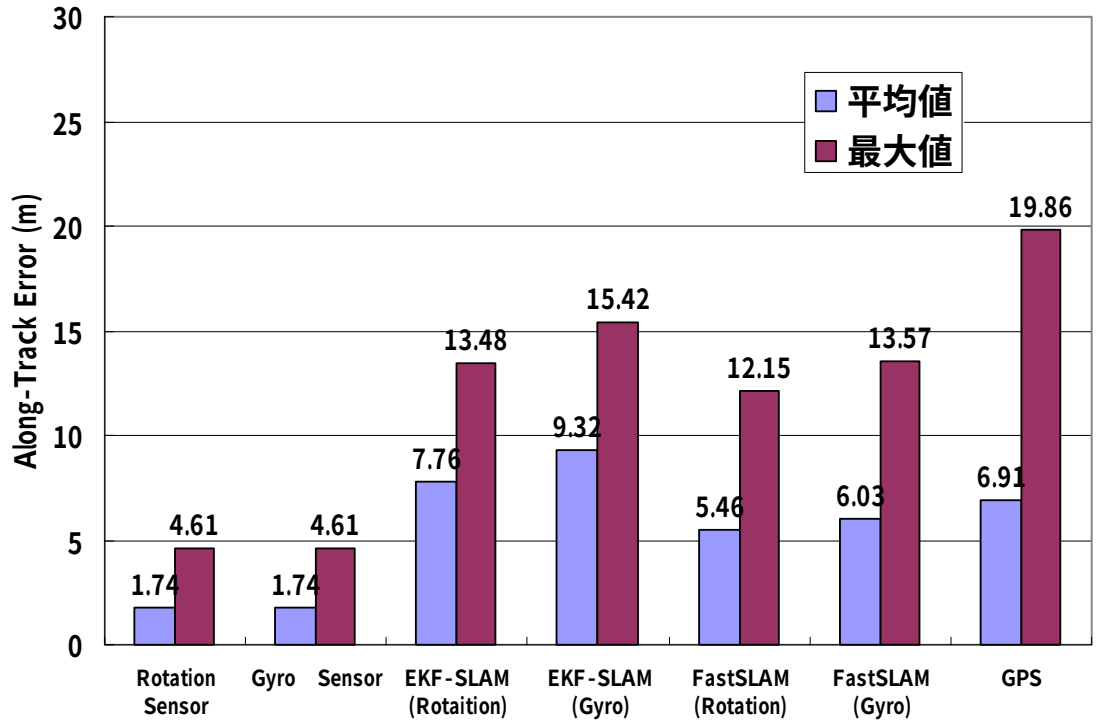


図 4.2.18　屋外シナリオ 2 の Along-Track Error

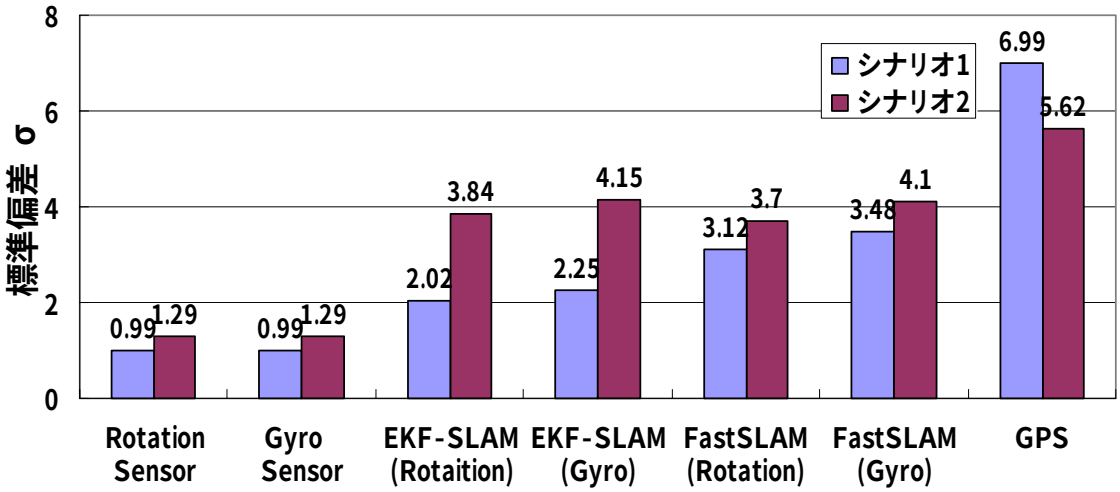
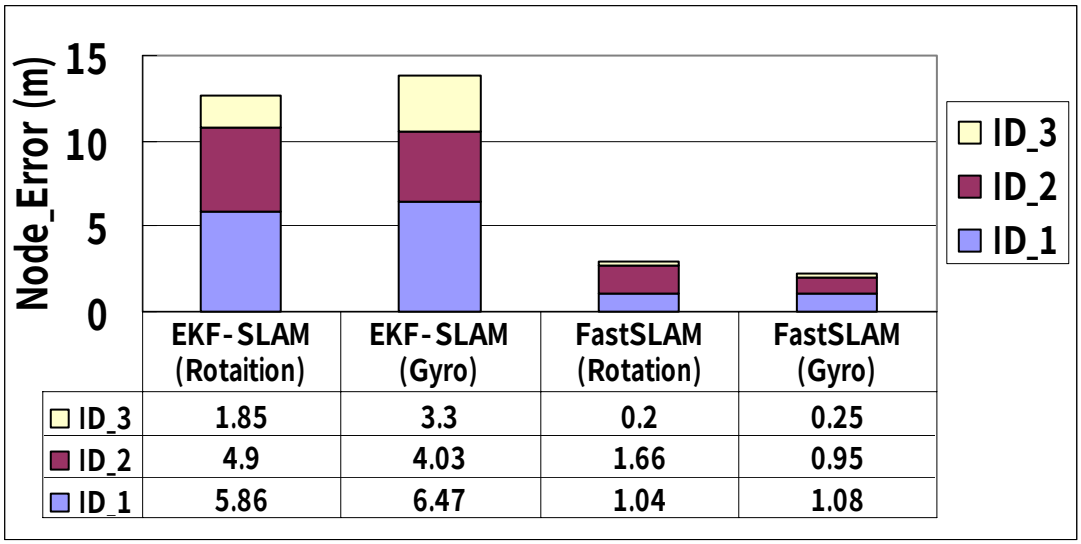


