修士論文

群飛行ロボットの協調位置推定に関する研究

東京大学大学院
新領域創成科学研究科　基盤情報学専攻

学籍番号:36325
細井　一弘

2005 年 1 月 31 日提出
指導教官：杉本 雅則　助教授

要旨

自律ロボットの構成において、不確実性を含む状況下で，自己位置を推論する技術は重要である．しかし，飛行ロボットでは，ペイロードの制約から，位置推定に必要なセンサ群を，十分に搭載することができない．そこで本研究では，群飛行ロボットによる協調位置推定システムを提案する．複数台の飛行ロボットがお互いの環境情報を共有することで，位置推定の向上を図る．位置推定手法には，確率論的なアプローチである Monte Carlo Localization を応用し，飛行ロボットに適用する．本稿では，この位置推定手法について述べ，その評価実験の結果から，提案手法について議論する．

目 次

第1章　序論

1.1　背景

近年，自律移動ロボットの実現に向けた研究は盛んに行われている．オフィスや工場での使用を目的とした産業用ロボットだけでなく，ASIMO や AIBO に代表されるようなペットや友人としての役割を果すエンターテイメント性を持ったロボットなど，多種多様のロボット開発されている．しかし，それらの大半は地上で活動するロボットを対象としており，空中や水中を移動するロボットの研究は前者に比べて多くはない．本稿では，この後者のうち空中を移動するロボットについて取り扱う．

空中を移動する飛行ロボットは，地上ロボットに比べて次のような利点がある．

- 整地されていない場所や，瓦礫が散乱した場所でも移動が可能.（3 次元移動可能.）
- 高い位置から観測することで，一度に広域の情報を取得することができる．
- 屋内で使用する場合，天井付近を移動することで，人間の作業の邪魔にならない.（人間の生活空間とロボットの移動空間が分離）

このような大きな利点を持つ飛行ロボットは実用化に向けて様々な研究がなされてきた．主にヘリコプター [26] や飛行機 [21]，飛行船を用いた屋外用のロボットで，かなりの技術力と予算が必要とされていた．しかし，これらの研究のほとんどは遠隔飛行の実現に留まり，自律移動を実現しているものは少ない．

これに対して屋内用飛行ロボットは，屋外用に比べて規模が小さく，風などの外界の影響も少ない．また比較的低コストで実現できるので自律移動を目指した研究が行われてきた．屋内で自律移動可能な飛行ロボットが実現できれば，今までにはないロボットシステムへの応用が考えられる．特に地上ロボットには不可能な 3 次元移動能力は様々な可能性を秘めている．簡単な例を挙げると

1. 災害地での被害状況の観測や被災者の探査（救助支援）
2. 段差のある出入口や階段を経由する場所へのナビゲーション.
3. 博物館のように展示品によって多くの死角ができる場所での監視.
4. 曲芸飛行や編隊飛行を可能とするエンターテイメントロボット.

等が考えられる．特に 1 番目の被災地での救助支援活動は，最近の度重なる大地震を契機に社会的に重要な問題となっている．このため救助支援を目指すレスキューロボットの研究への期待は大きい．

飛行ロボットが実際の環境内で自律移動するためには，移動経路の計画や目標の発見，未知物体との衝突回避などが重要な課題である．それらの課題を実行するためには，ロボットが現在どのような状況下にあるかの情報，特に自己位置を知ることが必要である．地上ロボットではそれらの情報を獲得するために，ロボットに赤外線センサ，超音波センサ，レーザーレンジファインダ，カメラ等の

様々なセンサを搭載し，このセンサ情報を基に自己位置を推定している．しかし，飛行ロボットではペイロードの問題から，これら全てのセンサを搭載することはできない．そのため一般的な屋内用飛行ロボットの研究では，センサとしてカメラを用いている．カメラはロボット周辺の多くの情報を一度に取得する視覚センサとして有望であるが，実際に飛行ロボットに搭載可能なカメラは軽量であることを優先させるため，低画質の画像から自己の位置を推定しなければならない．

本研究では，以上の問題を踏まえて群飛行ロボットによる協調位置推定システムを提案する．単体の飛行ロボットでは，十分に得られない外界の情報を，複数台の飛行ロボットがお互いの情報を共有することで補う．それによって，自己位置推定の誤りを軽減させる．また，群飛行ロボットによるアプローチはその応用を考えたときに，複数台の飛行ロボットが協調して作業を行うことで，単体の飛行ロボットよりも効率の良い作業を行えることが期待できる．

本研究では飛行ロボットのプラットフォームとして飛行船を用いる．飛行船は空気より軽い気体（通常はヘリウム）をエンベロープ（気球部分）に充填することで浮力を得るので，飛行ロボットを実現させるのに比較的簡単な手段である．ペイロードは，エンベロープ（積載重量）の大きさに応じて決まる．

利点としては次の通りある．

- 飛行が静かで危険を伴わない．
- 壁に衝突してもロボット自体にほとんど損害を受けない．
- 航空力学の知識が十分になくても開発できる．
- 材料が安価で入手しやすく，メンテナンスが容易である．

欠点としては

- 慣性や空気抵抗の影響が大きい．
- ヘリウムが必要である．
- ノンホロノミック系で制御が簡単でない．

などがある．

また飛行船は比較的簡単に空中停止ができ，探査や監視といった応用にも対応できる．センサとしてはワイヤレスカメラを搭載し，環境に設置された landmark を観測することで自己位置推定を行う．また各飛行ロボットにもお互いを認識できるようにマーカを付ける．このマーカをカメラで認識することでお互いの相対位置関係の情報得る．これらの情報から，ロボット全体の位置情報を統合して位置推定を行う．

1.2 論文の構成

本稿の構成は以下の通りである．第 2 章で飛行ロボットと位置推定に関する先行研究について述べ, 第 3 章でシステムの構成について述べる．第 4 章では自律飛行に必要となる行動プランニングについて述べ，第 5 章では飛行ロボットの力学モデルについて述べる．第 6 章では具体的な協調位置推定アルゴリズムについて述べる．第 7，8 章で評価実験とその考察を述べ，第 9 章で結論を述べる．

第2章　関連研究

本章では屋内用飛行ロボットに関する先行研究と，移動ロボットの位置推定に関する先行研究について説明する．

2.1　屋内用飛行ロボットの分類

屋内用飛行ロボットはその推進機構から (1) 小型飛行船，(2) 昆虫型，(3) ヘリコプター，(4) 飛行機の４つに分類される．以下にそれぞれの説明を示す．

(1) 小型飛行船 小型飛行船の自律移動に向けた研究としては，Zhang[32] や Zufferey[33]，Iida[11]，Welsby[30] の研究が挙げられる．Zhang や Zufferey の研究では，小型飛行船にセンサ類としてカメラだけを搭載し，目標物までの自動操縦を実現させるシステムを研究した．また，Iida らの研究では進化的ロボティクスの観点から，昆虫を模倣したビジョンシステムによる飛行船の自律移動の研究を行った．

Welsby らの研究では，複数台の飛行船を用いて群ロボットとしての自律移動の研究を行っている．この研究では，位置推定のために超音波センサや赤外線センサを利用しいる．

(2) 昆虫型 昆虫や小鳥のように羽を羽ばたかせて揚力を得るロボットであるが，屋内環境で自由にコントロールできるレベルには至っていない．現在，カリフォルニア大学 Berkeley 校において MFI(Micromechanical Flying Insect)[3] と呼ばれる，大きさ 4 分の 1 インチ，重さ 10 分の 1 グラムの小型飛行ロボットの実現を目指すプロジェクトが進行している．

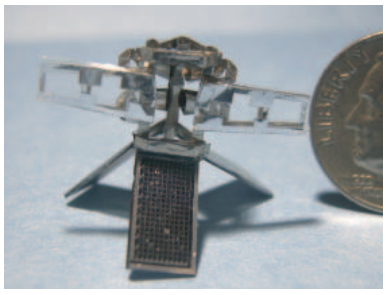

図 2.1: MFI（[3] より引用）

(3) ヘリコプター (回転翼) 一般的なヘリコプターは騒音が激しく，また屋内で利用するには危険である．さらに市販のラジコンヘリコプターは，重量が大きく価格も高価である．このため，ヘリコプターは屋内型自律移動ロボットの研究には向いていない．しかしながら，数人の研究者

たちは屋内利用の小型ヘリコプターについて研究を行っている [16][2]．これらの小型ヘリコプターの問題点としては壊れやすく，ペイロードにも敏感で，飛行時間もきわめて短い．

図 2.2: Pixel（[2] より引用）

(4) 飛行機（固定翼） モータやエンジンによって推進し，翼面の上下に発生する流体の圧力差から揚力を得て飛行する．揚力を得るためにある程度の速度で飛行しなければならない．このため，屋内環境で低速で飛行するのは難しい．また障害物の多い所でも飛行はほとんど不可能である．このように飛行機は屋内環境での利用には不向きであるが，Nicoud ら [22] は屋内環境で利用可能な小型で低速の飛行機の実現に向けた研究を行っている．

図 2.3: Model C4（[22] より引用）

以上の飛行ロボットを比較するために，屋内利用，観測，コスト，安全性の 4 つの観点から比較したものを表 2.1 に示す．

表 2.1: 各種飛行ロボットの比較

	屋内利用	観測	コスト	安全性
昆虫型	△	○	high	○
ヘリコプター	△	○	high	△
飛行機	○	×	low	△
飛行船	○	○	low	○

2.2 移動ロボットの位置推定

ロボットの位置推定は，ロボットが自律移動するために重要な役割を果たすため，これまで多くの研究が行われてきた．ロボットの位置推定アルゴリズムには，使用するロボットや環境に応じて様々なものがある．本稿では位置推定アルゴリズムを単体のロボットのみで行われる「自己位置推定」と，群ロボットなどとして知られる複数台のロボットがお互いの情報を共有して位置推定を行う「協調位置推定」の２つに分類して説明する．

2.2.1 移動ロボットの自己位置推定

移動ロボットの自己位置推定とは，オドメトリ[1] やビジョンなどのようなセンサ情報からロボットの位置と姿勢を決めることである．ロボットの位置推定は，問題設定に応じて *Position Tracking,Global Localization Problem,Kidnapped Robot Problem* ３つに分類することができる．

Position Tracking

最も簡単な問題で，ロボットの初期位置やロボットの置かれる環境情報（大きさや障害物など），センサのノイズモデルが予め与えられている．また，オドメトリの誤差も小さい．ロボットはオドメトリの誤差を修正しながら自己位置を推定する．

Global Localization Problem

ロボットの初期位置が未知の状態で位置推定をスタートし，自己位置を推定する．また，オドメトリの誤差や，センサノイズが小さいという仮定もできない問題設定である．

Kidnapped Robot Problem

Global Localization の問題設定で，ある時点でロボットを（人間の手によって）別の場所に移動させる．このときロボットには移動したという情報は与えず，移動後のセンサ情報から位置推定を行う（図 2.4 参照）．この問題設定は，Global Localization Problem においてセンサ情報に致命的なエラーがあった場合に起こりうる状況と同等で，ロボットが間違った位置を推定しまった時に，正しい状態に復帰できる能力を持っているかテストするときによく用いられる．

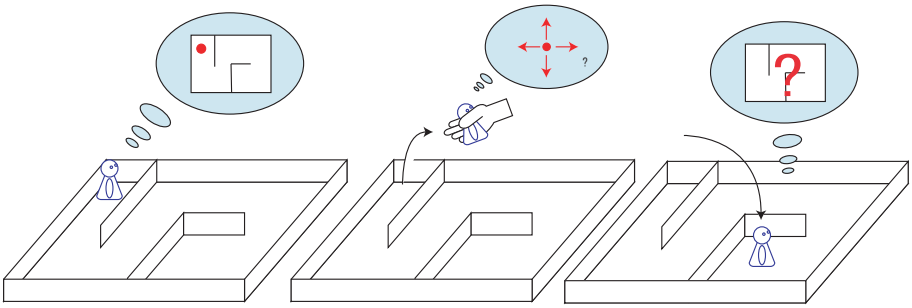

図 2.4: Kidnapped robot problem

[1]オドメトリとは車輪型の移動ロボットにおいて，回転エンコーダなど利用して車輪の回転角度から，位置姿勢を推定する方法である．移動距離は回転量に比例して変化するので，移動ロボットの移動距離を推定することができる．また，左右の車輪の回転数の差から移動ロボットの旋回角度を推定できる．

以上の 3 つの問題は，それぞれの説明からも分かるように Position Tracking，Global Localization Problem，Kidnapped Robot Problem の順に問題が難しくなる．ロボットの自己位置推定に関する研究は数多く存在するが，それらの大半は Position Tracking を問題の対象として扱っている．主に Kalman Filter [2]を用いた手法が提案されている．Kalman Filter を用いることで過去の位置情報と観測データから位置を予測し，現在の観測データと比較することで現在位置を推定する．この位置推定の過程からも分かるように，Kalman Filter を用いる場合は過去の位置情報が既知である必要がある．つまり，位置推定の開始時には初期位置が与えられている必要がある．また，センサデータやオドメトリに大きな誤差が生じる場合，ロボットは自己位置を見失う可能性がある．従って，以上の理由から基本的な Kalman Filter では Global Localization Problem や Kidnapped Robot Problem において，正しい位置推定ができない．

Global Localization を解決する手法としては，Kalman Filter を応用した Multi-Hypothesis Kalman Filter[12] や Markov Localization[7]，Monte Carlo Localization[6, 29] 等があげられる．

Multi-Hypothesis Kalman Filter

Gaussian Mixture(ガウス混合分布) を用いて Kalman Filter を構成する．Gaussian Mixture を用いることで，複数の異なった仮定を分離して取り扱うことができる．マルチモーダルな表現を利用して Global Localization を行うことが可能である．しかし，それぞれのセンサデータをどの仮定と結びつけるかという問題があり，データの関連付けにヒューリスティックな技術が必要となる．

Markov Localization

ロボットの置かれる環境（状態空間）を有限個のセルに分離し，それぞれのセルに対してロボットの存在確率を割り当てる．このときロボットの存在確率はセルを単位として区分的に一定である．利点としては任意の確率分布をモデル化することができ，従って Global Localization も可能である．欠点としては状態空間を分割してできる膨大な数のセルについて存在確率を計算する必要があり，ほとんどリアルタイムに位置推定が行えないことである．また状態空間を粗く分割した場合，推定される位置にも大きな誤差を含んでしまう．

Monte Carlo Localization

Particle Filter を用いた位置推定法である．Markov Localization のように状態空間を分割せずにロボットの存在確率を“ サンプル ”で表現することで確率分布を近似し，リアルタイムな位置推定を可能にしている．Monte Carlo Localization は第 6 章の 6.3 で詳しく説明する．

以上の自己位置推定法は，ベイジアンフィルタに基づいたものである（ベイジアンフィルタについては第 6 章の 6.2.1 で詳しく述べる）．ベイジアンフィルタでのデータの取り扱いについてこれらの手法を分類すると，図 2.5 のようになる．

2.2.2 協調位置推定

群ロボットシステムにおいて，グループ内のそれぞれのロボットがどのような位置関係にあるのか（相対位置），またグループ全体が環境内のどこにいるのか（絶対位置）を知ることは，群ロボットが協調して自律移動行う上で必要不可欠である．群ロボットでの位置推定は各ロボットが自己位置推

[2]Kalman Filter は一般的に現在までの測定値から一期先を予想し，新しい計測値が得られたらその値と予測との誤差を評価して予測精度を改善するアルゴリズムである．

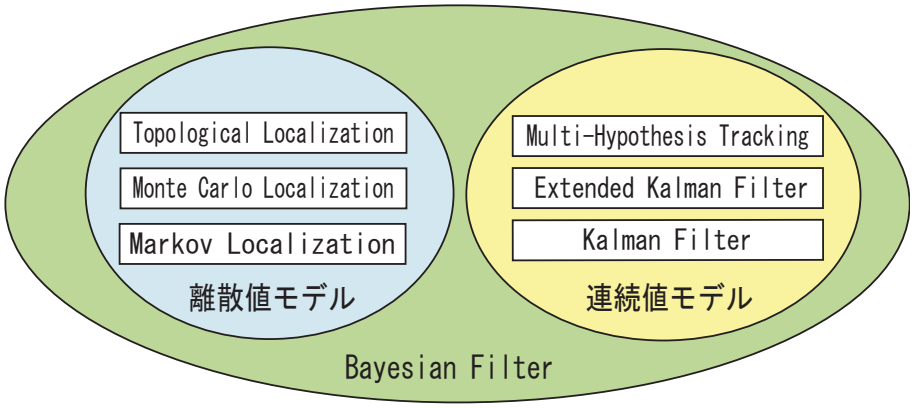

図 2.5: Bayesian Filter を用いた位置推定法の分類

定を行うだけでなく，お互いの位置情報や観測情報を共有することで，センサのノイズなどの影響による推定の誤りを軽減し，位置推定の精度を高めることが可能である．

このような観点から，さまざまな群ロボットシステムの協調位置推定法やアルゴリズムが研究されてきた．

Mover-Observer Strategy

Kurazume ら [24] は，最初に群ロボットによる協調ポジショニング法，CPS（Cooperative Position System) を提案した．このシステムでは，ロボットのグループが 2 つのチームに分かれて行動する．仮にこの 2 チームを A,B とする．A チームが移動する時，B チームは位置推定のための landmark として振舞う（静止している）．Kurazume らはこの移動可能な landmark を“ Potable Landmark ”と呼んでいる．A チームの移動が終わると，今度は A チームが landmark となり，B チームが移動する．この処理を繰り返すことでグループ全体がゴールまで移動する．このシステムは [19, 18] でさらに改良・最適化されている．