図 4.2.19　Along-Track Error の標準偏差（屋外実験）

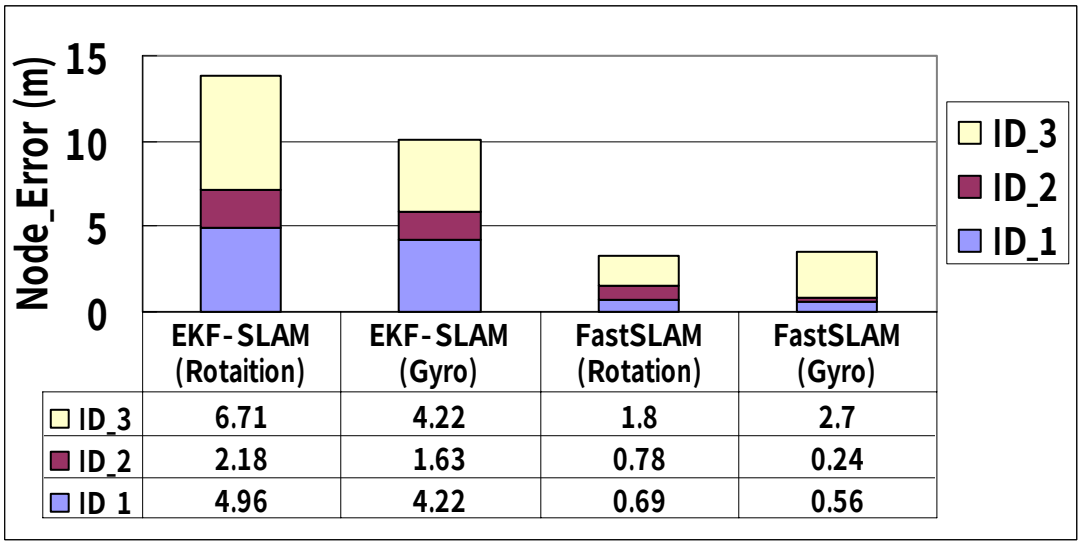
3．Node Error

シナリオ 1 とシナリオ 2 において Node Error をそれぞれ図 4.2.20，図 4.2.21 に示す.



	EKF-SLAM (Rotaition)	EKF-SLAM (Gyro)	FastSLAM (Rotation)	FastSLAM (Gyro)
ID_3	1.85	3.3	0.2	0.25
ID_2	4.9	4.03	1.66	0.95
ID_1	5.86	6.47	1.04	1.08

図 4.2.20　屋外シナリオ 1 の Node Error の標準偏差



	EKF-SLAM (Rotaition)	EKF-SLAM (Gyro)	FastSLAM (Rotation)	FastSLAM (Gyro)
ID_3	6.71	4.22	1.8	2.7
ID_2	2.18	1.63	0.78	0.24
ID_1	4.96	4.22	0.69	0.56

図 4.2.21　屋外シナリオ 2 の Node Error の標準偏差

4．考察

自己位置推定の精度においては，全般的に EKF SLAM よりも Fast SLAM のほうが精度を上げていることが分かる．しかし，Cross-Track Error に関して最も動作モデルが曖昧だったジャイロセンサでは，FastSLAM よりも EKF SLAM の方が自己位置精度を上げている．また，Along-Track Error において，ローテーションセンサとジャイロセンサの動作モデルのエラー数値が低く同じなのは，どちらも車輪のオドメトリから正確な速度を算出し，動作モデルの算出に同じ値を割り当てているからだと考えられる．シナリオ 1 とシナリオ 2 に関しての違いであるが，決定付けられるほどの結果を得ることは出来なかった．つまり，それだけ動作モデルの影響が大きかったことが分かる．GPS に関しては，本実験において位置推定が行えなかったため，他の手法に比べて明らかな精度の悪さが表れている．

ターゲットセンサの位置推定精度に関しては，どちらのシナリオ共に EKF SLAM より FastSLAM の方が精度良く推定している．FastSLAM では最大エラー値が 2.7 [m]

に抑えられているのに対し，EKF SLAM では最大エラー値が 6.71 [m]になってしまっている．つまりそれだけ，FastSLAM のアルゴリズムが優秀であることが分かる．

4.3. 屋内実験

屋外実験と同様に，図 4.3.5 と図 4.3.6 の二つのシナリオについて実験し，得られた実験結果について，以下に評価・考察を示す．ここでは，屋外実験と比べて実験範囲が狭くなっていることを注意する．

4.3.1. RSSI 値の評価

4.2.1 項と同様に，屋内実験の移動ロボットの知覚に当たる環境計測モデルについて評価する．シナリオ 1，シナリオ 2 で得られた観測結果を，それぞれ図 4.3.1，図 4.3.2 に示す．

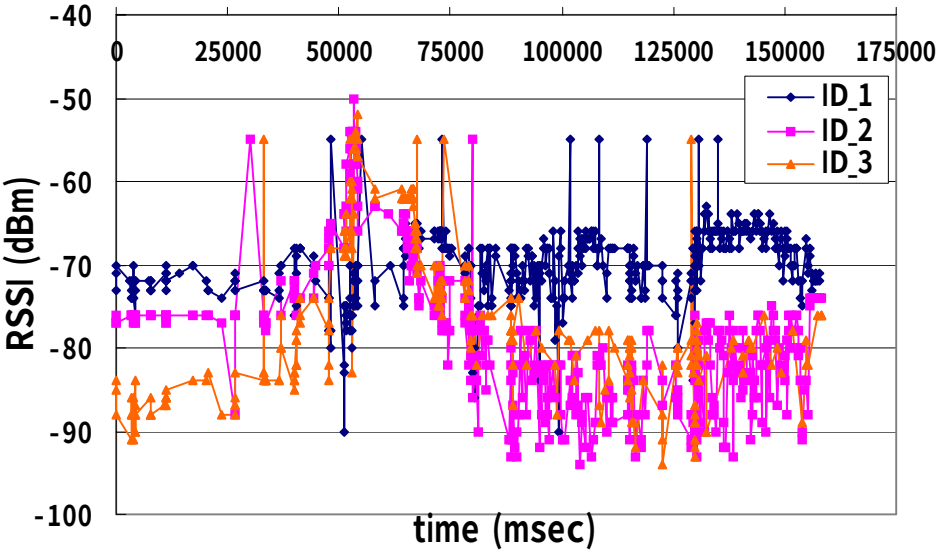


図 4.3.1　屋内での環境計測（シナリオ 1）

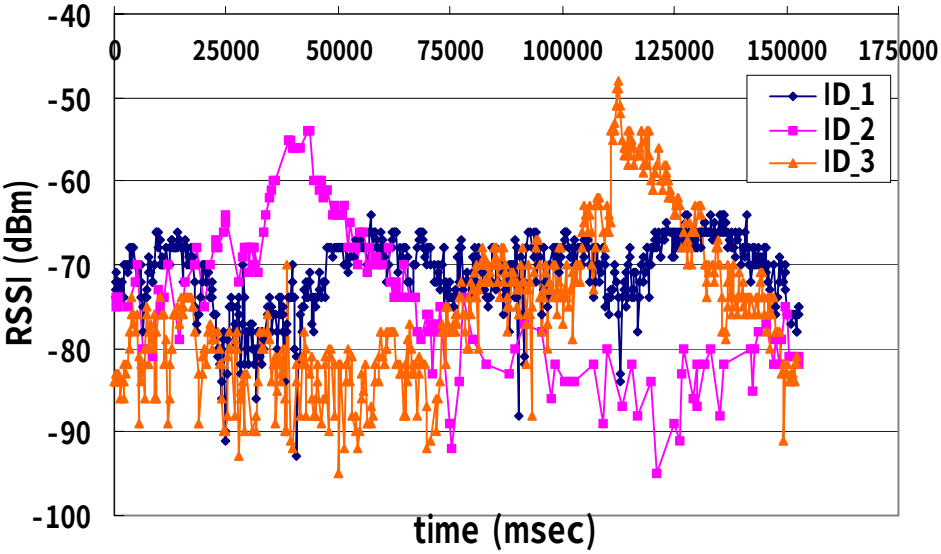


図 4.3.2　屋内での環境計測（シナリオ 2）

次に，屋内実験におけるターゲットセンサと移動ロボットの電波到達距離とパケットの到達率（PRR）の関係性を図 4.3.3 に示す．

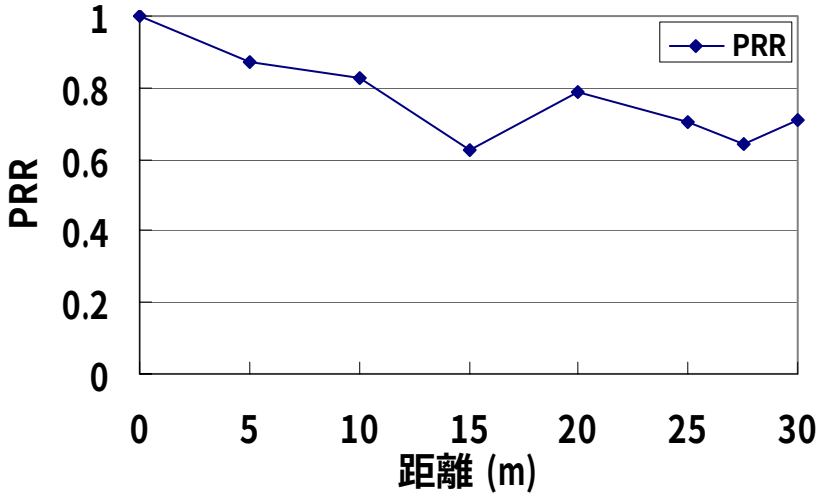


図 4.3.3　距離と PRR の関係性（屋内実験）

図 4.3.3 を見ると 30 [m] 離れた場所からでも 70%の PRR 比を得られていることが分かる．つまり本屋内実験においては，どのビーコン端末（ターゲットセンサ）からでも移動ロボットを見渡せていることが分かる．それにもかかわらず，両シナリオ共に各 ID の環境計測のサンプル数が均等に得られていなく，乱れが生じている．両シナリオ共に，ノード ID_1 のサンプル数が最も多い結果となった．シナリオ 2 においては，ID_2 のサンプル数は，ID_1 の約 5 分の 1，ID_3 の約 4 分の 1 と少なくなっている．このことは，屋内での実験のため，壁や柱などによる反射，建物内の設計構図，アンテナ（表 2-3 参照）の状態なども電波の強い弱いに影響を与えたものと考えられる．その他，ターゲットセンサと移動ロボットの通信には ZigBee を用いているため，建物インフラ設備の無線 LAN などによる干渉も考えられる．しかし上記の影響を受けながらも，図 4.3.1，図 4.3.2 共に RSSI 値と距離の関係性は現れている．

4.3.2. 移動精度の比較

4.2.2 項と同様に，シナリオ 1，シナリオ 2 で得られた移動動作の結果を，それぞれ図 4.3.4，図 4.3.5 に示す．

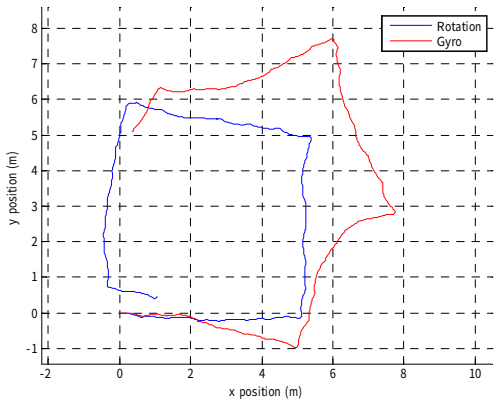


図 4.3.4　屋内の移動精度比較(シナリオ 1)

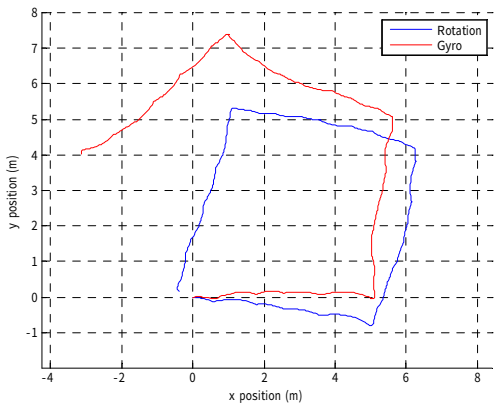


図 4.3.5　屋内の移動精度比較(シナリオ 2)