Grabowski ら [8] は，小型ロボット（Millibots）によるチームで位置推定を行う研究をした．Grabowski らの研究では 1 台のロボットが移動している間，残りのロボットが正三角形の体勢を採ることで位置推定を行っている．

以上の研究は,「移動するロボット:Mover」と「landmark となるロボット:Observer」となるように，グループ内のロボットが別々の役割を果たすアプローチ“ Mover-Observer Strategy ”を採用している．

また，グループ内のロボットを Mover，Observer と分けないで，全てのロボットが同時に移動しながら位置推定を行う位置推定法も研究されている．このようなアプローチでは，確率理論に基づいたアルゴリズムを基に協調位置推定が行われる．特に Kalman Filter や最尤推定法，Particle Filter を用いた推定法が有名である．

Kalman Filter

Roumeliotis ら [27, 25] は，Kalman Filter を以下のようにしてロボットの位置推定に利用している．グループに属する各々のロボットは自分の動きと一致するセンサデータを蓄積し，他のロボットと出会った時にそのデータを共有する．この時，Kalman Filter はそれぞれのロボットが共有するデータをフィルタリングする機能として働く．この処理を繰り返し行うことで，ロボットは自己で推定した位置の“ 不確かさ ”を減少させることができる．Roumeliotis らが提案した位置推定法では，観測者となるロボットなしで全てのロボットは移動しながら協調的に位置推定ができる．

最尤推定法

Howard ら [9] は，Kalman Filter の代わりに最尤推定法を用いてそれぞれのロボットの情報を統合している．それぞれのロボットは予め位置を推定しておき，これを推定量 H で表す．次に，ロボットのセンサからの情報を用いて観測値の組 O を生成する．最後に，数値最適化によって観測値 O に最も見合う推定値 H を求める．

Particle Filter

Fox ら [5] は環境地図を備えた 2 台のロボットについて，Monte Carlo Localization を拡張したアルゴリズムを構築している．2 台のロボットがお互いを発見した時に，その観測データをそれぞれの存在確率密度関数に取り込む．この位置推定法もまた，観測者となるロボットなしで全てのロボットは移動しながら協調的に位置推定ができる．

Howard ら [10] はグループ内のロボットの相対位置関係を決めるために，Particle Filter を用いている．各ロボットは自己を中心とするマップを独立して持っており，自己の観測情報と他のロボットからの観測データによってこのマップを更新していく．このマップを更新するときに Particle Filter が使われている．

第3章　システムの構成

本研究では群飛行ロボットの自律移動のための自己位置推定システムを構築する．本章ではそのシステム構成について述べる．

3.1　構成の概要

はじめに飛行ロボットが単体（1 台）の場合についての基本構成を述べる．システムの基本構成は

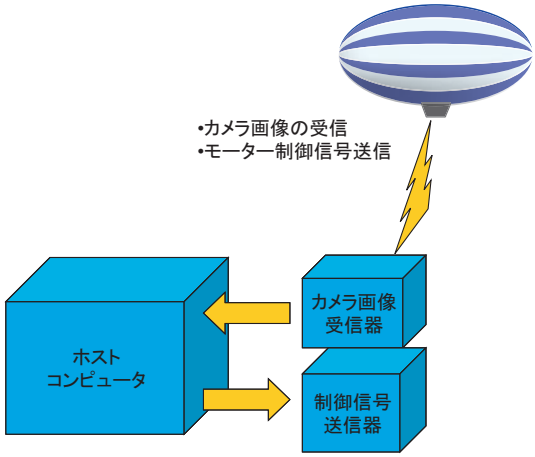

図 3.1: 基本構成

図 3.1 に示すように，主に飛行ロボット，カメラ画像受信機，制御信号送信機，ホストコンピュータの４つから構成される．飛行ロボットにはカメラが搭載されおり，カメラから撮影された画像をカメラ画像受信機で受信し，A/D 変換を通してホストコンピュータに送れらる．

3.2　ハードウェア

3.2.1　飛行ロボット

飛行ロボットには，市販の屋内用ラジコン飛行船 (株) タカラ ドリームフォース 02 スカイシップ [28]) を用いる (図 3.2 参照)．飛行船のサイズは全長約 90cm でエンベロープ部に約 70 リットルのヘリウムを充填できる．ラジコン飛行船は浮力を得るためのエンベロープ部とモータ，プロペラ，制御信号受信機，バッテリ（ニッケル水素電池）を統合した駆動部から構成される．さらに外界の情報を得るために，小型ワイヤレスカメラ（カメラ部）を搭載する．カメラはカメラモジュールと画像送信

図 3.2: (株) タカラ ドリームフォース 02 スカイシップ

機が一体になっている．以上をまとめると，飛行船は図 3.3 のように構成される．

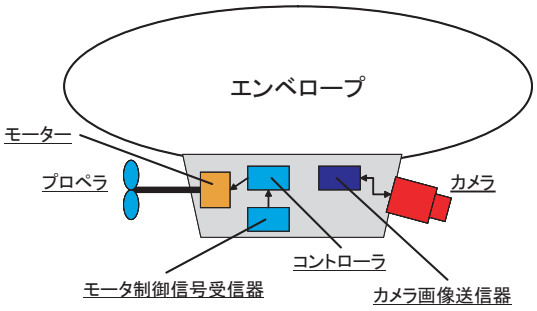

図 3.3: 飛行船の構成

ペイロード

飛行船のペイロードは，エンベロープに充填するヘリウムの体積によって決まる．ヘリウム 1 リットルあたりのペイロードは，

$$(\text{空気の密度} - \text{ヘリウムの密度}) \times 1 \text{ リットル} = (1.2250kg/m^3 - 0.1785kg/m^3) = 1.0465g/\text{リットル} \tag{3.1}$$

となり約 1 gであることがわかる．ラジコン飛行船の稼動部は 37.4g であり，カメラは 49g なので必要なペイロードは合計 86.4g となる．理論的には約 87 リットルのヘリウムで浮くことになるが，実際にはエンベロープの中を 100 %ヘリウムに維持するのは難しい．またカメラや稼動部とペイロードを接着するためのテープの重さなども考慮に入れるので，計算値より大きいエンベロープを用意する必要がある．しかし適切な大きさのエンベロープは入手困難であるため，市販のエンベロープ (容積：約 90 リットル) を 2 つ使い，図 3.4 の飛行ロボットを構築した．浮力の調整はバラスト（重り）により行った．

図 3.4: 飛行ロボット

3.2.2 推進機構

スカイシップには，図 3.5 に示すようにゴンドラの左右にプロペラが備え付けられている．左右のプロペラは独立して正転・逆転することができ， それにより左右の旋回はもちろん，その場での信地旋回などの動きを可能にする．また，2 つのプロペラをつなぐ主軸を回転させることでプロペラ自体を上向き・下向きにし，上昇・下降の操作も可能にしている（ティルト機構）．

スカイシップではこれらの機構を構成するため，合計 3 つのモータを用いている．

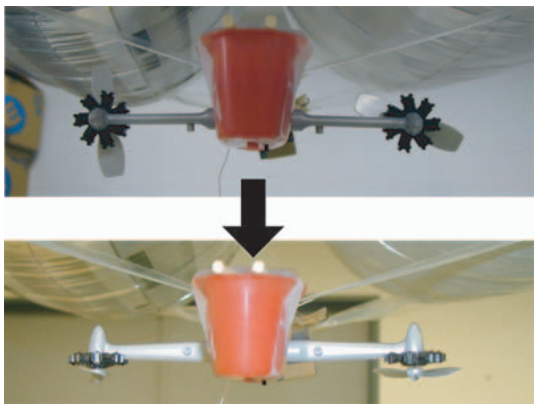

図 3.5: 推進機構

3.2.3 搭載カメラ

本研究で使用する小型カメラには，The Card（(株)RF システム）を採用した．このカメラの仕様は表 3.1 の通りである．このカメラは赤外線カメラなので，暗所でも赤外ライトをつけることで撮影が可能になる．

表 3.1: カメラ (The Card) 仕様

サイズ	8 × 49 × 83 (mm)
重量	49g
画素数	27 万画素
電源	内蔵電池 (1.2V 330mA)
電波エリア	100m(見通し)
信号	NTSC

図 3.6: The Card

また，カメラの受信機には専用受信機 (BS-10G) を用いる．この受信機の出力は NTSC 信号なので，アナログ－デジタルコンバータ (canopus(株) ADVC-100) により，デジタル信号に変換する．

3.2.4 制御機構

飛行ロボットの制御には，スカイシップ付属のコントローラに用いる．コントローラは，3.2.2 で説明したプロペラとティルトモータを制御可能である．

コントローラ仕様

コントローラには，Highland 社製のラジコン用 LSI（TX6C：図 3.7 参照) が用いられている．TX6C は RX6C（本体に内蔵）とセットで用いられ，リモコンからの操作を本体に送信することができる．TX6C/RX6C は全部で 7 つの操作（前進，回転など）を割り当てることができ，それぞれの操作の on，off を切り替えることができる．スカイシップではこの LSI に 6 つの操作（表 3.2 参照）を割り

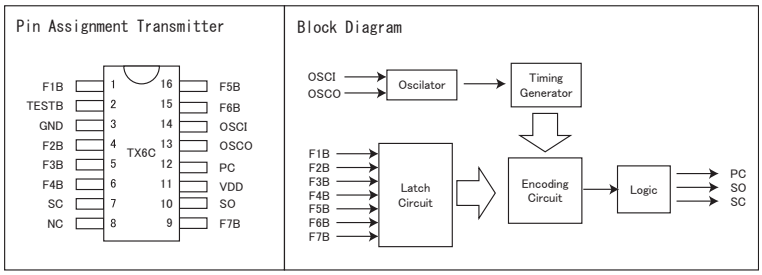

図 3.7: TX6C

当て，コントローラから本体に制御信号を送信している．

表 3.2: コントローラの操作表

1	左プロペラモータ正回転
2	左プロペラモータ逆回転
3	右プロペラモータ正回転
4	右プロペラモータ逆回転
5	ティルトモータ正回転
6	ティルトモータ逆回転

注）1 と 2，3 と 4，5 と 6 は同時に on にはしない．

シリアルコンバータ

PC から飛行ロボットを直接制御するために，PIC（Peripheral Interface Controller）を介してシリアル通信によりコントローラを制御できるシリアルコンバータ（図 3.8 参照）を製作した．PIC のポートをコントローラの TX6C に直接接続することで，PC からコントローラを操作する．PIC 自体は，シリアル通信により PC から「どのポートを on にするか」という命令を送信する．回路構成は図 3.9 のようになる．

3.3 ソフトウェア

3.3.1 処理の概要

内部コンピュータ内では飛行ロボットの位置を推定し，次の行動命令を生成する処理が行われる．位置の推定には外部環境に設置された landmark(位置を予めシステムに登録されている．) を利用する．位置推定についての詳細は，第 6 章で説明する．位置が推定されるとその位置に応じた行動を決定し，その行動を採るための制御信号が生成される．

図 3.1 で示したコンピュータの内部処理は，図 3.10 に示す通りである．処理は大きく分けて，画像処理，landmark 位置検出，自己位置推定，行動決定，制御信号生成の 5 つに分割できる．

画像処理 入力された画像から landmark 領域を抽出する．

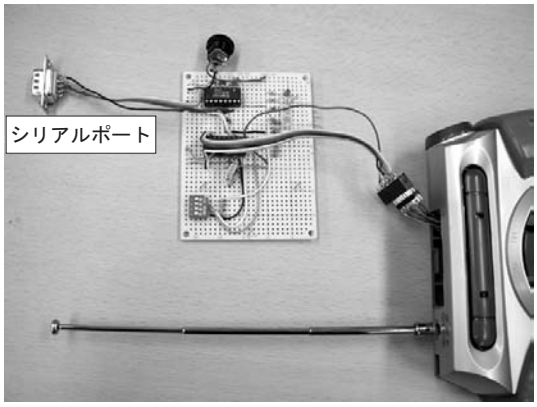

図 3.8: シリアルコンバータ

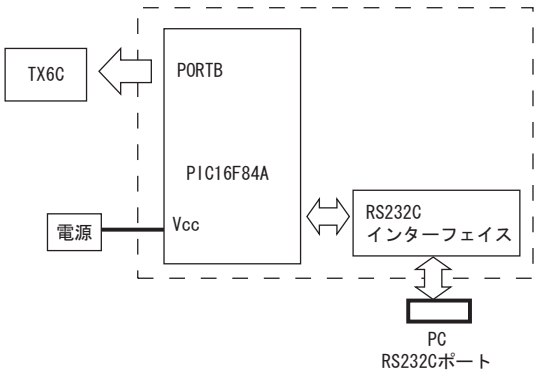

図 3.9: シリアルコンバータの回路構成

landmark 位置検出 求まった landmark の領域から，landmark の大きさとスクリーン上の座標を求める．あらかじめ landmark の情報（大きさ，形）をシステムに登録しておくことで，これらの情報からカメラ座標系内の landmark の位置を計算できる．

自己位置推定 カメラ座標系での landmark の位置情報から，ワールド座標系での飛行ロボットの位置（カメラの位置）を推定する．ワールド座標系での飛行ロボットの位置を求めるためには

- カメラ座標系（ロボット座標系）の landmark の位置
- ワールド座標系の landmark の位置
- カメラ（ロボット）の姿勢

の３つの情報が必要である（詳細は第 6 章の 6.4.2 で説明する）．しかし，本システムでは「カメラの姿勢」情報は使わずに，ロボットの位置に“ 曖昧さ ”を持たせる．この曖昧さを時系列の情報と照らし合わせることで，正確なロボットの位置を推定する．この位置推定についての議論は第 6 章で詳しく述べる．

行動決定 現在のロボットの位置情報を基にロボットの行動命令を決定する．過去に決められた行動

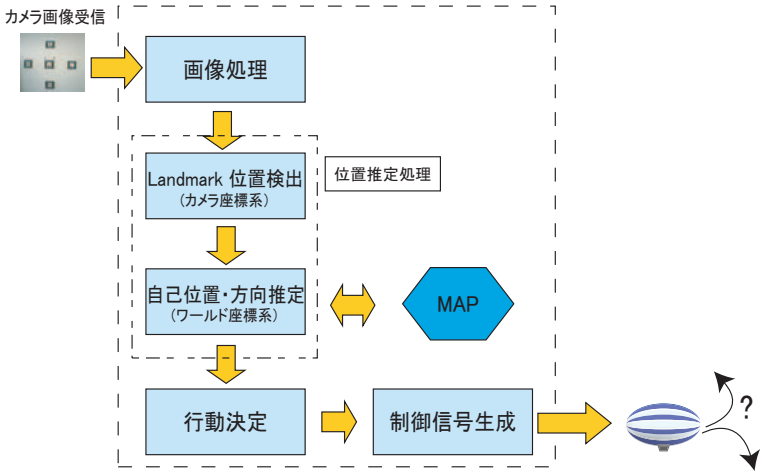

図 3.10: 処理の基本構成

命令から推定される位置と現在位置のズレを求め，その大きさによって，行動命令に修正を与える．行動決定については，第 4 章で詳しく述べる．

制御信号生成 決定された行動命令にあった制御信号を生成し，シリアルポートに出力する．

3.3.2 ARToolKit

ARToolKit[13, 14] は拡張現実感アプリケーション[1] (図 3.11 参照) を容易に開発するための C 言語で書かれたソフトウェアライブラリである．拡張現実感アプリケーションの他にもロボットのナビゲーション [4] にも用いられている．

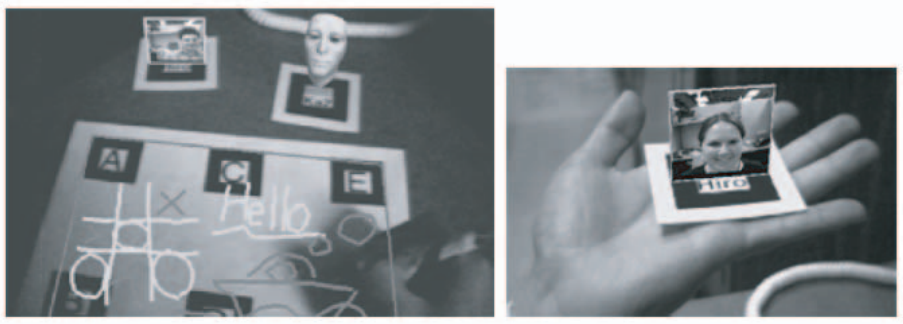

図 3.11: 拡張現実感アプリケーション（[14] より引用）

ARToolKit はカメラから得られた画像情報を基に画像処理を行い，あらかじめ登録されたマーカを検出し，(カメラ座標系での）その 3 次元位置と方向を計算する．マーカには識別可能な独自のパターン (図 3.12 参照) を与えることで，複数の仮想物体の取り扱いにも対応している．