前述の通り，屋内での行路は凸凹が少なく摩擦抵抗が少ないため，予想通り移動ロボットのローテーションセンサ(回転角度値)を用いた場合の移動精度が高かった．GPS に関しては，屋内実験のため，電波を受信することはできなかった．

4.3.3. EKF SLAM の結果

4.2.3 項同様に EKF SLAM の実行結果を以下に示す．

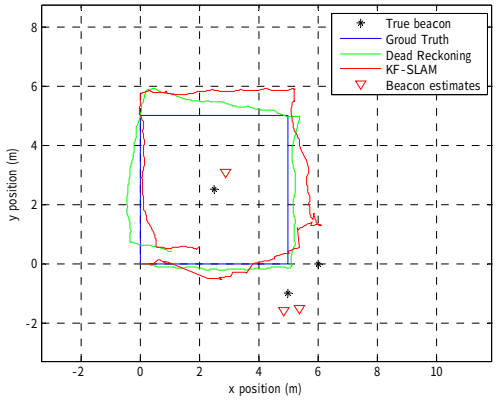


図 4.3.6 屋内シナリオ1のEKF SLAM
(移動動作モデル：Rotation Sensor)

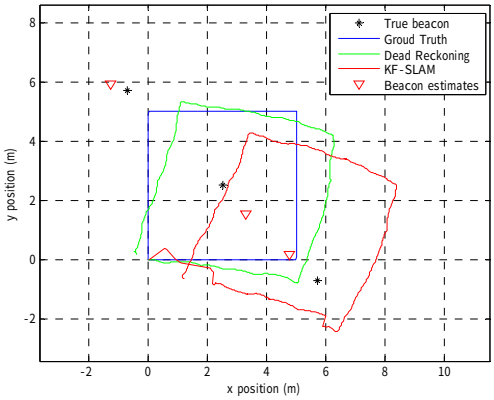


図 4.3.8 屋内シナリオ2のEKF SLAM
(移動動作モデル：Rotation Sensor)

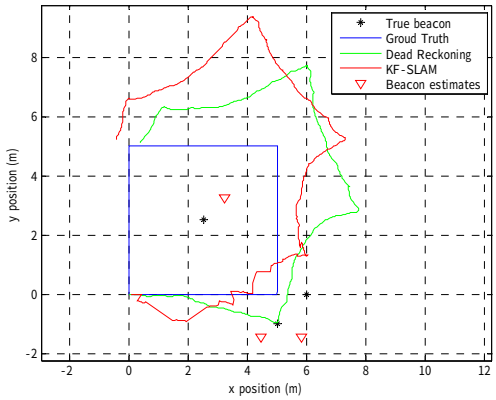


図 4.3.7 屋内シナリオ1のEKF SLAM
(移動動作モデル：Gyro Sensor)

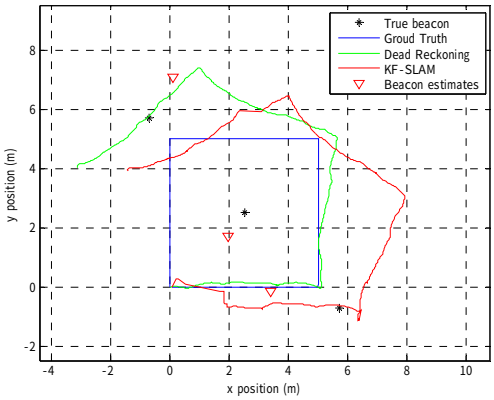


図 4.3.9 屋内シナリオ2のEKF SLAM
(移動動作モデル：Gyro Sensor)

4.3.4. FastSLAM の結果

4.2.4 項と同様に，FastSLAM2.0 の実行結果を以下に示す．

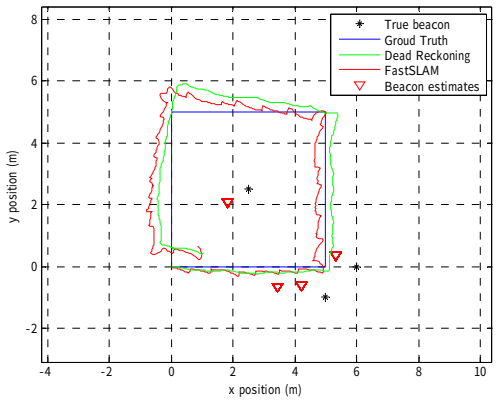


図 4.3.10　屋内シナリオ1のFastSLAM
(移動動作モデル：Rotation Sensor)

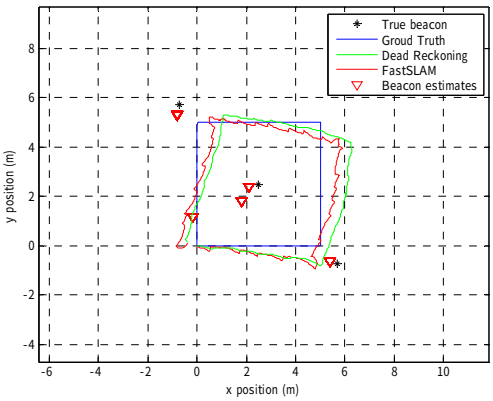


図 4.3.12　屋内シナリオ2のFastSLAM
(移動動作モデル：Rotation Sensor)

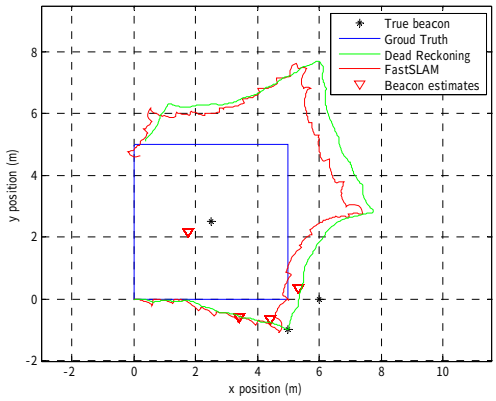


図 4.3.11　屋内シナリオ1のFastSLAM
(移動動作モデル：Gyro Sensor)