[1] 仮想物体を現実世界の中に投影あるいは対応させることによって，現実に対する知覚を情報的に拡張させたアプリケーション．

図 3.12: ARToolKit で用いるマーカ

ARToolKit によるマーカの位置推定処理を，図 3.13 に示す．処理の流れは

1. 環境に設置されたマーカを検出．
2. マーカの 3 次元位置と方向を計算．
3. マーカの種類を認識．

の順序で行われる．ARToolKit ではこの処理の結果で得られるマーカの種類と位置情報を基に，仮想物体の種類，表示位置，大きさを決定している．

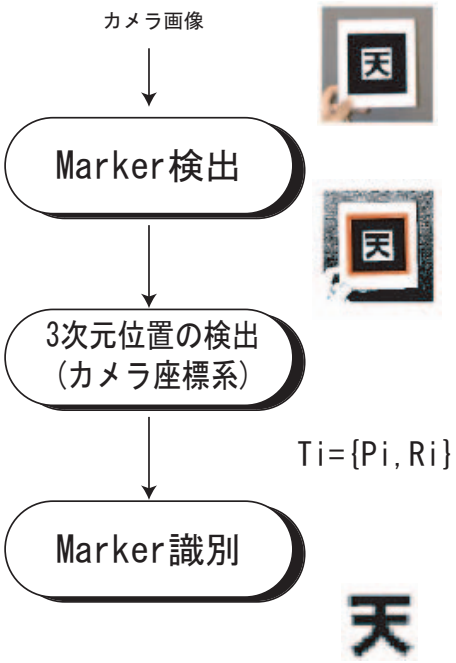

図 3.13: マーカの位置推定処理

3.4 群飛行ロボットへの拡張

前節までは，1 台の飛行ロボットによるシステムの構成について述べた．本節では，その飛行ロボットを複数台用いるアプローチについて述べる．

飛行船の数は建物・部屋の広さや飛行ロボット自体の大きさにも影響するが，現段階では研究室程度の広さに $90cm \times 90cm \times 40cm$ の大きさの飛行ロボットを 2 つ用いる．各飛行ロボットは前節に示したようにカメラ画像により位置を推定し，行動計画と自己位置を比較しながら行動を決定して自律移動をする．しかし，移動に伴いその推定の誤差が積み重なるため，正確な自己位置の判断ができなくなってしまう．この問題を解決するひとつの方法は，センサ類を増やしてより多くの情報からその推定の正しさを判断することである．しかし，飛行船型の飛行ロボットではペイロードの問題から複数のセンサを搭載するのは難しい．そこで，本研究では複数の飛行ロボットが協力して位置を推定する群飛行ロボットのアプローチをとる．飛行ロボットがお互いの情報を共有することで，誤った位置推定に補正をかけるようにするのである．

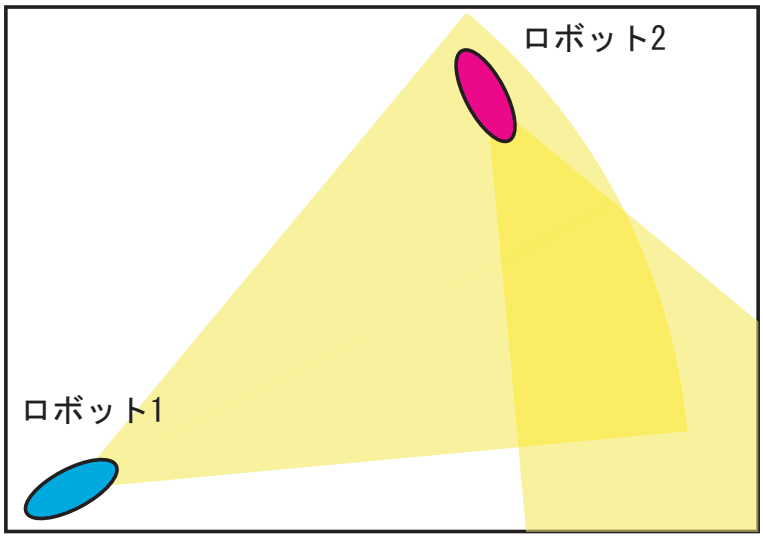

図 3.14: 基本構成

具体的には，飛行ロボットに飛行ロボットの ID となるマーカを貼り，他の飛行ロボットがこのマーカを観測することで，それぞれの飛行ロボット同士の相対位置関係を推定する．例えば図 3.14 のように飛行ロボットが 2 台存在する環境で，飛行ロボット 1 が飛行ロボット 2 を観測したとする．この時，飛行ロボット 1 が 2 台の飛行ロボットの相対位置関係と飛行ロボット 1 の絶対位置を飛行ロボット 2 に送る．飛行ロボット 2 はその情報を受け取り，自己位置推定による結果と比較する．もし飛行ロボット 2 自身の位置推定の結果と大きく異なった場合は，自己位置推定に誤りがある可能性があるので，改めて自己位置推定をする．このようにすることで，お互いの誤った推定を補正するのである．

群飛行ロボットでのシステム構成を図 3.15 に示す．単体の飛行ロボットの場合と異なる点は，各飛行ロボットのためのコンピュータの他に，全体の情報を管理するサーバがあることである．各飛行ロボットからのカメラ画像はそれぞれのコンピュータで処理され，自己位置を検出する．この情報をサーバに送り，サーバはこれらの情報を統合して各飛行船の位置を推定する．この推定結果に基づ

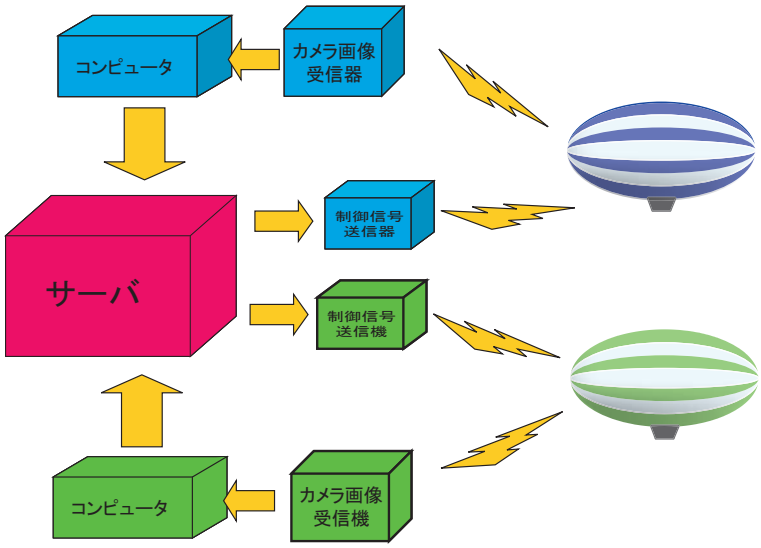

図 3.15: 基本構成

き，各飛行船の行動命令を決定する．

サーバ内のソフトウェアの処理を図 3.16 に示す．それぞれの飛行船は受け取った画像データを処理し，自己位置を検出する．他の飛行ロボットを観測した場合は，その飛行ロボットと自己との相対位置関係を求める．この 2 つの情報をサーバに送信する．サーバではこの情報に基づき，位置推定が行われる．詳細については第 6 章の 6.6 で述べる．得られた位置から各飛行船の行動を決定し，制御信号を生成する．

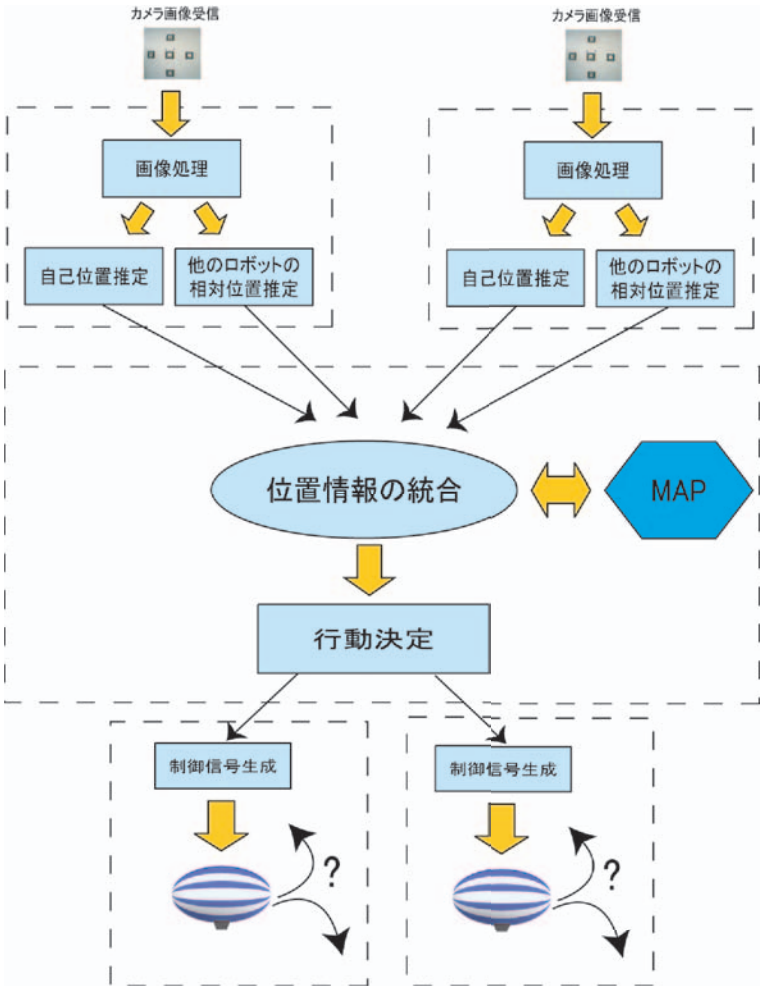

図 3.16: 群飛行ロボットの位置同定システム

第4章　飛行ロボットの行動決定

飛行ロボットをある地点からゴールまで導く場合，ゴールに辿り着くまでの経路を求め，その経路を通るように飛行ロボットの行動を逐次変化させる必要がある．このとき，飛行ロボットは簡単に移動経路を決めることができない．なぜなら，本研究でも用いる飛行ロボットは飛行船型で，その運動はノンホロノミック系である．さらに慣性の影響が大きいことから，その行動は過去の行動命令に大きく影響してしまい,「右に曲がれ」「前へ進め」といった行動命令に対しても，簡単に制御信号を生成することができない．また，飛行ロボットの移動できる空間は 3 次元空間であり，環境には障害物や壁などの制約があるため，これらも考慮して，移動経路とその制御信号を生成する必要がある．そこで本研究では，Rapidly-exploring Random Tree (RRT)[20] を用いて，飛行ロボットの移動経路とその行動を決めることにする．

4.1　Rapidly-exploring Random Tree(RRT)

4.1.1　概要

Rapidly-exploring Random Tree (RRT) とは、比較的最近開発された高次空間でも効率的に探索できるデータ構造（図 4.1 参照）である．特に実時間での経路計画、軌道生成などに使用されている [17, 1, 15].

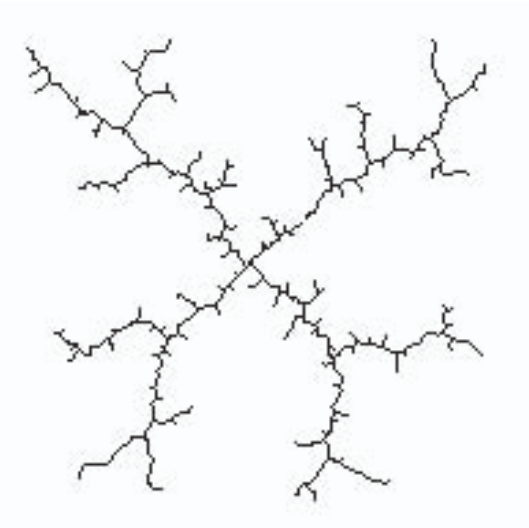

図 4.1: RRT の例

4.1.2 基本アルゴリズム

表 4.1 に基本的な RRT 生成のアルゴリズムを示す。表 4.1 に示したアルゴリズムで RRT が生成さ

表 4.1: RRT 生成アルゴリズム

RRTmain(q_{init})

1. 初期状態 q_{init} によりツリー G を初期化する.
2. 次のアルゴリズムを 1 から K まで繰りかえす.
3. q_{rand} に $RANDOMCONFIG$ で生成されたランダムな状態を代入する.
4. $EXTEND(G, q_{rand})$ によりツリーを伸ばす

EXTEND(G, q_{rand})

1. G 内のノードで q_{rand} に一番近いものを $NEARESTNEIGHBOR(q, G)$ により探し、q_{near} に代入
2. q_{near} から q_{rand} までの長さが ϵ 以下の時は q_{rand} を q_{new} に，ϵ 以上の時は q_{near} から q_{rand} に ϵ だけ進んだ方向に q_{new} を作成する．この時障害物などが原因で、q_{new} が作成できなければ、Trapped を返す.
3. q_{near} と q_{new} を結んで G の一部にする
4. $q_{new} = q_{rand}$ の時は $Reached$ を、その他の時は $Advanced$ を返す

れる過程を，2 次元の場合について説明する．最初に RRT を初期化し，初期ノード q_{ini} を配置する．ロボットの移動経路を考える場合，初期ノードはスタート地点である．次に環境の中へ q_{rand1} をランダムに配置する．この q_{rand} に最も近い RRT のノードを q_{near} とする．始めはノードが q_{ini} しかないので q_{ini} を q_{near} にする．q_{near} から q_{rand1} に向って直線を引き，q_{near} から一定間隔離れた場所に q_{new1} を生成する（図 4.2(a) 参照）．新しく生成した q_{new1} を RRT の新しいノードにする．そしてまた新たに q_{rand2} をランダムに抽出し，図 4.2(b) のようにツリーを延ばす．図 4.2(b) では (a) と同じ q_{ini} が q_{near} に決定され，(c) では (a) で生成された q_{new1} が q_{near} に決定される．RRT は q_{rand} の配置の仕方で，図 4.2(d) のように木が枝分かれしていく．

RRT はその成長の過程にデータの制約を予め考慮して，逐次的にデータ構造を生成することができる．ロボットの移動経路探索問題について言えば，環境内に障害物などがある場合，ロボットが障害物を回避しながら移動するように経路を生成することである．この成長過程を，図 4.3 を用いて説明する．図 4.3(a) では新たな q_{rand3} によって木を成長するところである．この場合は q_{near} に選ばれた q_{new1} と q_{rand3} の間に障害物はないので，新しいノード q_{new3} を生成することができる．しかし，図 4.3(b) の場合では q_{rand4} によって RRT を成長させることができない．q_{near} に選ばれた q_{new2} から q_{rand4} に枝を伸ばそうとしても，その間に障害物があるため q_{new4} は生成することができないのである（もし q_{new4} の生成される位置が障害物を超えなければ q_{new4} を新たなノードとして RRT に追加することができる）．このように障害物を避けながらツリーを成長させることができる．

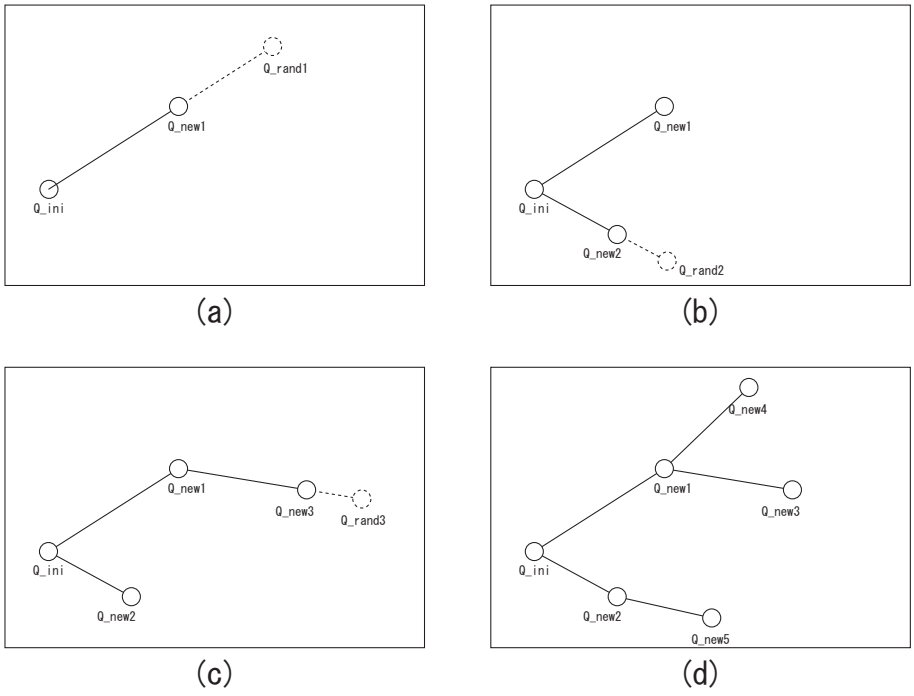

図 4.2: RRT の生成

以上が RRT の基本的なアルゴリズムである．RRT を用いてロボットの経路を考える場合は，q_{rand} を選ぶときに一定の確率でゴールの位置を出現させるようにする．このようにすることでゴールへ辿り着く経路を短時間で探索することができる．

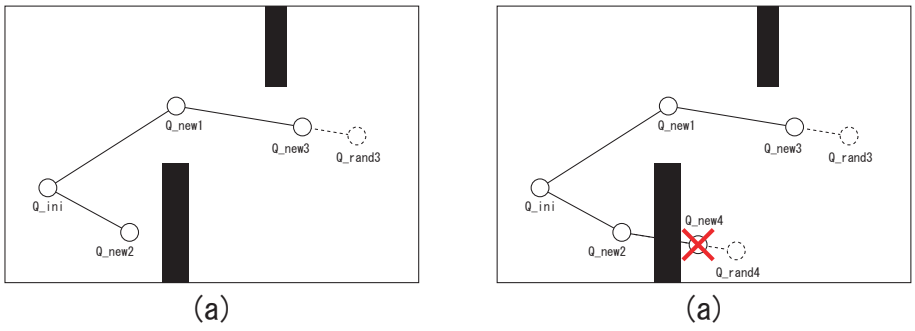

図 4.3: RRT の生成（障害物がある場合）

RRT が効率よく探索できるデータ構造である理由は，次のように説明できる．RRT の各ノードは環境を分割するボロノイ領域の頂点とみなすことができる（図 4.4 参照）．このとき大きなボロノイ領域を持つ頂点は，高い確率で x_{near} として選択される．RRT のノード数が増えていき，データ構造が大きくなると、各ノードが形成するボロノイ領域のサイズは小さくなる．よって一様に素早く特定の状態空間を見つけることが出来る。

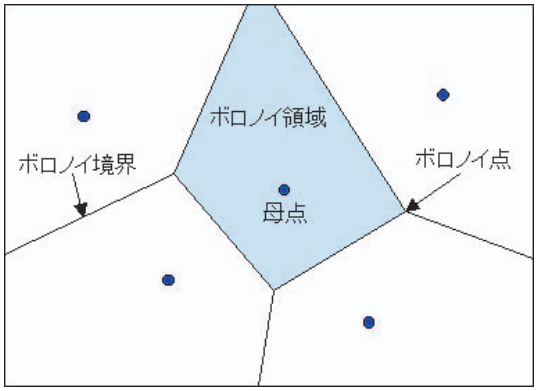

図 4.4: ボロノイ図

4.2 RRT の飛行ロボットへの拡張

本研究で用いる飛行ロボットは飛行船型であるため，その行動には制約がある．従って RRT の基本アルゴリズムをそのまま適用するのではなく以下のように改良を加える．

q_{near} の選択方法

基本アルゴリズムでは RRT の成長させる箇所 q_{near} を，q_{rand} からのユークリッド距離で最短なノードを選んでいた．この「ユークリッド距離が最小」は q_{near} を選択する時のひとつの基準であり，問題によっては他の基準を設けた方が理想的なデータ構造を生成する場合もある．飛行ロボットの移動においても同様のことが言える．なぜなら飛行ロボットの移動において各ノードは時系列の位置を表しており，推進機構の制約からあるノードから別のノードへの移動ができない場合も考えられるからである．つまりノードとノードの「距離が最小」という基準よりも，ノードからノードへの「移動の負荷が最小」になる基準を設けた方がスムーズな移動を可能にする経路を探索できる．そこで本研究では各ノードにロボットの位置に加えてロボットの姿勢（2 次元ではヨー角のみ）を与える．q_{near} を選ぶ際には距離と移動方向と姿勢の差を比較し，総合的に見て最小となるノードを q_{near} として選択する（図 4.5 参照）．

q_{new} の決定方法

q_{new} は，基本アルゴリズムにおいて新しく RRT に追加されるノードであった．ロボットの場合，この新しく決定されるノード q_{new} は行動命令に従って決定される．本研究では図 4.6 のように飛行ロボットが q_{near} においてどの行動をとれば q_{rand} に近づくかを計算し，その行動にあった制御信号と q_{near} を生成する．

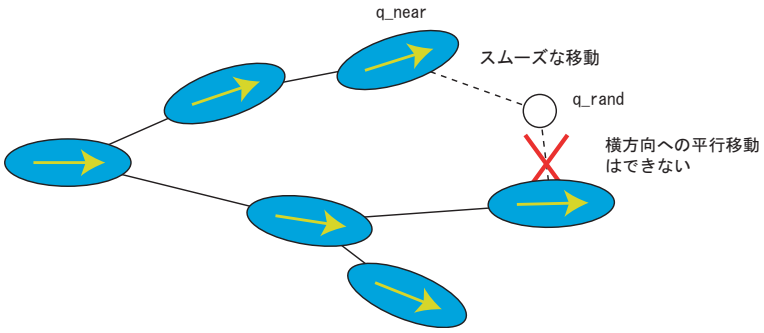

図 4.5: q_{near} の選び方

※　q_{rand} との距離は下のノードの方が小さいが，移動方向とヨー角の差が 90 度程度あるので実際には移動し難い．上のノードは q_{rand} との距離も小さくヨー角と移動方向の差も大きくないので移動し易い．

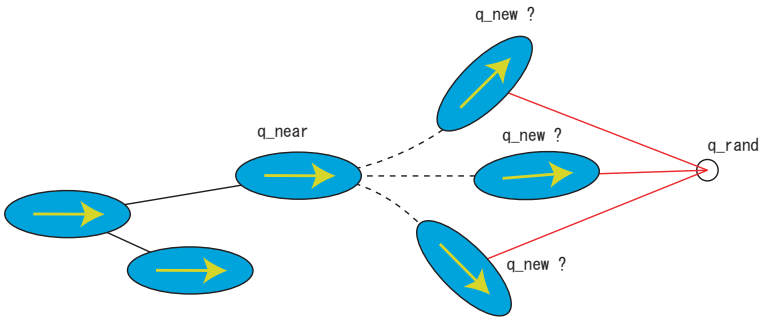

図 4.6: q_{new} の生成方法

第5章　飛行船の力学モデル

本章では飛行船の力学モデルについて述べる．

5.1　座標系

まず飛行船の動座標系（飛行船座標系）と外界の静止座標系（ワールド座標系）を，図 5.1 のように定義する．u, v, w はそれぞれ x, y, z 軸方向への速度を表しており，p, q, r はそれぞれの軸の回転方向の角速度を表している．また α, β, γ は飛行船の姿勢を表すパラメータで，それぞれワールド座標系における動座標系のヨー角（α），ピッチ角（β），ロール角 (γ) を表す．

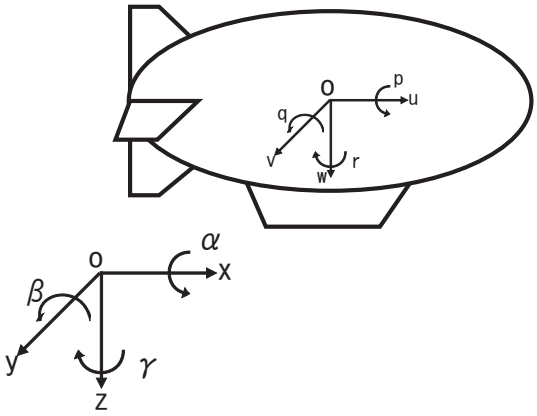

図 5.1: 座標系

5.2　飛行船の力学モデル

飛行船の数学的なモデルをつくるため，先ず飛行船に関わる物理法則を述べる．飛行船の運動に関わる主な物理的な要因には以下のものがある．

- 気体の静力学
- 流体力学
- 質量中心

第１に重要な要因として，気体の静力学を考える必要がある．飛行船の飛行原理は，空気に対して相対的な運動を行う「翼」に生じる「動的な浮力」（揚力）を利用して飛行する飛行機とは異なり，

気体の静力学に基づいている．飛行船の浮力は，アルキメデスの原理により生じる．空気中の飛行船は，飛行船のエンベロープによって排除された空気の質量に等しい力を受ける．このときエンベロープ内に空気より比重の軽い気体（水素ガス，ヘリウムガス，熱空気）をエンベロープに詰めることで,「静的な浮力」を得ることができる．従って飛行スピードには，依存しない力である．

第 2 に流体力学を考慮する．飛行船は移動に伴い，エンベロープによって空気中の流体粒子を移動させる．その結果，流体には運動エネルギーが与えられ，飛行船は移動に伴って抵抗を受ける．この効果は慣性と付加質量を考慮することで説明がつく．飛行船が空気中を飛行することで相対的に質量の大きい空気を移動させるので，この特性は飛行船の飛行モデルを考える上で，特に重要な要因になる．またこの要因により飛行船は従来の物理モデルで考えるよりも，大幅に高い慣性モーメントと質量を持っているように振舞う．

最後に質量中心について述べる．飛行中の飛行船の質量中心を求めるのは難しく，時間と共に変化する．このため，飛行船の運動は図 5.1 に示すようにエンベロープの幾何学的な体積中心で直交する軸から成る系を基準とする．このとき体積中心は飛行船の浮力の中心と一致することも仮定している．

以上を踏まえて飛行船の力学モデルを考える．まず剛体[1]の車両の力学モデルは，式 5.1 に示す 6 次元の自由度を持つニュートン・オイラー方程式で表される．

$$M\dot{\mathbf{x}} + \mathbf{c}(\mathbf{x}) + D(\mathbf{x})\mathbf{x} + g(\alpha, \beta, \gamma) = \tau \tag{5.1}$$

式 5.1 の各項は以下の通りである．

速度ベクトル x

$$\mathbf{x} = \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix} \tag{5.2}$$

速度は 6 次元のベクトルで表され，3 次元の軸方向ベクトル U と 3 次元の角速度ベクトル P によって表現される．これらのベクトルはいずれもワールド座標系を慣性基準においた変量で，また運動方程式は飛行船座標系で表現する．

$$\mathbf{U} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{5.3}$$

から成る．

質量行列 M

質量行列は 6×6 の行列で次式で表される．

$$M \equiv M_{RB} + M_A \tag{5.4}$$

ここで M_{RB} は剛体の質量と慣性モーメントを表す行列であり，M_A は付加質量と付加慣性モーメントを表す．

コリオリ力と遠心力のベクトル $c(\mathbf{v})$

$c(\mathbf{v})$ は 6 次元のベクトルで，コリオリ力と遠心力を含んだ 2 次式で表される．

[1] 簡単のため弾性は考慮しないことにする．

制動行列 $D(\mathbf{v})$

行列 $D(\mathbf{v})$ は，6×6 の行列で空気力学上の制動行列である．制動行列はさらに次式のように分解できる．

$$D(\mathbf{v}) \equiv D_S(\mathbf{v}) + D_H(\mathbf{v}) \tag{5.5}$$

ここで，D_S は薄い乱流境界層が引き起こす摩擦の抵抗を表す．また，D_H は飛行船の艇体の流線型が引き起こす渦の放出による制動を表す．

重力と浮力のベクトル $g(\alpha, \beta, \gamma)$

ベクトル g は重力と浮力による復元力で，６次元のベクトルで表す．ここで重力はワールド座標系で定義されるため，これを飛行船座標系に変換する必要がある．この変換にはロール，ピッチ，ヨーをパラメータとする回転行列が用いられる．

推進ベクトル τ

ベクトル τ は，６次元の推進ベクトルで，剛体に働く全ての力とモーメントを表す．これはさらに次式のように２つの成分にわけられる．

$$\tau \equiv \tau_A + \tau_P \tag{5.6}$$

τ_A は舵面による力で，τ_P は推進力を表す．

5.3 小型飛行船における線形近似

前節で述べた飛行船の力学モデルを数値解析するのは困難であるため，Zwaan らのモデル [34, 35, 31, 23] を参考にして，前節で求めた運動方程式を線形近似した．Zwaan のモデルでは対象とする飛行船が

- 1m 程度の小型飛行船
- 屋内を低速で飛行する
- 気体の揺れが小さい

という仮定の基に一般的な飛行船の運動方程式の線形化を行っている．

飛行船の運動がある平衡状態についての小さな摂動であると考えれば，式 5.1 は

$$M\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \tau(t)) \tag{5.7}$$

と書くことができる．ここで $\mathbf{x}, \tau$ は

$$\mathbf{x}(t) = \tilde{\mathbf{x}}(t) + \mathbf{x}_\delta(t) \tag{5.8}$$

$$\tau(t) = \tilde{\tau}(t) + \tau_\delta(t) \tag{5.9}$$

であり，平衡状態 $\tilde{\mathbf{x}}(t), \tilde{\tau}(t)$ 周りの小さな摂動 $\mathbf{x}_\delta(t), \tau_\delta(t)$ を用いて表現できる．この平衡状態では次式を得ることが出来る．

$$\dot{\mathbf{x}}(t) = A\mathbf{x}_\delta(t) + B\tau_\delta(t) \tag{5.10}$$

ここで A, B は平衡状態における $f(\mathbf{x}(t), \tau(t))$ の 1 次のテイラー展開から得られるヤコビアン行列である．

$$A = \frac{\partial f}{\partial \mathbf{x}}(\tilde{\mathbf{x}}(t), \tilde{\tau}(t)) \tag{5.11}$$

$$B = \frac{\partial f}{\partial \tau}(\tilde{\mathbf{x}}(t), \tilde{\tau}(t)) \tag{5.12}$$

このとき小さい摂動を仮定しているので，各変数の積や二乗は無視できるほど小さくなる．よって式 5.1 におけるコリオリ力と遠心力の項 $\mathbf{c}(\mathbf{v})$ は無視できる．

また式 5.1 の他の項については次のように簡単化することができる．

質量行列 M

ゴンドラを質点として考えた場合，Z 軸に沿ってできる X-Z 平面と Y-Z 平面は，質量分布において対称的な平面になる．このとき慣性モーメントの行列はクロスカップリングの成分 (I_{xy}, I_{xz} など）はゼロになり，対角行列となる．また付加質量と付加慣性モーメントを表す M_A も対角成分のみとなり，

$$M_A = \begin{bmatrix} A_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & A_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & A_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & A_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & A_{66} \end{bmatrix} \tag{5.13}$$

の対角行列に近似できる．

制動行列

飛行船が小型で低速である場合，乱流境界層が線形の表面摩擦係数のみで表現できると仮定する．このとき制動行列は次式の対角行列に近似できる．

$$M_A = \begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_w & 0 & 0 & 0 \\ 0 & 0 & 0 & K_p & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r \end{bmatrix} \tag{5.14}$$

推進ベクトル

低速・小型飛行の場合，船舵面による力の効果はなくなるので $\tau = 0$ になる．この場合，推進力はプロペラの幾何学的な配置のみ（図 5.2 参照）を考慮した関数で表すことができる．

$$\tau = \begin{bmatrix} T_{cmn} \\ 0 \\ T_v \\ 0 \\ d_z \cdot T_{cmn} \\ d_y \cdot T_{diff} \end{bmatrix} \tag{5.15}$$

ここで T_{cmm} はプロペラの左右の出力 T_s, T_p の合力 $T_{cmn} = T_s + T_p$ である．また T_{diff} は左右のプロペラの出力の差 $T_{diff} = T_s - T_p$ である．T_v は Z 軸方向への推進力を表す．d_z と d_y は図 5.2 で示すように体積中心からプロペラまでの距離を表す．

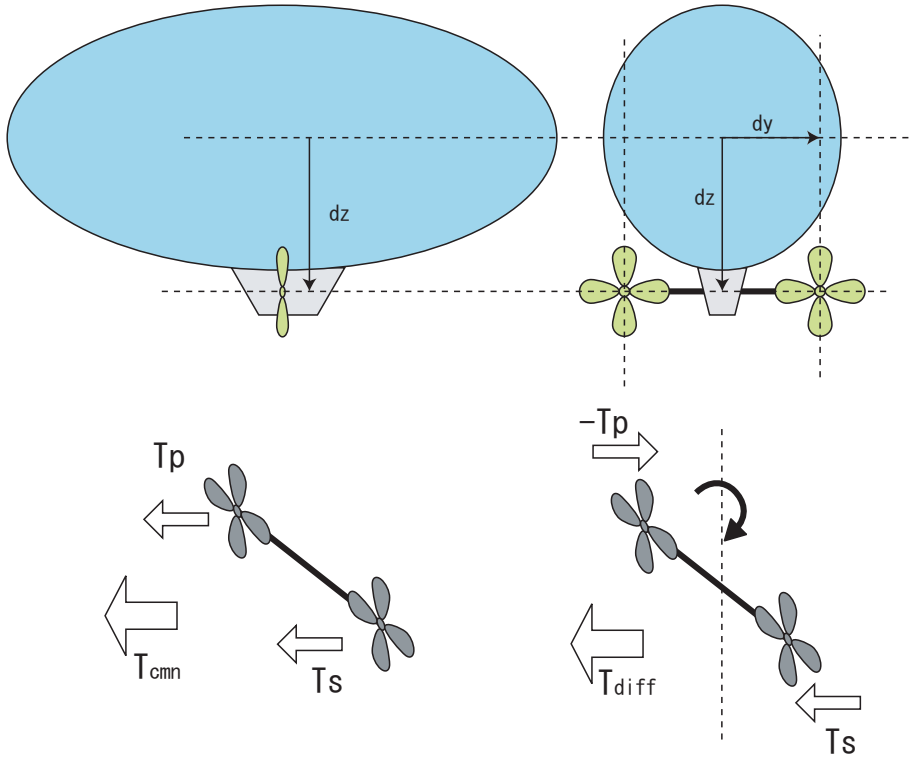

図 5.2: プロペラの配置と推進力

以上を踏まえて小型飛行船の力学モデルをまとめる．$c(\mathbf{x}) = 0$ が成り立つことにより，力学モデルを x－z 平面、x－y 平面状での関係式に分離できることを考慮する．また，飛行船の操作においてロール角とピッチ角の変化は微小であるので，$\beta = \gamma = 0$ と仮定する．

5.3.1 X-Z 平面における運動方程式

X-Z 平面上の状態ベクトルと入力ベクトルは以下の通りである．

$$\begin{aligned} \mathbf{x} &= \left[\begin{array}{cccc} \delta u & \delta w & \delta q & \delta\beta \end{array} \right]^T \\ \tau &= \left[\begin{array}{cc} \delta T_{cmn} & \delta T_v \end{array} \right]^T \end{aligned}$$

ここで運動方程式で扱う行列を正方行列にして扱いやすくするため，状態ベクトルにピッチ角 β を取り入れ，浮力と重力の関係式を下記の方程式に含ませた．

$$M\dot{\mathbf{x}} = A\mathbf{x} + B\tau \tag{5.16}$$

$$
\begin{aligned}
M &= \begin{bmatrix} m+A_{11} & 0 & ma_z & 0 \\ 0 & m+A_{33} & -ma_x & 0 \\ ma_z & -ma_x & I_{yy}+A_{55} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A &= \begin{bmatrix} -X_u & 0 & 0 & -(mg-f_b) \\ 0 & -Z_w & 0 & 0 \\ 0 & 0 & -M_q & -a_z mg \\ 0 & 0 & 1 & 0 \end{bmatrix} \\
B &= \begin{bmatrix} 1 & 0 & d_z & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^T
\end{aligned}
$$

5.3.2 X-Y 平面上の運動方程式

X-Y 平面上の状態ベクトルと入力ベクトルは以下の通りである.

$$
\begin{aligned}
\mathbf{x} &= \begin{bmatrix} \delta v & \delta p & \delta r & \delta\gamma \end{bmatrix}^T \\
\tau &= \delta T_{diff}
\end{aligned} \tag{5.17}
$$

X-Z 平面の場合と同様で，運動方程式で扱う行列を正方行列にして扱いやすくするため，状態ベクトルにロール角 γ を取り入れ，浮力と重力の関係式を下記の方程式に含ませた.

$$
\begin{aligned}
M\dot{\mathbf{x}} &= A\mathbf{x} + B\tau \\
M &= \begin{bmatrix} m+A_{22} & -ma_z & ma_x & 0 \\ -ma_z & I_{xx}+A_{44} & 0 & 0 \\ ma_x & 0 & I_{zz}+A_{66} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A &= \begin{bmatrix} -Y_v & 0 & 0 & mg-f_b \\ 0 & -K_p & 0 & -a_z mg \\ 0 & 0 & -N_r & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\
B &= \begin{bmatrix} 0 & 0 & d_y & 0 \end{bmatrix}^T
\end{aligned} \tag{5.18}
$$

運動方程式中の各パラメータの意味は下記の通りである.