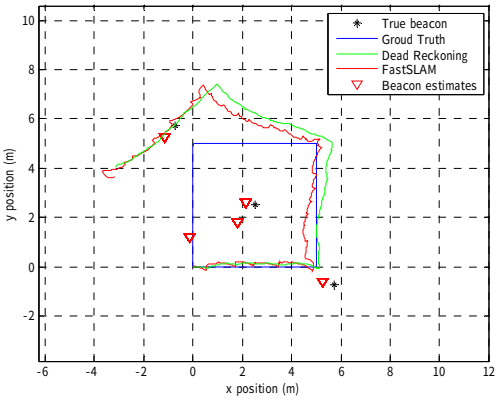


図 4.3.13　屋内シナリオ 2 のFastSLAM
(移動動作モデル：Gyro Sensor)

4.3.5. **エラーメトリック**

4.2.5 項と同様に，上記の結果についてエラーメトリックを用いて解析し，考察を行う．

1．Cross-Track Error

シナリオ 1 とシナリオ 2 において Cross-Track Error の平均値と最大値をそれぞれ図 4.3.14，図 4.3.15 に示す．また，標準偏差値を用いて解析した結果を図 4.3.16 に示す．

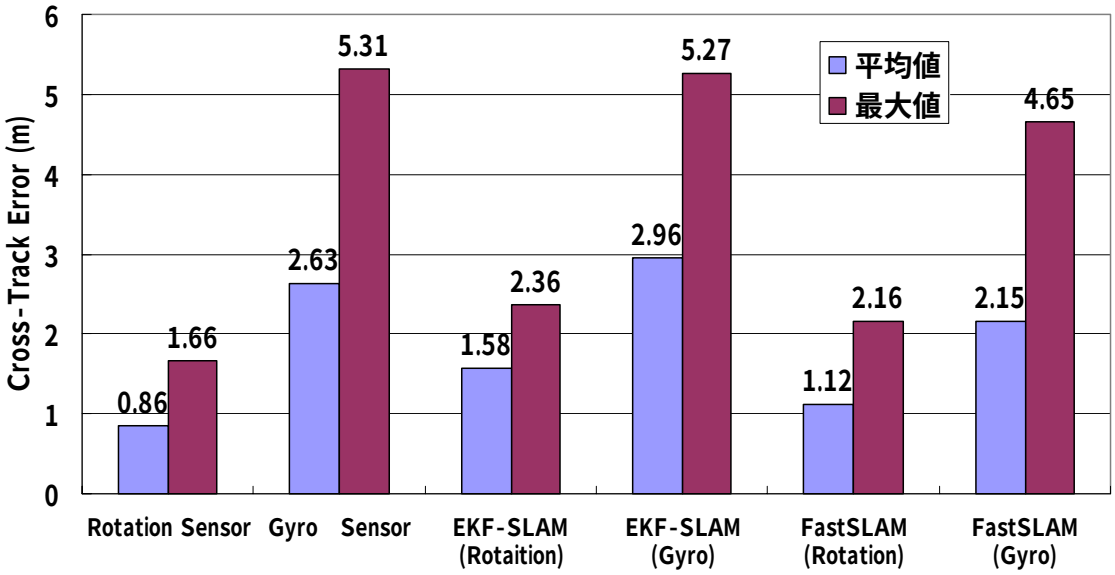


図 4.3.14　屋内シナリオ 1 の Cross-Track Error

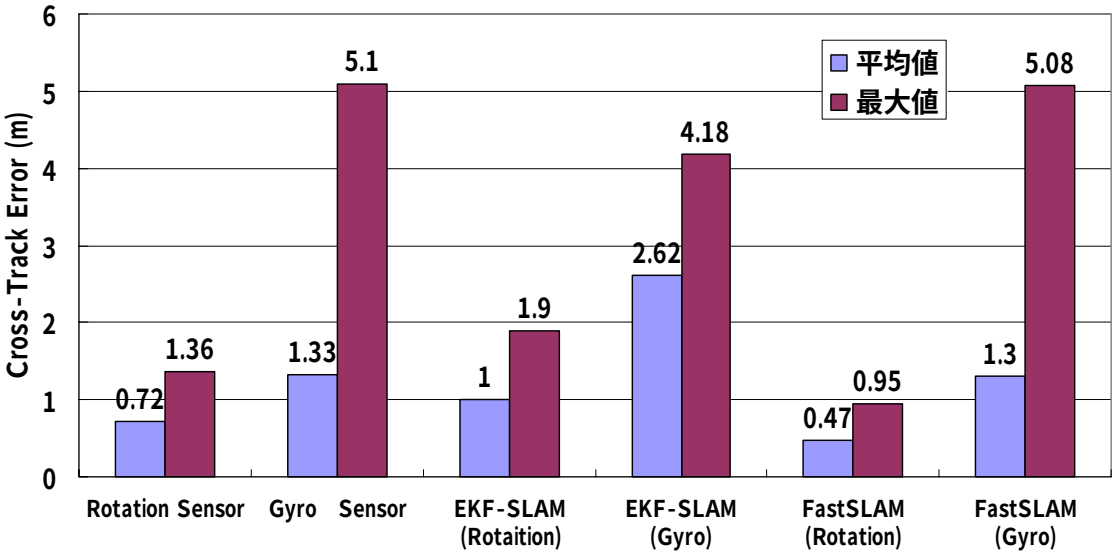


図 4.3.15　屋内シナリオ 2 の Cross-Track Error

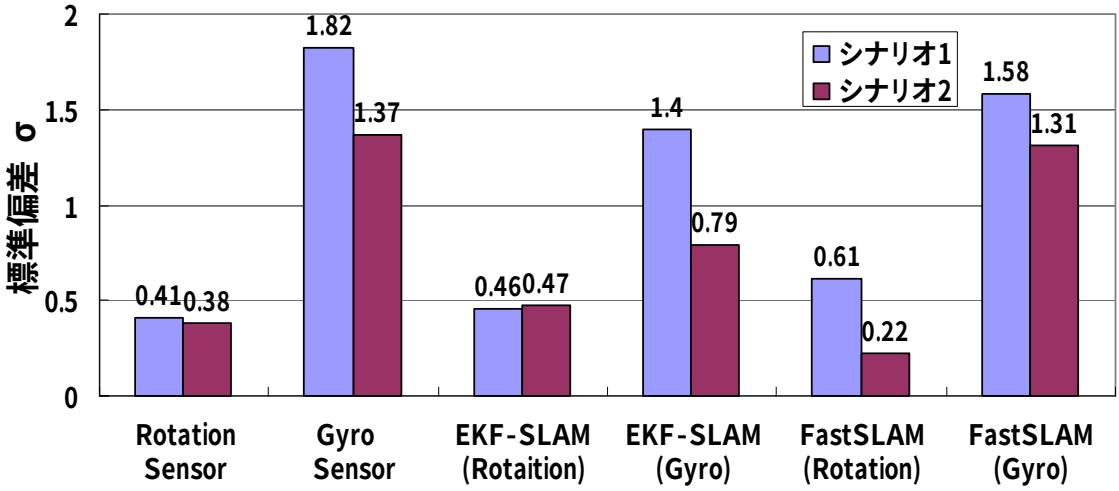


図 4.3.16　Cross-Track Error の標準偏差（屋内実験）

2．Along-Track Error

Cross-Track Error 同様に，Along-Track Error を図 4.3.17，図 4.3.18 に示す．また，標準偏差値を用いて解析した結果を図 4.3.19 に示す．