- m：飛行船の質量
- g：重力加速度
- f_b：浮力
- a_x, a_z：体積中心からプロペラまでの距離
- I_{xx}, I_{yy}, I_{zz}：それぞれ動座標系の X 軸、Y 軸、Z 軸周りの慣性モーメント

- $\tau_{cmm}, \tau_v, \tau_{diff}$: 順に動座標系の X 軸、Y 軸、Z 軸の推進力
- δ : 比例定数

これらのパラメータは重量、長さ等の実測値によって決定し、空気抵抗などの直接測定不可能なものについては、近似値によって決定する．また上記の運動方程式により求められる位置座標は動座標系であるため、静止座標系に変換する必要がある．

第6章　飛行ロボットの位置推定

6.1　座標系の取り扱い

位置推定で扱う座標系を，始めに定義しておく．位置推定を考える場合には，第5章で説明した全ての座標系の基準となるワールド座標系と，飛行ロボットを中心とするロボット座標系の他に，センサを中心とするセンサ座標系が存在する．本研究では，センサとしてカメラのみを用いるのでセンサ座標系をカメラ座標系と呼ぶことにする．

これらの座標系の関係は図6.1に示される．ここで，全ての座標系のZ軸は平行で正方向も同じとする．これは飛行ロボットに飛行船を採用しているためX軸回りの回転(ロール)，Y軸周りの回転（ピッチ）がほとんどないことを考慮して計算上の簡単化を図っている．

カメラ座標系とロボット座標系においては相対的な回転はないものとする．よって飛行ロボットの進行方向とカメラの視線方向が一致する．また，明らかにカメラ座標系からロボット座標系への変換は平行移動のみとなる．

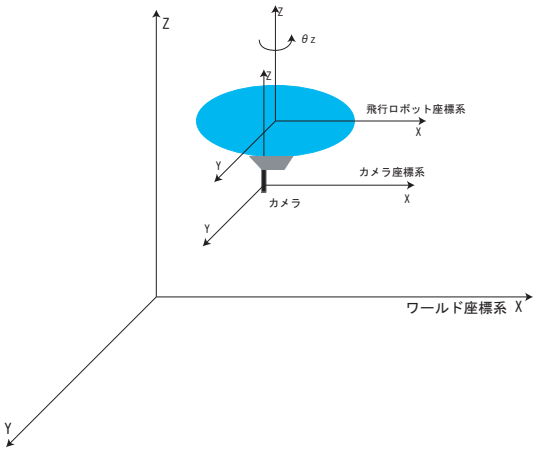

図 6.1: 飛行ロボットの各種座標系

6.2　確率理論に基づく位置推定法

一般的に現実世界のロボットは，行動に伴って自己位置推定の数値計算などに誤差を蓄積していく．この誤差が蓄積されていくと，ロボットは自身の状態を正しく判断できなくなってしまう．誤差が生じる原因は，ロボットの行動やセンサ情報にノイズが含まれているためで，これをゼロにすることは困難である．そこで，ロボットの状態を図6.2に示すように確率分布で表す．ロボットの状態を

確率分布で表すことによって，ノイズから生まれる状態の“ 不確かさ ”を表現できる．

このような確率論に則った位置推定法としては，主にベイジアンフィルタがよく用いられる．

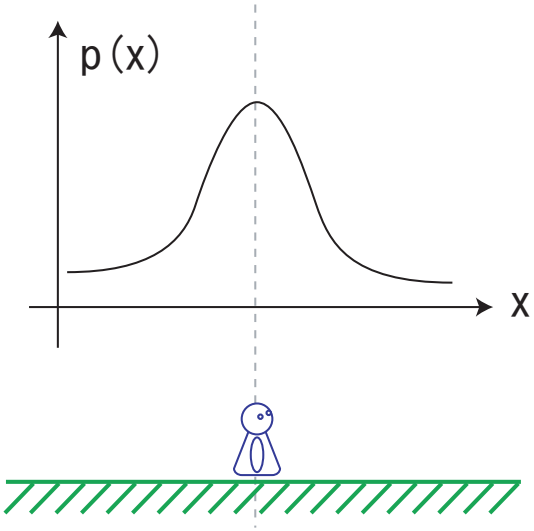

図 6.2: 位置の確率表現

6.2.1 ベイジアンフィルタを用いた位置推定法

ベイジアンフィルタとはベイズ推定[1]に基づくフィルタである．ベイジアンフィルタを用いることで，センサデータから“ ダイナミックシステム ”の“ 状態 ”を推定することが可能である．これをロボットの位置推定問題に当てはめると，“ ダイナミックシステム ”はロボットとロボットのまわりの環境を表し，“ 状態 ”はロボットの位置と姿勢を表す．またセンサデータとしてはレンジデータ[2]やカメラ画像，オドメトリのデータが用いられる．ベイジアンフィルタでは“ 環境がマルコフ ”[3] であることを仮定している（この仮定は後の説明で利用する）．

ベイジアンフィルタの原理は，データを束縛条件として状態空間全体の確率密度関数を見積もることである．この結果得られる事後確率は“ 信念 ”(belief) と呼ばれ，次式で表現される．

$$Bel(x_t) = p(x_t|d_{0,\ldots,t}) \tag{6.1}$$

ここで x はロボットの状態を表し，x_t は時間 t における状態である．また $d_{0\ldots t}$ は時間 0 から t までに得られたデータを表す．ロボットの場合，データには次の 2 種類のデータが存在する．

観測データ レンジデータやカメラ画像によるデータ

行動データ オドメトリなどのロボットの動きに関するデータ

[1]ベイズ推定とは「過去から現在までの事象を用いて未来を予測する」ことで，近年さまざまな研究で利用されている．
[2]レンジファインダ（対象物にレーザを照射することで対象物との距離を測る装置）によって得られた距離画像．
[3]一般的にマルコフ過程と呼ばれる．マルコフ過程とは「未来の事象は現在の状態のみに依存する（過去の事象は独立）.」という確率過程である．

観測データ o と行動データ a を用いて式 6.1 を表現すると次式になる．

$$Bel(x_t) = p(x_t|o_t, a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0) \tag{6.2}$$

ここで観測データと行動データは交互に連続して入力されるものとする．また行動データ a_{t-1} は $t-1$ から t の間に起きた行動を表す．

ベイジアンフィルタでは，この信念を再帰的に計算していく．信念の初期値はシステムの初期状態に依存する．もしシステムの初期状態がわからなければ，状態空間全体に渡って一様分布を与える．ロボットの場合，初期値に一様分布を与えるということは，ロボットの初期位置が未知であることを表すので Global Localization Problem に当てはまる．

次に再帰的に信念を計算するために更新則を導く．先ずベイズの定理を用いて，式 6.2 を変形する．

$$Bel(x_t) = \frac{p(o_t|x_t, a_{t-1}, \ldots, o_0)p(x_t|a_{t-1}, \ldots, o_0)}{p(o_t|a_{t-1}, \ldots, o_0)} \tag{6.3}$$

式 6.3 の分母は x_t に関して一定なので，式 6.3 は次式で表すこともできる．

$$Bel(x_t) = \eta p(o_t|x_t, a_{t-1}, \ldots, o_0)p(x_t|a_{t-1}, \ldots, o_0) \tag{6.4}$$

ここで η は正規化定数であり次式で表現される．

$$\eta = \frac{1}{p(o_t|a_{t-1}, \ldots, o_0)} \tag{6.5}$$

次にベイジアンフィルタがマルコフ過程に従う（つまり未来のデータは過去のデータに依存せず，現在のデータによって決まる）ことを利用する．マルコフ過程を数式で表すと次式になる．

$$p(o_t|x_t, a_{t-1}, \ldots, o_0) = p(o_t|x_t) \tag{6.6}$$

従って式 6.4 は次式のように簡略化される．

$$Bel(x_t) = \eta p(o_t|x_t)p(x_t|a_{t-1}, \ldots, o_0) \tag{6.7}$$

式 6.7 の右辺の $p(x_t|a_{t-1}, \ldots, o_0)$ の項を時間 $t-1$ の状態についての積分表示で表すと，式 6.7 は次式のように拡張される．

$$Bel(x_t) = \eta p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1}, \ldots, o_0)p(x_{t-1}|a_{t-1}, \ldots, o_0)dx_{t-1} \tag{6.8}$$

ここでもう一度マルコフ過程を利用して式 6.8 の右辺の項 $p(x_t|x_{t-1}, a_{t-1}, \ldots, o_0)$ を簡略化する．

$$p(x_t|x_{t-1}, a_{t-1}, \ldots, o_0) = p(x_t|x_{t-1}, a_{t-1}) \tag{6.9}$$

式 6.9 を式 6.8 に代入する．

$$Bel(x_t) = \eta p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1})p(x_{t-1}|a_{t-1}, \ldots, o_0)dx_{t-1} \tag{6.10}$$

最後に，式 6.10 中の $p(x_t|a_{t-1}, \ldots, o_0)dx_{t-1}$ を $Bel(x_{t-1})$ と置き換えて[4]次式を得る．

$$Bel(x_t) = \eta p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1})Bel(x_{t-1})dx_{t-1} \tag{6.11}$$

[4] $Bel(x_{t-1}) = p(x_{t-1}|o_{t-1}, a_{t-2}, o_{t-2}, a_{t-3}, \ldots, o_0)$ であり，$p(x_{t-1}|a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0)$ と比較すると a_{t-1} が余分になり一致しないが，a_{t-1} は x_t にのみ依存するデータであり x_{t-1} に関しては独立である（x_{t-1} の状態はこれから（$[t-1, t]$ の間）起きる行動 a_{t-1} には無関係）．よって $p(x_{t-1}|o_{t-1}, a_{t-2}, o_{t-2}, a_{t-3}, \ldots, o_0) = p(x_{t-1}|a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0)$ となる．

式 6.11 はベイジアンフィルタの更新則である．この式 6.11 を用いて再帰的に信念 $Bel(x_t)$ を計算するには，２つの条件付確率を求める必要がある．１つは $p(x_t|x_{t-1}, a_{t-1})$ で，行動モデルにより決定される．もうひとつは $p(o_t|x_t)$ で，これは観測モデルによって求められる．両方のモデルは時不変性が成り立つので，一般的には表記を簡単化した $p(o|x), p(x'|x, a)$ を用いる．

6.2.2 確率論的モデル

6.2.1 で示したベイジアンフィルタを用いてロボットの位置推定を行うためには，行動モデルと観測モデルの性質を知る必要がある．本節はこれらのモデルを確率論的に扱う方法について説明する．

行動モデル

行動モデルは，ロボットの運動方程式によって表現される．本研究で取り扱うロボットは飛行船型の飛行ロボットであり，一般の地上ロボットの運動モデルとは異なる．このため詳細な説明は，第 5 章で説明する．

ロボットの行動モデルは行動命令 a を入力として持ち，ある状態 x を初期値にして，a を入力した時の出力を運動方程式を解いて計算している．出力として得られるのが次の状態 x' である．しかし，この結果は一般的に理想的な環境の下で，センサノイズや行動の誤差がない場合の結果である．6.2 でも述べたように現実世界のロボットではロボットの行動に伴い，誤差が積み重なる．よって，実際には運動方程式から得られる状態 x' に至らないこともある．つまり，行動モデルに“ 不確かさ ”を取り入れた確率論的行動モデルを生成する必要がある．状態の不確かさを表現するために，次状態を事後確率密度分布として表現する．具体的には運動方程式によって得られる次状態に，０平均のガウス密度関数を加えて表現することが多い．

観測モデル

観測モデルは行動モデルと同様で，環境の変化やセンサの物理的な誤差によって生じるノイズを考慮する必要がある．そこで観測モデル $p(o|x)$ の計算を次の３段階に分けて行う．ある状態 x において観測データ o を取得したとする．

- 理想的なセンサモデルの場合について，観測データが o である時の位置 x を計算する．
- センサのノイズモデルからノイズの分布を作成する．
- 理論値とノイズ分布を組み合わせて $p(o|x)$ を作成する

6.3 Monte Carlo Localization

ベイジアンフィルタでは，式 6.11 を繰り返し計算することで位置推定が行われていた．6.11 の具体的な計算手法としては，状態空間を分割して離散値モデルとして取り扱う Markov Localization などが挙げられる．しかし，実際のシステムを構築するにあたって，式 6.11 を計算するのは簡単ではない．Markov Localization の場合では状態空間の分割数に応じて計算時間が大きくなるため，オンライン処理は極めて困難である．

Monte Carlo Localization ではこの問題を解決するために，ロボットの状態を表す信念 $Bel(x)$ をm個の重み付けされたサンプルの集まり（サンプルセット）によって表現する．

$$Bel(x) \approx \{x^{(i)}, w^{(i)}\}_{i=1,\ldots,m} \tag{6.12}$$

ここで $x^{(i)}$ は状態 x のサンプルで，ロボットの採りえる状態を示す．また $w^{(i)}$ はサンプルの重要度を表し，正の数で表現される．ここで

$$\sum_i w^{(i)} = 1 \tag{6.13}$$

である．$w^{(i)}$ は“ サンプルの重み付け ”によって更新されていく．このときサンプルの分布によって $Bel(x)$ の確率分布が表現される（つまり，サンプルが集中している位置にロボットが高い確率で存在する）．このように $Bel(x)$ をサンプルセットで表現することで，ロボットの状態の不確かさを表現する (図 6.3 参照).

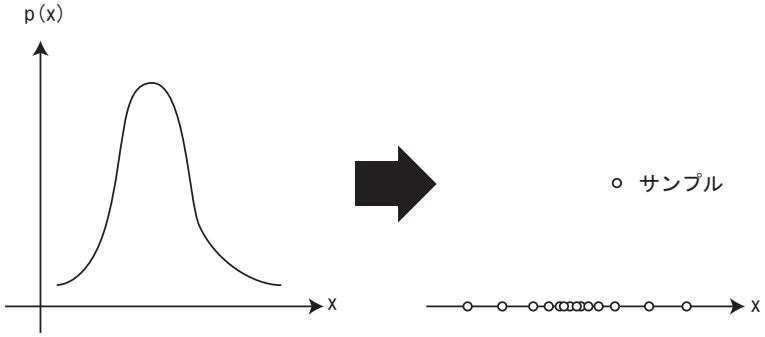

図 6.3: サンプルによる位置確率の表現

図 6.4 はロボットが移動した時のサンプルセットの状態を表したものである．移動に伴いサンプルが散乱していくことが分かる（つまり確率分布が広がっていく）．

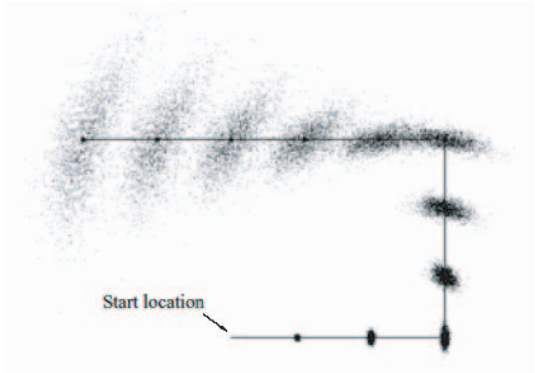

図 6.4: サンプリング手法による位置確率推移の様子（[6] より引用）

Monte Carlo Localization の処理の流れは，図 6.5 に示すパーティクルフィルタの原理に沿って行われる．具体的な計算方法は，以下の通りである．

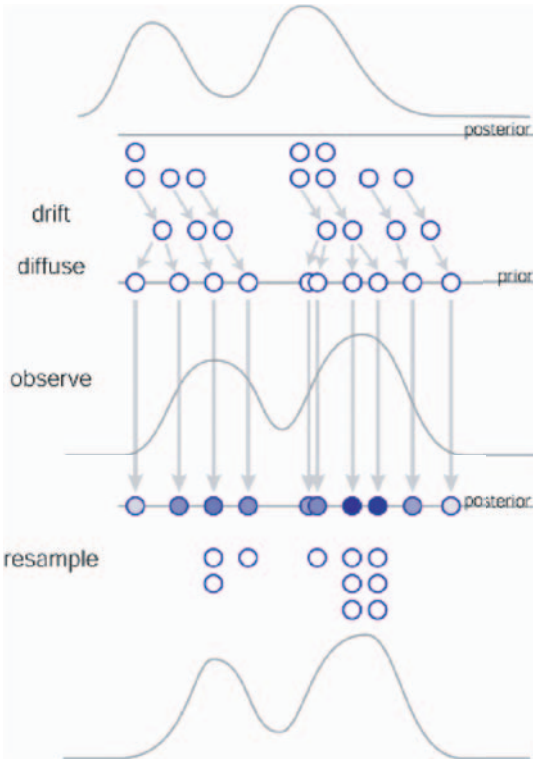

図 6.5: パーティクルフィルタの原理

- 時間 $t-1$ での事後確率分布 $Bel(x_{t-1})$ から重み付けされたサンプル $x_{t-1}^{(i)}$ を抽出する．それぞれのサンプルは，$Bel(x_{t-1})$ に従って分布する．
- サンプル x_{t-1} に行動命令 a を作用させて，時間 t での状態 $x^{(i)}$ を生成する．このサンプルは明らかに

$$q_t := p(x_t|x_{t-1}, a_{t-1}) \times Bel(x_{t-1}) \tag{6.14}$$

に従って分布する．この分布を事前確率密度分布と呼ぶ．
- 観測データから $w^{(i)}$ を更新する．

$$w^{(i)} = p(o_t|x_t^{(i)}) \tag{6.15}$$

- $w^{(i)}$ を正規化する．

以上の処理を全てのサンプルについて，繰り返し計算する．従って，Monte Carlo Localization のアルゴリズムは表 6.1 のようになる．

6.4 2次元画像からの位置推定

本研究では，前節までに述べた確率論的な位置推定をシステムに採用する．また，本研究ではカメラからの 2 次元画像を，観測データとして用いる．本節では，観測データをどのように位置推定で利用するかについて，その原理と問題点について述べる．

表 6.1: Monte Carlo Localization アルゴリズム

Algorithm MCL(X, a, o):

$X' = \emptyset$

for $i = 0$ to m do

generate random x from X according to $w_1, \ldots, w_m$

generate random $x' \sim p(x'|a, x)$

$w' = p(o|x')$

add $< x', w' >$ to X

endfor

normalize the importance factors w' in X'

return X'

6.4.1 問題設定

具体的な処理を述べる前に，先ず本研究で取り扱う問題の整理をする．

ロボット 飛行船型のロボットを用いる．飛行船型ロボットは飛行機型の飛行ロボットに比べて自由度は低く，X 軸方向,Y 軸方向,Z 軸方向, ヨー角の合計４つの自由度を持っている（ロール角とピッチ角は，ほぼ0と見なす）．

センサ 飛行船のペイロードの制約から，１台のカメラのみを用いる．よって，一度に取得可能な観測データとしては１枚のカメラ画像のみとなる．

環境情報 屋内利用を考えるため，予め大まかな環境情報は取得可能である．本研究では環境（部屋）の形，大きさ，障害物の有無を与える．また，位置推定のために landmark を配置し，landmark の形（種類），位置，姿勢，大きさを与える．

6.4.2 単一画像からの自己位置検出

視覚情報を用いた位置推定法では２枚の画像を用いたステレオ視によるものが多いが，本研究では 6.4.1 の制約から単一の画像のみを用いて行う．

先ず landmark が観測されたカメラ画像 (図 6.6 参照) から，カメラ座標系における landmark の位置を検出する．

3.3.2 で述べたように，本研究では landmark の位置検出のために ARToolKit を用いている．ARToolKit による landmark の位置検出手法は，[14] を参照されたい．ARToolKit から得られた landmark の位置を $(x, y, z) = (x_c, y_c, z_c)$，landmark の姿勢（カメラとの相対的な角度) を $(yaw, pitch, roll) = (\alpha, \beta, \gamma)$ とおく．ここで計算を簡単にするため，6.4.1 の制約から $\beta = 0, \gamma = 0$ として，姿勢は α のみを用いて計算する．

カメラ座標系とワールド座標系の関係は，図 6.7 に示すように飛行船の位置と姿勢によって決まる．図 6.7 から，ワールド座標系におけるマーカの位置 (x_w, y_w, z_w) をカメラ座標系のパラメータを用い

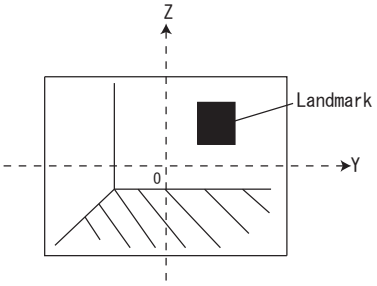

図 6.6: カメラ画像でのランドマーク

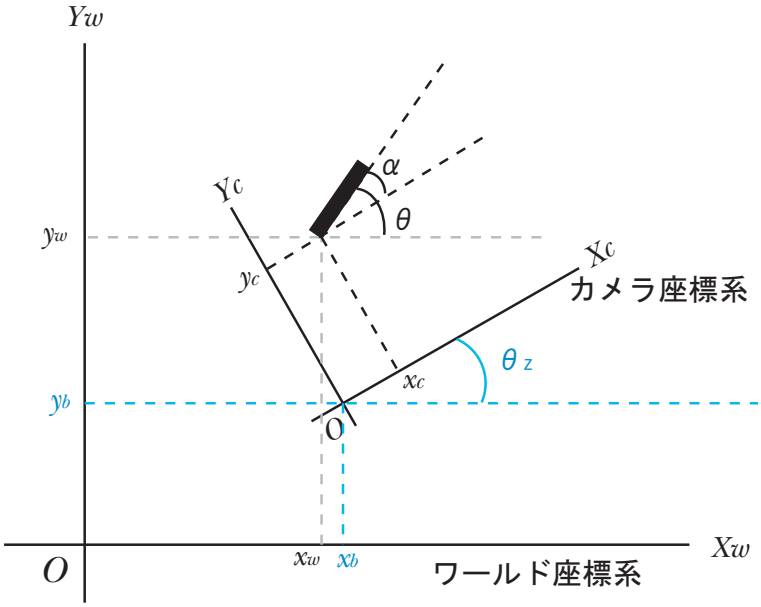

図 6.7: カメラ座標系からワールド座標系への変換

て表現すると，次式が得られる．

$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = R(\theta_z) \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} + \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} \tag{6.16}$$

式 6.16 において，x_b, y_b, z_b はワールド座標系におけるカメラの座標，θ はワールド座標系における landmark の姿勢[5]を表す．また R はワールド座標系におけるカメラの回転を表す 3×3 の回転行列である．回転成分がヨー角 θ_z のみの場合，回転行列 R は次式で表される．

$$R = \begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.17}$$

[5]計算を簡略化するため，飛行船の姿勢と同様にピッチ角，ロール角は 0 とみなす．

式 6.16 からワールド座標系におけるカメラの位置は，次式で求められる．

$$\begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = -R(\theta_z) \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} + \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} \tag{6.18}$$

6.4.1 で述べたように landmark のワールド座標 (x_w, y_w, z_w) とヨー角 α は予め与えられている．よって，ワールド座標系におけるカメラの位置は，カメラ座標系のランドマークの位置 (x_c, y_c, z_c) とカメラのヨー角 θ_z に依存する．図 6.7 から分かるように，カメラのヨー角は $\theta_z = \theta - \alpha$ となる．つまりカメラのヨー角はカメラ座標系における landmark のヨー角 α に依存する．

もしカメラ画像にノイズがあり，カメラ座標系における landmark の位置とヨー角に誤差が生じた場合，ワールド座標系におけるカメラの位置は図 6.8, 図 6.9 のように本来の位置からずれる．

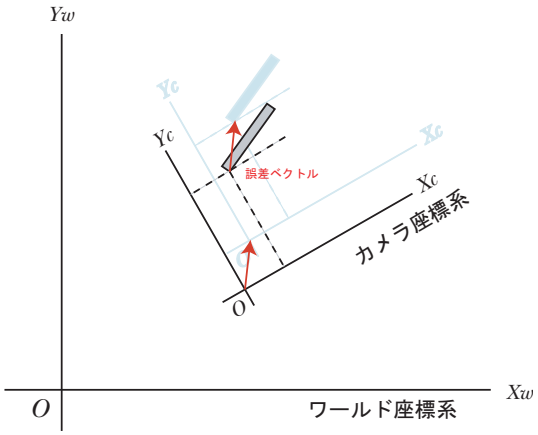

図 6.8: landmark の位置の誤差による影響

landmark の位置の誤差ベクトルは，そのままワールド座標系におけるカメラの位置の誤差ベクトルとして表れる．しかし，ヨー角の誤差は図 6.9 に示すように特に landmark とカメラの相対距離が長くなった時に，カメラ位置に大きな影響を与える．

実際にカメラと landmark の距離が長くなれば，画像中の landmark も小さく表示されるので，landmark の位置とヨー角の誤差は大きくなり易い．また，本研究で用いるカメラはペイロードの制約からカメラの精度を十分に保つことができないので，この誤差は大きく表れる．

図 6.10 は本研究で用いるカメラを利用して，ARToolKit のヨー角の精度を表したものである[6]．図 6.10 を見て分かるようにヨー角の 0 度付近では，最大 20 度以上の誤差がある．これは 0 度付近で，画像中の landmark の位置と大きさの変化が小さいためである．

6.5 円環による位置確率分布

ヨー角が定まらない場合，飛行ロボットは図 6.11 に示すように，landmark を中心とした円周上のどこかに存在することは分かる．本研究ではこれを利用して，飛行ロボットの位置を landmark を中

[6]ARToolKit は拡張現実感アプリケーション用のマーカ追跡のために用いられるので，比較的近距離（1 m程度）のマーカ（landmark）検出を得意としている．よって ARToolKit が姿勢の検出を出来ないわけではない．

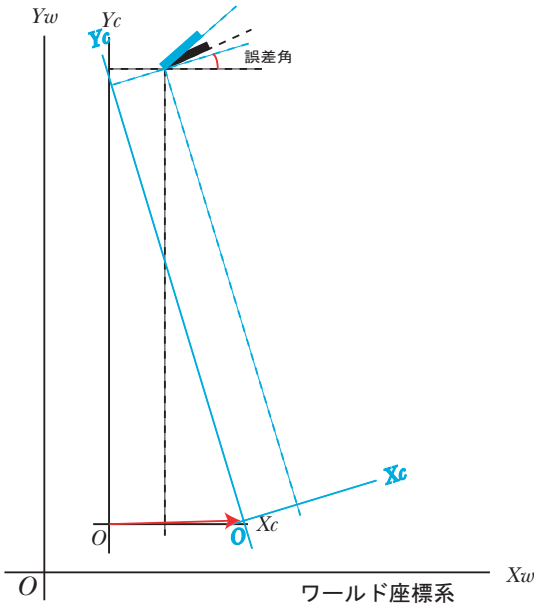

図 6.9: landmark のヨー角の誤差による影響

心とした円の円周の近傍に分布するサンプルで表現する．従ってサンプルセットは図 6.12 に示すような円環状の分布になる．

観測データが得られた時に，$(x^{(i)}, y^{(i)}, z^{(i)})$ に位置するサンプル i の重要度 w は次式で更新される．

$$w(x^{(i)}, y^{(i)}, z^{(i)}) = P(r - r_o, z^{(i)} - z_o) \tag{6.19}$$

P は，$(0, 0)$ を平均とする確率密度関数である．一般的には，2 次元ガウス密度関数 N を用いる．r, r_o, z_o は以下の観測データを用いて求められる（図 6.13,6.14 参照）．

観測データ

- ワールド座標系の landmark の位置 (x_{mw}, y_{mw}, z_{mw})
- カメラ座標系の landmark の位置 (x_{mc}, y_{mc}, z_{mc})

r, r_o は

$$r = \sqrt{(x^{(i)} - x_{mw})^2 + (y^{(i)} - y_{mw})^2} \tag{6.20}$$

$$r_o = \sqrt{x_{mc}^2 + y_{mc}^2} \tag{6.21}$$

として与えられる．また z_o に関しては図 6.14 の位置関係になり，

$$z_o = z_{mw} - z_{mc} \tag{6.22}$$

として与えられる．

本来サンプルの重要度の計算にはサンプルの状態 (x, y, z, θ_z) の全てを用いるべきであるが，本研究では (x, y, z) の 3 変数のみを用いている．これは θ_z を用いた場合，観測データと一致するサンプルが極端に減少して，ほとんどのサンプルは重みが 0 になり，結果的に自己位置を見失う可能性が大

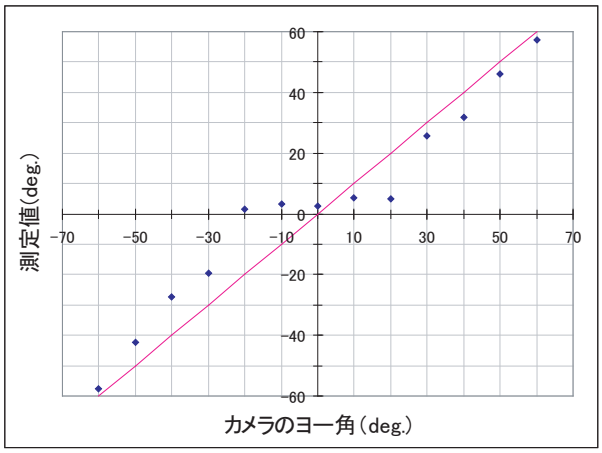

図 6.10: ARToolKit のヨー角の精度
カメラと landmark の距離を 3m に保ち，landmark が画像中心にくるようにして測定.