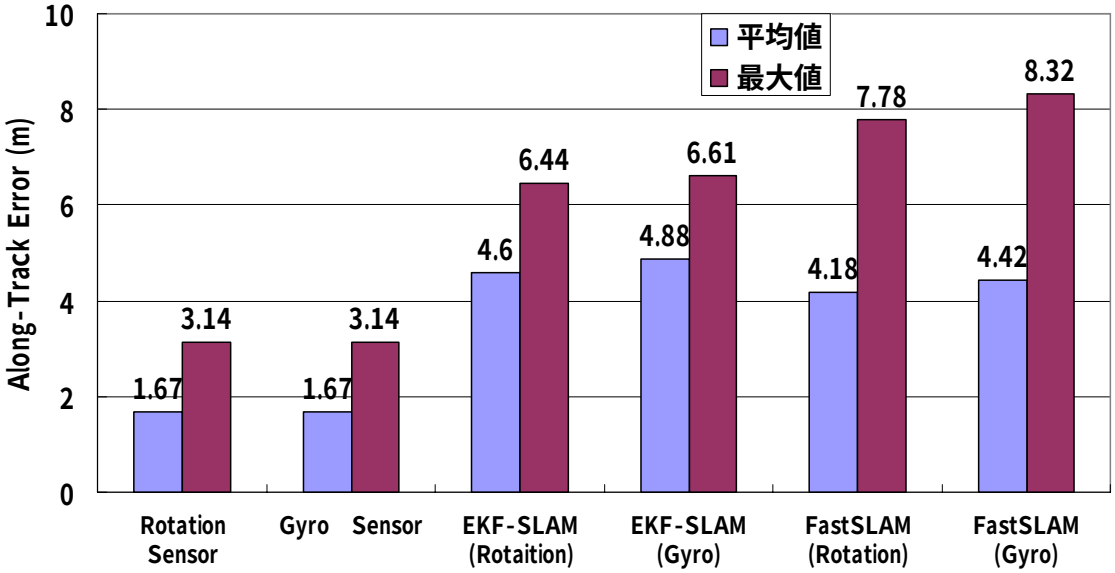


図 4.3.17　屋内シナリオ 1 の Along-Track Error

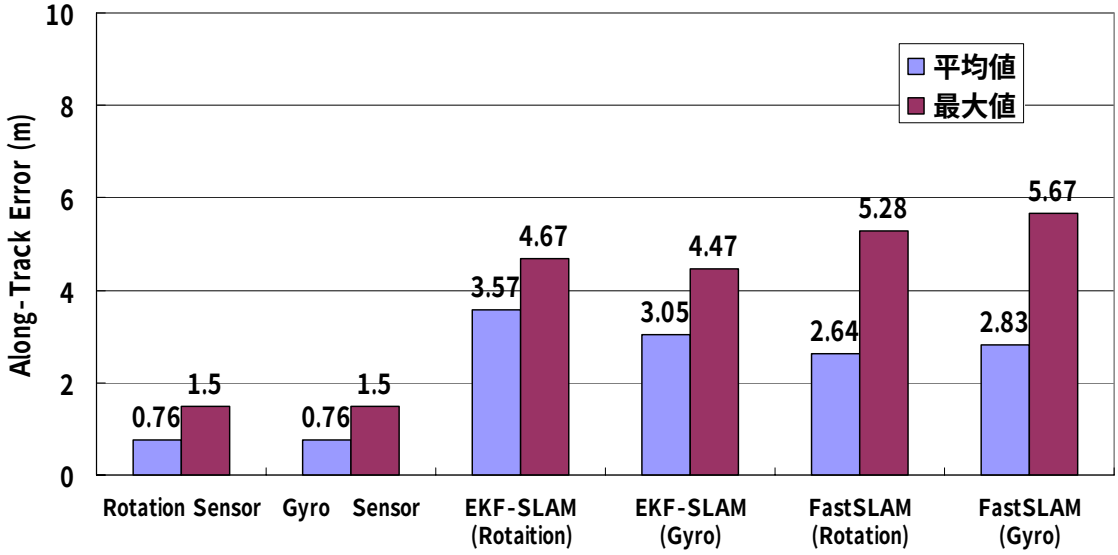


図 4.3.18　屋内シナリオ 2 の Along-Track Error

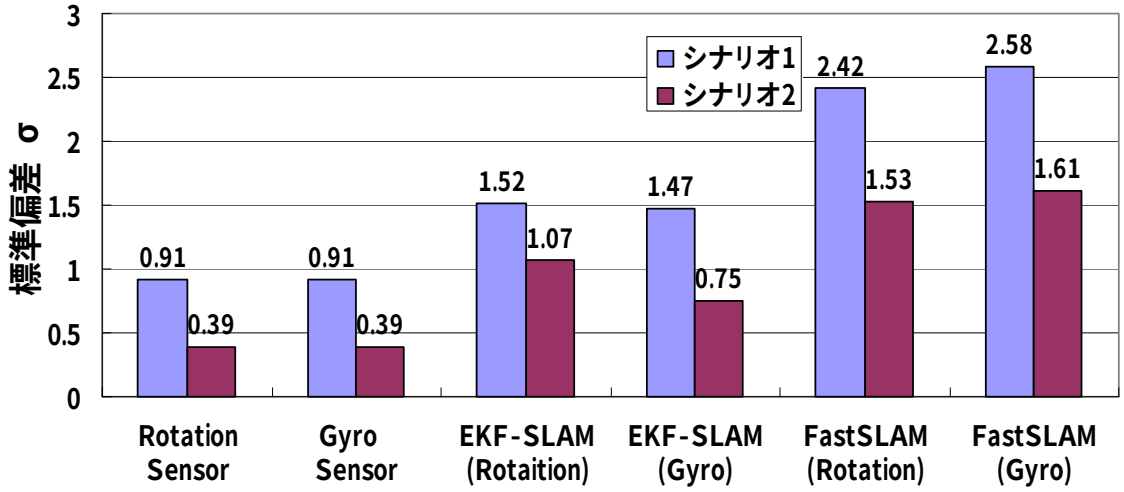
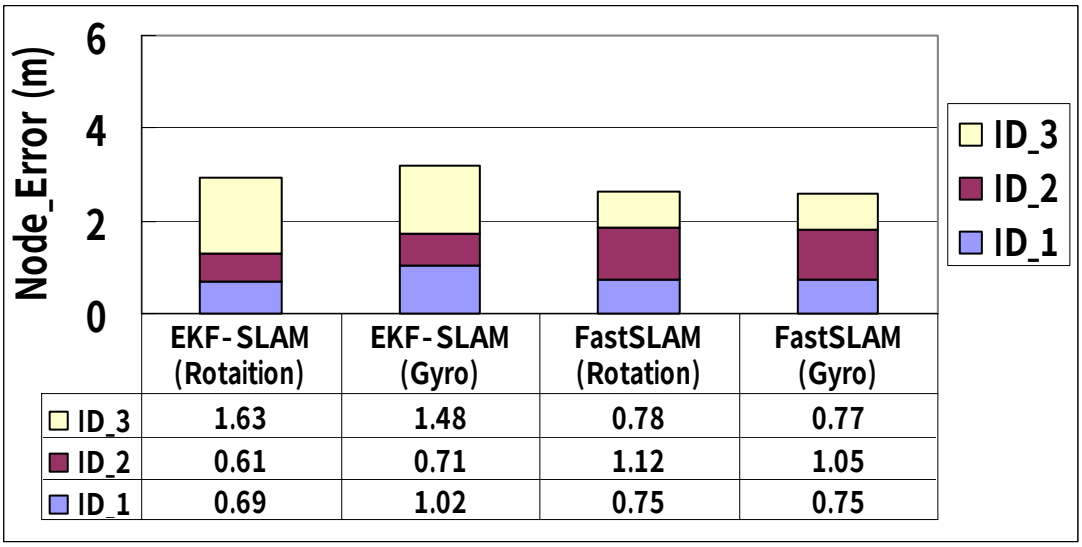


図 4.3.19　Along-Track Error の標準偏差（屋内実験）

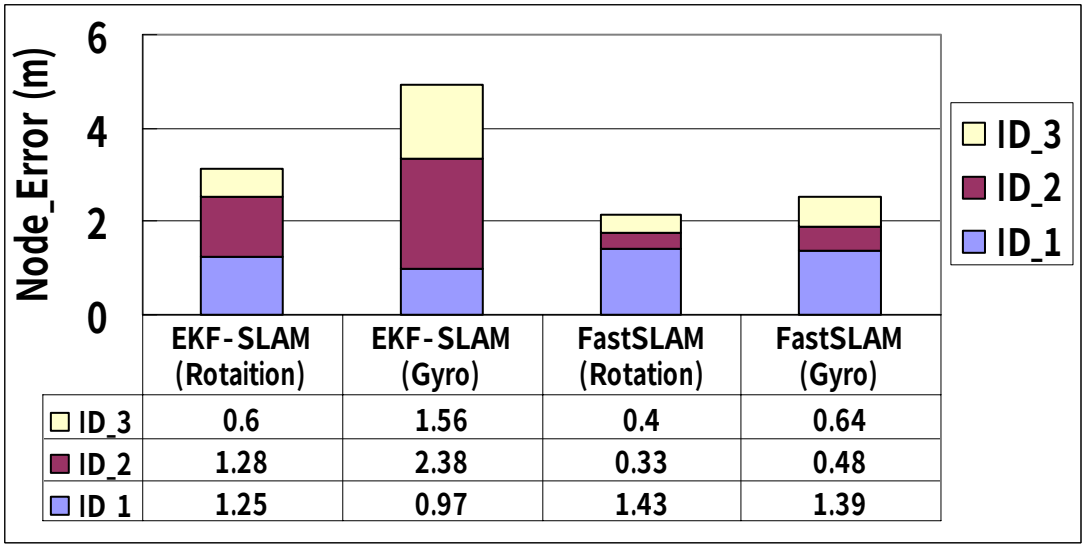
3．Node Error

シナリオ 1 とシナリオ 2 において Node Error をそれぞれ図 4.3.20，図 4.3.21 に示す.



	EKF-SLAM (Rotaition)	EKF-SLAM (Gyro)	FastSLAM (Rotation)	FastSLAM (Gyro)
ID_3	1.63	1.48	0.78	0.77
ID_2	0.61	0.71	1.12	1.05
ID_1	0.69	1.02	0.75	0.75

図 4.3.20　屋内シナリオ 1 の Node Error の標準偏差



	EKF-SLAM (Rotaition)	EKF-SLAM (Gyro)	FastSLAM (Rotation)	FastSLAM (Gyro)
ID_3	0.6	1.56	0.4	0.64
ID_2	1.28	2.38	0.33	0.48
ID 1	1.25	0.97	1.43	1.39

図 4.3.21　屋内シナリオ 2 の Node Error の標準偏差

4．考察

自己位置推定の精度においては，屋外実験ほどではないが，全般的に EKF SLAM よりも Fast SLAM のほうが精度を上げている傾向がある．屋外実験同様，動作モデルが曖昧だったジャイロセンサでは，FastSLAM よりも EKF SLAM の方が自己位置精度を上げる傾向がある．シナリオ 1 とシナリオ 2 に関しての違いであるが，4.3.3 項の EKF SLAM の結果を見ると，シナリオ 1 のように，局所的に受信電波が強いところでは，位置補正を向上させていることが分かる．つまり，電波環境が悪い屋内などの環境中では，ターゲットセンサの配置で局所的に受信電波を強くさせる箇所を増やすことによって，自己位置推定の精度を上げられることが分かる．