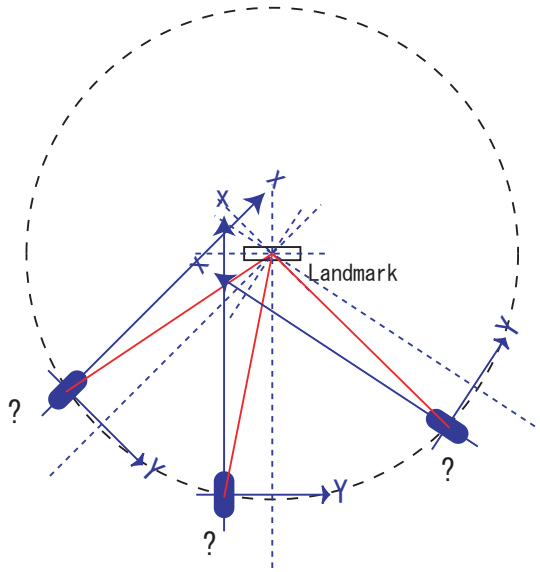

図 6.11: ヨー角が定まらない場合の飛行ロボットの位置

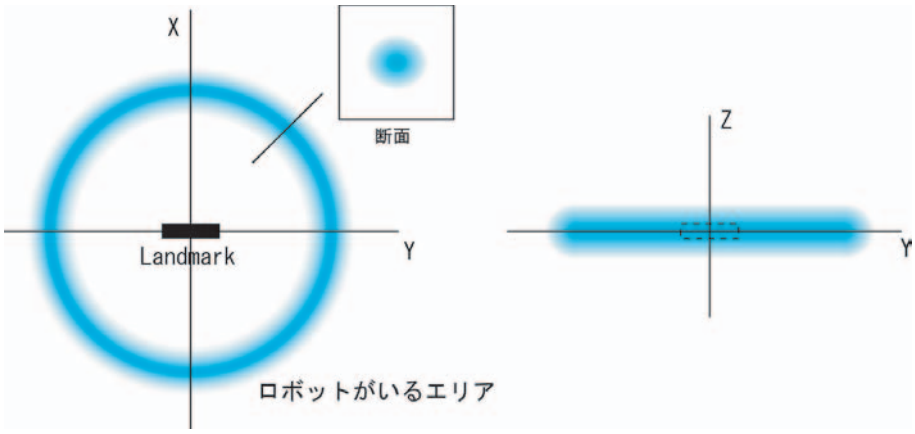

図 6.12: 円環状に分布するサンプル

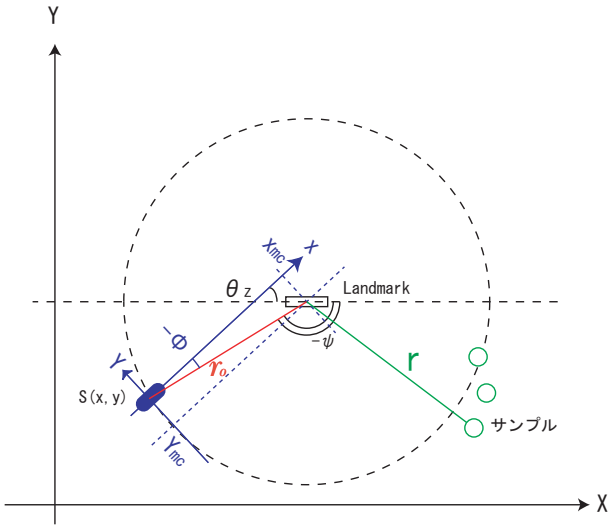

図 6.13: 各サンプルの位置とヨー角

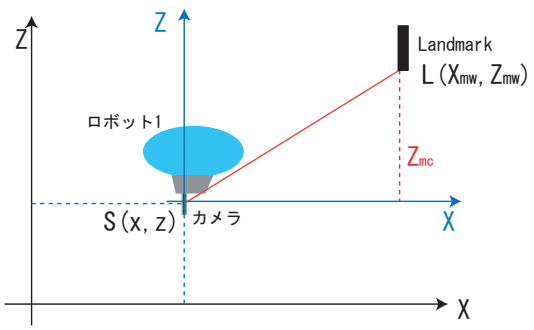

図 6.14: 各サンプルの位置（X-Z 平面）

きくなるからである．そこで本研究では各サンプルの重要度を更新するときに観測データに従って θ_z も更新する．

$$\theta_z = \pi - \psi + \phi \tag{6.23}$$

ここで ψ, ϕ は

$$\phi = \tan^{-1} \frac{y_{mc}}{x_{mc}} \tag{6.24}$$

$$\psi = \tan^{-1} \frac{y^{(i)} - y_{mw}}{x^{(i)} - x_{mw}} \tag{6.25}$$

として与えられる．よって各サンプルは，その位置に応じたヨー角 θ_z が与えられる（図 6.15 参照）．

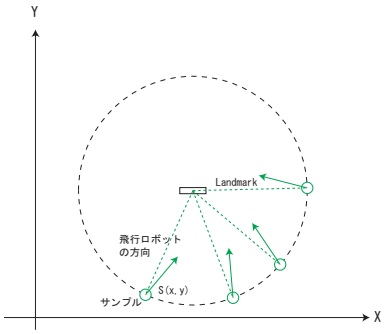

図 6.15: 各サンプルのヨー角（X-Y 平面）

さらに，各サンプルはそのサンプルがどの円環に属するかの情報を保つため，円環の中心座標を保存しておく．つまり最後に観測したワールド座標系での landmark の位置を記憶しておく．

$$C(X_{cw}, Y_{cw}) = (x_{mw}, y_{mw}) \tag{6.26}$$

この情報は後の協調位置推定で用いる．

サンプルセットの移動

landmark を観測したロボットが行動データに基づいて移動する場合，ロボットの位置を表すサンプルセットは，円環の形を保ちながらサンプルセットが属する円環の半径を変化させるように移動する（図 6.16 参照）．

行動データにより飛行ロボットの飛行ロボット座標系での速度ベクトルとワールド座標系を基準としたヨー角の角速度が得られた場合[7]，各サンプルの位置 (x, y, z) は次式で更新される．

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \int R(-\theta_z - V_\theta(t)) \begin{pmatrix} V_{xc}(t) \\ V_{yc}(t) \\ V_{zc}(t) \end{pmatrix} dt \tag{6.27}$$

ここで $R(\theta)$ は z 軸周りの回転を与える回転行列であり，θ_z は飛行ロボット (カメラ）のヨー角，$V_\theta(t)$ は時間 t における飛行ロボットの角速度，$(V_{xc}(t), V_{yc}(t), V_{zc}(t))^T$ はカメラ座標系における飛行ロボットの速度ベクトルを表す．飛行ロボットは推進機構の制約から Y 軸方向への平行移動はほぼ 0 とみなせる．よって $V_{yc} = 0$ である．

[7]行動データに基づき，第 5 章の運動方程式を用いて，その時点で飛行ロボットの速度と角速度を計算している．

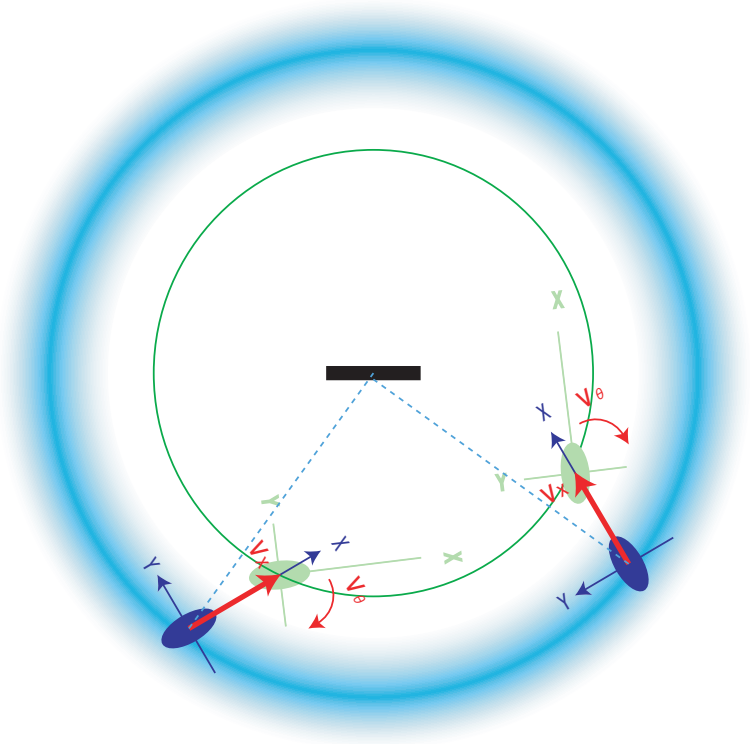

図 6.16: サンプルの移動

2 つの landmark を観測した場合

飛行ロボットの移動する環境に 2 つ以上の landmark が存在する場合，2 つの landmark を観測することでロボットの位置の“ 不確かさ ”を減少させることが出来る．つまり，サンプルの分布を特定の場所に集中させることができる．

図 6.17 を用いて飛行ロボットの位置が収束する原理を述べる．最初に飛行ロボットは landmark1 を観測し，Ladmark1 を中心とする円環状に分布するサンプルセットのみが残る．そして，飛行ロボットが移動することで，この円環状の半径が変化する．飛行ロボットが landmark1 を見失った後もこの円環状を保ちながらサンプルは移動する．landmark1 を見失った後にある時点で landmark2 を観測したとする．このとき（landmark2 の）観測前の事前確率分布は図 6.17 で示される緑の円環で表される．landmark2 の観測データから青の線で示される円が生成される．ここで観測データに基づき各サンプルの重要度を更新する．この円を円環の中心線とする円環内に存在するサンプルは重要度を高くし，その他のサンプルについては重要度を低くする．そして重要度に基づいたリサンプルを行うことで，図 6.17 に示されるように緑の円（円環）と青の円の交点にサンプルが集中する．観測前の各サンプルのヨー角と観測後に計算されるヨー角を比較することで，さらに 1 点にしぼることができる（図 6.17 では下部にある点が正しい位置であり，ヨー角（カメラ座標系の X 軸とワールド座標系の X 軸とのなす角）も一致していることがわかる）．しかし，先にも述べたようにヨー角を用いてサンプルの重要度を更新すると，制約が強すぎるためほとんどのサンプルの重要度が 0 になってしまう．そこで，本研究では landmark の配置の仕方に工夫を凝らし，環境の制約を与えることで円の 2 つの交点のうち誤った方を除去する．具体的には landmark を壁沿いに配置する．このよ

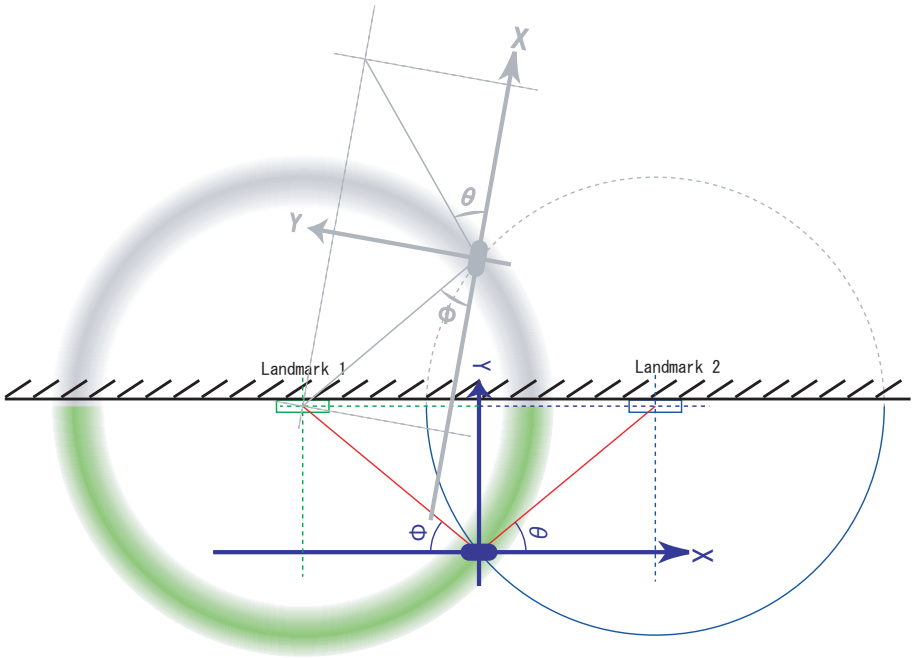

図 6.17: 2 つのランドマークを観測した場合

うにすると，図 6.17 からも分かるように誤った点（図 6.17 参照）はロボットの移動環境の外側になるので，環境の境界条件から誤りだということが分かる．また，図 6.18(a) のように landmark1 を観測した後に反対側の landmark2 を連続して観測する場合は，2 点の位置が残ってしまうが，環境に設置される landmark の数が十分にあれば図 6.18 の (a) よりも (b) のように同じ壁側に設置された landmark を続けて観測されることが多くなると考えられる．また，壁に landmark を設置することは実用上から考えても簡単に設置することができ，環境に与える変化も少なくて済む．

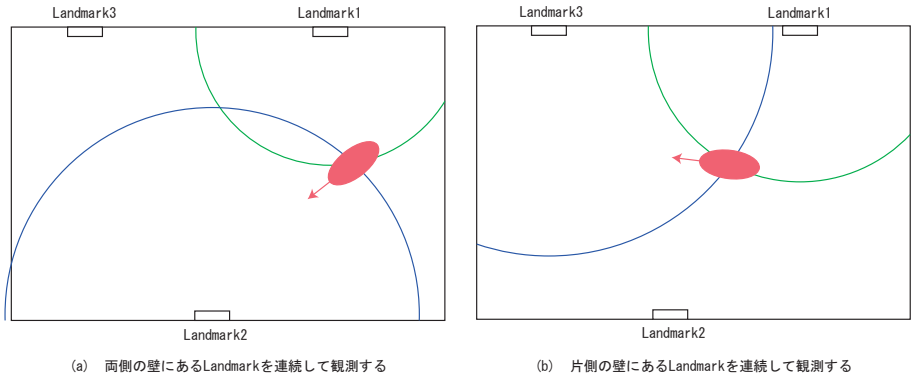

図 6.18: 連続して 2 つの landmark を観測する場合

6.6 協調位置推定

第 3 章で述べたように，本研究では複数台の飛行ロボットがお互いの情報を共有することで，それぞれの位置推定の誤りを修正すると共に，飛行ロボットの位置の曖昧さを減少させ，協調位置推定を行う．それぞれの飛行ロボットの情報を共有することで，広がった存在確率分布を鋭い分布へと変化させる．協調位置推定では，それぞれの位置の確率分布を共有するだけでなく，他の飛行ロボットを観測したときの相対位置関係を利用する．相対位置関係を確率分布として与え，それぞれの飛行ロボットの存在確率分布と照らし合わせることで，飛行ロボットの位置を推定する．具体的な処理の流れは以下の通りである．ここでは，2 台の飛行ロボットを用いたについて述べる．2 台の飛行ロボットの場合の協調位置推定の手順は，以下の通りである．

1. 飛行ロボット 1 の自己位置推定
2. 飛行ロボット 2 の自己位置推定
3. 飛行ロボット 1 の観測による飛行ロボット 2 の位置推定
4. 飛行ロボット 2 の観測による飛行ロボット 1 の位置推定

※他の飛行ロボットが観測されない場合はスキップする．

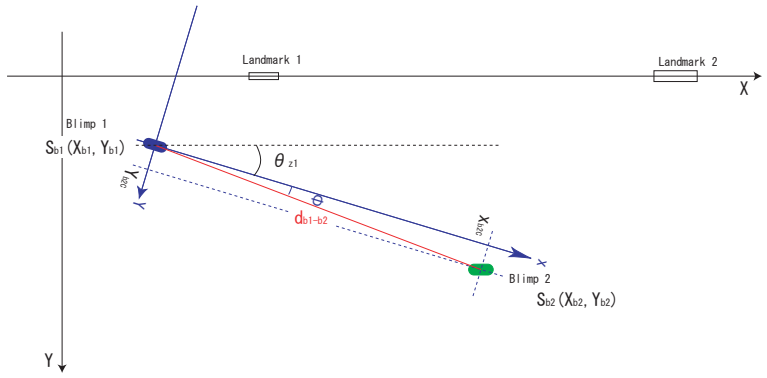

図 6.19: 他のロボットを観測した場合（X-Y 平面）

飛行ロボット 1 が飛行ロボット 2 を観測した場合（図 6.19 参照），飛行ロボット 1 と飛行ロボット 2 の相対距離 d_{b1-b2} は次式で求められる．

$$d_{b1-b2} = \sqrt{x_{b12c}^2, y_{b12c}^2} \tag{6.28}$$

ここで，(x_{b1c}, y_{b1c}) は飛行ロボット 1 のカメラ座標系における飛行ロボット 2 の座標を表す．また，このとき飛行ロボット 1 のカメラ方向ベクトルに対して，飛行ロボット 2 が存在する方向ベクトルとのなす角 ϕ は

$$\phi = \tan^{-1} \frac{y_{b12c}}{x_{b12c}} \tag{6.29}$$

である（飛行ロボット 1 のカメラ方向ベクトルを基準とする）．

以上の飛行ロボット 1 における飛行ロボット 2 の観測情報と，観測した時点での飛行ロボット 1 の状態（位置 (x_{b1}, y_{b1}, z_{b1}) とヨー角 θ_{z1}）を用いて，ワールド座標系における飛行ロボット 2 の位置を記述すると

$$x_{b2} = d_{b1-b2}\cos(\phi + \theta_{z1}) + x_{b1} \tag{6.30}$$

$$y_{b2} = d_{b1-b2}\sin(\phi + \theta_{z1}) + y_{b1} \tag{6.31}$$

$$z_{b2} = z_{b12c} + z_{b1} - h_2 \tag{6.32}$$

となる．ここで注意したいのは，z_{b2} に関しては図 6.20 を見て分かるように，飛行ロボット 1 の座標 z_{b1} に観測データ z_{b12c} を加算したものにはならない．なぜなら，飛行ロボットの構造上マーカとカメラの位置に差（図 6.20 の h_2）ができてしまうため，これを考慮するからである．

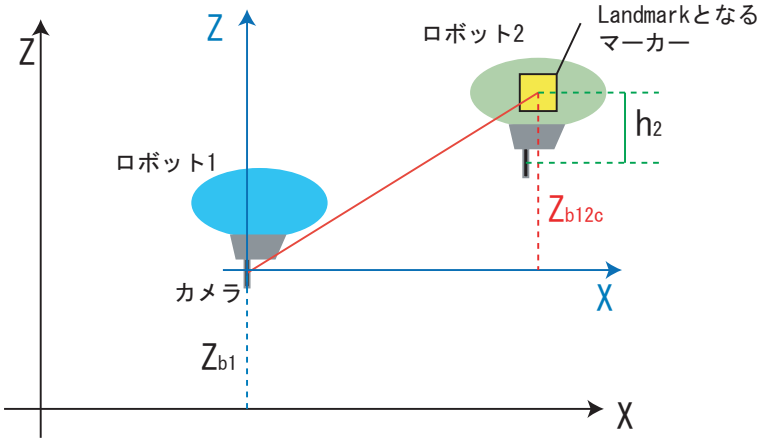

図 6.20: 他のロボットを観測した場合（X-Z 平面）

※マーカとカメラが同一座標にならないので，観測データをその差の分だけ補正する必要がある．

このようにして，飛行ロボット 1 の観測データから飛行ロボット 2 の座標が求まる．前節までに述べたように，各飛行ロボットの位置は確率分布によって表現されているので，(飛行ロボット 1 による飛行ロボット 2 の）観測データに基づいた飛行ロボット 2 の位置も確率分布になる．この観測に基