ターゲットセンサの位置推定精度に関しては，ばらつきがあるものの，EKF SLAM より FastSLAM の方が精度良く推定している．FastSLAM では最大エラー値が 1.43[m]，EKF SLAM では最大エラー値が 2.38 [m]であるように，EKF SLAM も，屋外に比べ実験範囲が狭かったためか，良く機能しているように見える．

第5章　結論

5.1. 総括

無線センサネットワークにおける，既存のセンサノード位置情報推定技術は，高密度にセンサ端末を配置するか，特殊な電波（超音波など）を利用するための専用設備が必要であるなど，個人ユースでの利用は困難であることが課題であった．また，既存の位置推定技術は受動的であることが前提とされ，可動性を考慮した能動的な設計が必要とされた．

そこで本研究では，プラットフォームとして市販のセンサノードと遠隔操作可能な廉価ホームロボットを利用し，特別なインフラ設備を必要としない位置推定システムを実験的に検証した．

ノード位置推定には，受信電波強度（RSSI 値）と距離との関係に着目し，その関係性をSLAM アルゴリズムに利用することでノード位置推定を実現した．

RSSI 値は無線通信時の副産物として得られるものなので，コストや消費電力を抑えることができる．また，移動ロボットを移動センサノードとして用いることで多数の固定ノードを必要とすることなく位置座標推定を行うことができる．

本実験では，移動ロボットが環境上を能動的にセンシングし，RSSI 観測値を取得することにより，位置の既知なセンサ端末の不要な，ノード位置推定手法を実装した．また，屋内外の 2 つのシナリオにおいて，EKF-SLAM と FastSLAM の手法を用いり，それぞれ検証を行った．以下は本実験まとめである．

- 移動ロボットとターゲットセンサとの RSSI と距離の関係性を示した．
- 屋外においても，車輪の動作モデルによる自己位置推定が良好であった
- 屋内のように電波環境が悪い条件においてもノード位置推定が行えた
- センサ配置の影響よりも移動ロボットの移動動作精度の影響の方が大きい
- GPS は屋内では使用できないが，15m×15m の範囲でも位置推定が難しい
- 今回の実験では EKF-SLAM より FastSLAM の方が，ノード位置推定精度が高い
- EKF SLAM は，電波環境が悪い屋内などでは，局所的に電波を強くすることで位置推定精度が向上する

最後に，本実験は，移動ロボットに LEGO 社の Mindstorms NXT とその純正センサ（ローテーションセンサ，ジャイロセンサ），センサ端末に Crossbow 社の Micaz MOTE，を用いたが，この実験機器は一例である．そのため，本実験結果などは，各社の製品仕様に依存することをここに記載しておく．

5.2. 今後の課題

今後の課題として以下のような事項が挙げられる．

- 位置情報推定精度の高精度化
  - 複数台の移動ロボットによる位置情報推定補完
- 位置推定の高速化
  - 適切な環境別 RSSI モデルの適用
  - プログラムの変更
- アプリケーションへの適用
- 新たなシステムの提案
  - 無線センサネットワークの強化(セキュリティやディペンダビリティ)
  - 画像技術との融合

参考文献

[1] R. Holman, J. Stanley, and T. Özkan-Haller, “Applying Video Sensor Networks to Nearshore Environment Monitoring”, IEEE Pervasive Computing, 2003
[2] S. Donay, “Body area sensor networks for health monitoring applications”, In Proc. of the 2nd International Workshop on Sensor and Actor Network Protocols and Applications (SANPA’04), 2004.
[3] L. Schwiebert, S. K. S. Gupta, and J. Weinmann, “Research Challenges in Wireless Networks of Biomedical Sensors”, MOBICOM’ 01, Jul 2001
[4] B. H. Wellenhoff, H. Lichtenegger and J. Collins, “Global Positioning System: Theory and Practice”, Fourth Edition, Springer Verlag, 1997
[5] Dragos Niculescu, Badri Nath, “Ad Hoc Positioning System (APS) Using AoA”, IEEE InfoCom 2003
[6] N.B. Priyantha, A.K.L.Miu, H.Balakrishnan, S.Teller, “The Cricket Compass for Context-aware Mobile Applications”, ACM/IEEE Mobicom, July, 2001
[7] N.Bulusu, J.Heidemann, and D.Estrin, “GPS-less low costoutdoor localization for very small devices”, IEEE PersonalCommunications Magazine, vol.7, no.5, pp.28切24, Oct.2000
[8] D.Nicolescu, and B.Nath, “Ad-Hoc Positioning Systems(APS)”, IEEE Glovecom 2001
[9] Tian He, Chengdu Huang, Brian M. Blum, John A, Stankovic and Tarek Abdelzaher, “Range-Free Localization Schemes for Large Scale Sensor Networks”, MOBICOM 2003
[10] NEST PROJECT
http://webs.cs.berkeley.edu/nest-index.html
[11] Swerling, P., “A proposed stagewise differential correction. procedure for satellite tracking and prediction”, Rand Corp. Paper P-1292, January 8,1958
[12] Kalman, R. E. “A New Approach to Linear Filtering and Prediction Problems”, Transactions of the ASME, Journal of Basic Engineering, Vol.82, pp. 35-45, 1960
[13] Derek Kurth, “Range-Only Robot Localization and SLAM with Radio”, CMU-RI-TR-04-29, Robotics Institute Carnegie Mellon University, May 2004
[14] Michael Montemerlo, “FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association”, School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213, 18th June 2003, CMU-CS-03
[15] Sebastian Thrun, Wolfram Burgard, Dieter Fox, “Probabilistic Robotics”, The MIT Press, September 1, 2005
[16] Crossbow Technology
http://www.xbow.com/
[18] LEGO.com Mindstorms NXT Home
http://mindstorms.lego.com/
[17] HiTechnic Products
http://www.hitechnic.com/

図一覧

図一覧

表一覧

謝辞

本研究を進めるにあたり，研究・生活の指導を初め、あらゆる面でご協力して下さった、甲藤 二郎 教授に深く感謝致します。

また，共に研究に励み，切磋琢磨したネットワーク班の皆様に感謝の意を表します.

2008 年 2 月

山田 寿夫