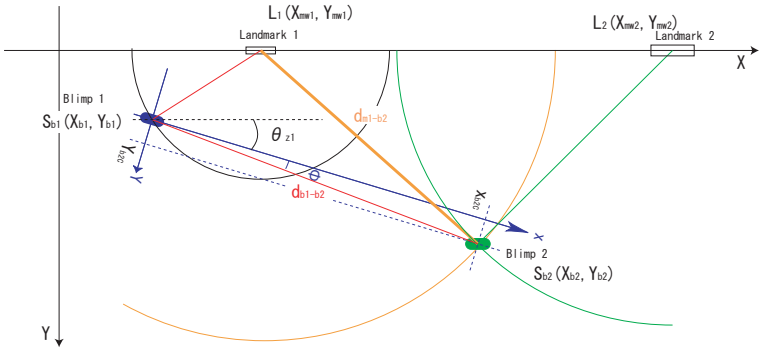

図 6.21: 他のロボットを観測した場合（X-Y 平面）

づく他の飛行ロボットの存在確率分布を，次の例を用いて求める．図 6.21 のように飛行ロボット 1 が landmark1 と飛行ロボットを同時に観測したとする．この場合,（飛行ロボット 1 の観測データから得られる）飛行ロボット 2 の存在確率分布は次のように求めることができる．

先ず landmark1 の観測結果から飛行ロボット 1 の自己位置推定を行い，飛行ロボット 1 の状態を表すサンプルセットをつくる．この中から任意のサンプル i を選び，サンプル i に基づく飛行ロボット 2 の位置 $(x_{b2}^{(i)}, y_{b2}^{(i)}, z_{b2}^{(i)})$ を求める．次に飛行ロボット 2 と landmark1 の距離を次式で計算する．

$$d_{b2-m1} = \sqrt{(x_{b2}^{(i)} - x_{mw1})^2 + (y_{b2}^{(i)} - y_{mw1})^2} \tag{6.33}$$

ここで (x_{mw1}, y_{mw1}) は landmark1 のワールド座標である．n は $-\pi < n < \pi$ の任意の実数である．

r_{b2-m1} は他のサンプルについて計算しても，ほぼ同じ値が得られる（landmark1 のサンプルの分布によって誤差が決まる）．つまり飛行ロボット 2 の存在確率分布は，landmark1 を中心に半径 r_{b2-m1} によって作られる円を円環の中心線とした確率分布になる．よって飛行ロボット 2 の位置は次式で表すことができる．

$$x_{b2} = (d_{b2-m1} + r_e)\cos n + x_{mw1} \tag{6.34}$$

$$y_{b2} = (d_{b2-m1} + r_e)\sin n + y_{mw1} \tag{6.35}$$

$$z_{b2} = z_{b12c} + z_{b1} - h_2 + z_e \tag{6.36}$$

ここで r_e, z_e はノイズによる誤差を表す．n は $-\pi \leq n \leq \pi$ の任意の実数である．

この確率分布を用いて飛行ロボット 2 のサンプルセットの重要度を更新する．この更新によって飛行ロボット 2 の位置が定まらない場合は飛行ロボット 2 の自己位置推定に誤りがあるか，飛行ロボット 1 の観測データに誤りがあるかのどちらかである．どちらの場合にしても各飛行ロボットのサンプルセットを一度初期化する．前者の場合であれば，飛行ロボット 2 が新たに観測データを取り入れて自己位置推定をし，再び飛行ロボット 1 からの観測データを作用させればよい．飛行ロボット 2 が観測データを得られない場合は，飛行ロボット 1 の観測データに従って飛行ロボット 2 の存在確率分布をつくる．後者の場合は飛行ロボット 1 のサンプルセットが初期化されることで,（飛行ロボット 1 による）飛行ロボット 2 の存在確率分布が初期化されるので，正しい観測データによって新たに推定を行えばよい．

飛行ロボット 2 が正しい自己位置推定を行えた場合，飛行ロボット 1 が飛行ロボット 2 を観測することで，飛行ロボット 1 自身の存在確率分布も正しい位置へと収束させることができる．なぜなら飛行ロボット 1 は飛行ロボット 2 を介して landmark2 の情報を得ているからである（図 6.22 参照）．

ここで飛行ロボット 2 の位置が定まった場合，飛行ロボット 1 の位置は，飛行ロボット 2 を中心として半径 d_{b1-b2} の円を中心線とする円環状の分布になる．つまり飛行ロボット 1 の位置は

$$x_{b1} = (d_{b1-b2} + r_e)\cos n + x_{b2} \tag{6.37}$$

$$y_{b1} = (d_{b1-b2} + r_e)\sin n + y_{b2} \tag{6.38}$$

$$z_{b1} = z_{b2} + h_2 - z_{b12c} + r_z \tag{6.39}$$

となる．ここで n は $-\pi \leq n \leq \pi$ の実数で，r_e, r_z は誤差を表す．この確率分布により飛行ロボット 1 のサンプルセットの重要度を更新することで，飛行ロボット 1 の位置の不確かさを減少させることができる．

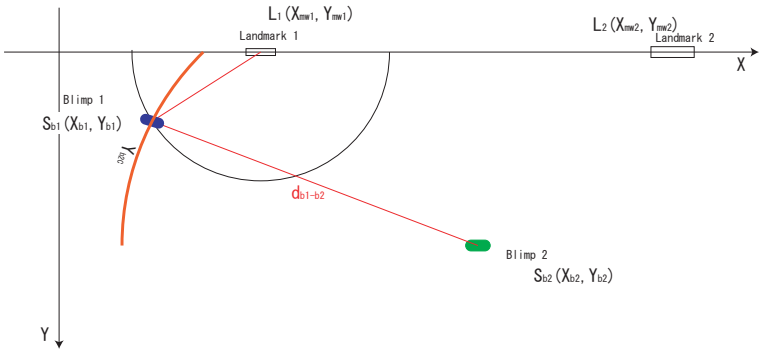

図 6.22: 他のロボットを観測した場合（X-Y 平面）

第7章　評価実験

本章では，提案する群飛行ロボットの協調位置推定法の評価実験について述べる．評価実験は，第3章で述べたシステムを用いて行う．

7.1　実験準備　～環境とソフトウェアの設定～

位置推定システムの評価実験は，幅7.7m 奥行き5.3m 高さ2.5m の部屋に，図7.1 のようにlandmark となるマーカを12 枚を設置した環境で行う．12 枚のマーカは全て異なったパターンを使い，0～11 の ID により識別する（図7.1 参照）．各マーカの設置位置を表7.1 に示す．

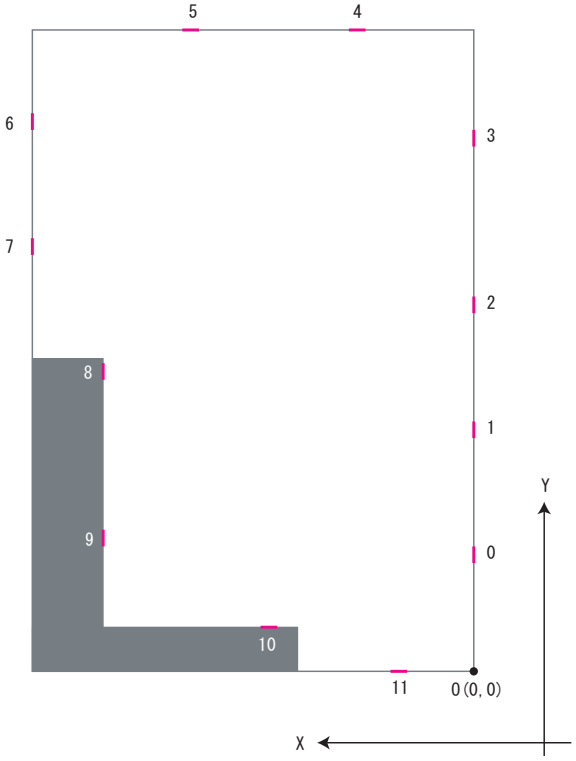

図 7.1: マーカ (landmark) の配置

表 7.1: マーカの座標

マーカ ID	x	y	z
0	0	150	150
1	0	300	150
2	0	450	150
3	0	650	150
4	150	770	150
5	350	770	150
6	530	650	150
7	530	500	150
8	445	350	150
9	445	150	150
10	236	53	150
11	100	0	150

用いた飛行ロボットは2 台で，図3.4 に示すものである．図3.4 を見て分かるように，各飛行ロボットにも環境に設置したマーカと同種のものを，エンベロープ両側面に計2 枚貼り付けてある．飛行ロボット用の2 枚のマーカは，同じパターンを用いる（飛行ロボットを観測した時に，それが左側か右側かは判断しない）．

実験項目は以下の通りである.

- 自己位置推定の評価
  - 飛行ロボットの静止時における位置推定の評価
  - 飛行ロボットの移動時における位置推定の評価
- 協調位置推定の評価

以上の項目全てについて，Global Localization Problem の設定で実験を行う（初期状態は与えず，自己位置がわからない状態からスタートする）.

また，位置推定アルゴリズムで用いるサンプルの個数は初期状態で 10000 個，マーカの観測後では 1000 個とした．マーカ観測後は少ないサンプル数でも飛行ロボットの位置を十分に正しく表現でき，また計算コストを減らすために減少させる設定で行った.

7.2 自己位置推定の評価実験

図 7.1 の環境に 1 台の飛行ロボットを置き，位置推定の精度を測った．自己位置推定に関する評価実験は，飛行ロボットが静止している場合と飛行ロボットが移動している場合の 2 通りで行う.

7.2.1 静止時における位置推定

静止時における飛行ロボットの位置推定能力を観測されるマーカの個数別に測定した．これは観測されるマーカ数が増加することによって，位置の“ 不確かさ ”が減少するかどうかをサンプルの分布を観測することで判断する．マーカとの距離や視角の変化による測定精度は，ARToolKit の精度に大きく依存するため本稿では省略する．行う実験は次の 3 項目である.

- 1 つのマーカを観測した場合のサンプルの分布
- 2 つのマーカを観測した場合のサンプルの分布
- 3 つのマーカを観測した場合のサンプルの分布

飛行ロボットの設置位置と観測するマーカの対応は図 7.2〜7.4 に示す.

7.2.2 移動時における位置推定

移動時における自己位置推定能力を測定する．図 7.5 のように飛行ロボットを移動させ，各サンプルは行動命令に従って，移動する．この時，環境中のマーカを観測することで，サンプルがどのように変化するかを，観測する.

7.2.3 結果

静止時における位置推定

自己推定の結果を，マーカ観測前と観測後のサンプルの分布によって示す．先ず，マーカが観測される前の初期状態を図 7.6 に示す．これは３つの実験とも同様なので，ここではマーカが１つの場合

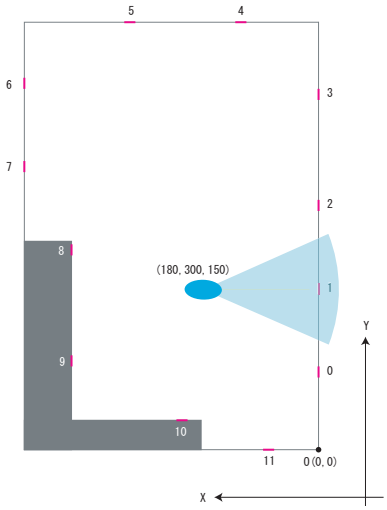

図 7.2: 1 つのマーカのみの観測

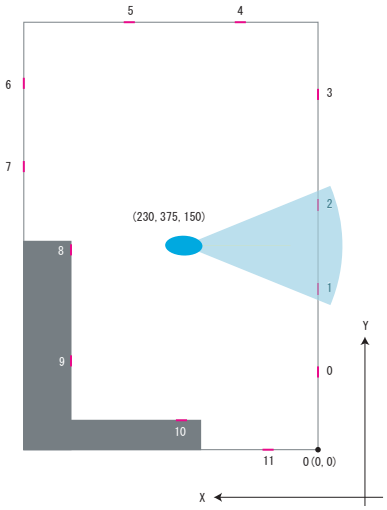

図 7.3: 2 つのマーカの観測

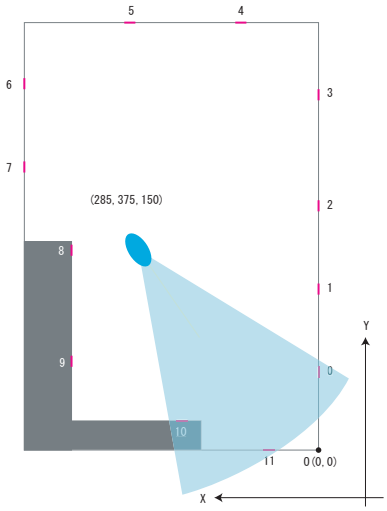

図 7.4: 3 つのマーカのみの観測

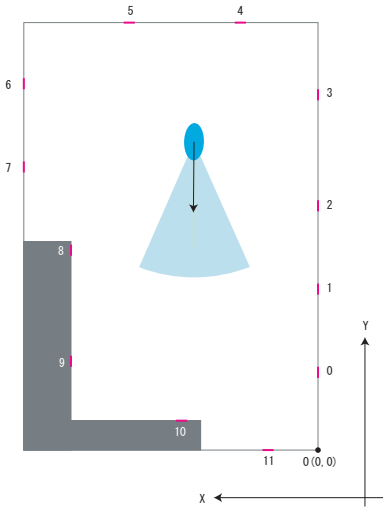

図 7.5: 移動時における自己位置推定

の実験結果のみを示し，残りの２つについては省略する．マーカを観測後のサンプルの状態は，マーカを１つ観測した場合が図 7.7，２つ観測した場合が図 7.8，３つ観測した場合が図 7.9 の結果になった．マーカを２つ観測した場合と３つ観測した場合で，それぞれ飛行ロボットの位置が正しく推定されているのがわかる．

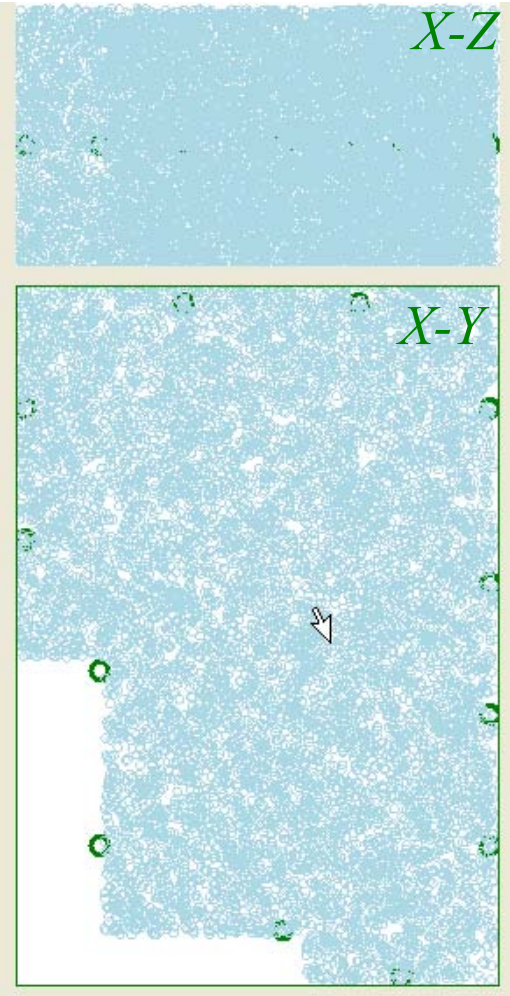

図 7.6: サンプルの初期状態

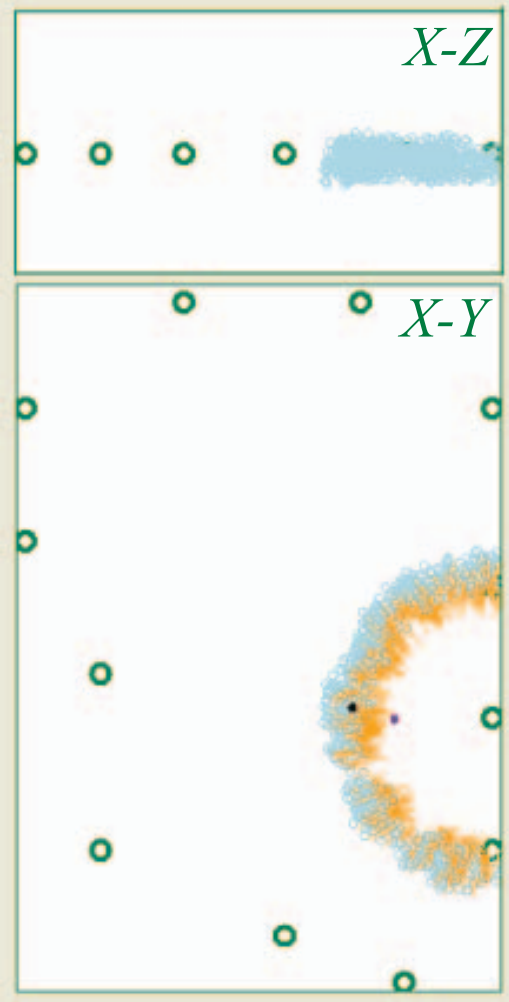

図 7.7: 結果：1 つのマーカ観測後のサンプルの状態

※点（水色）はサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す．

移動時における位置推定

移動時における自己位置推定の結果を，図 7.10〜図 7.13 に示す．移動に伴い，マーカを多数観測することで，サンプルが収束することがわかる．

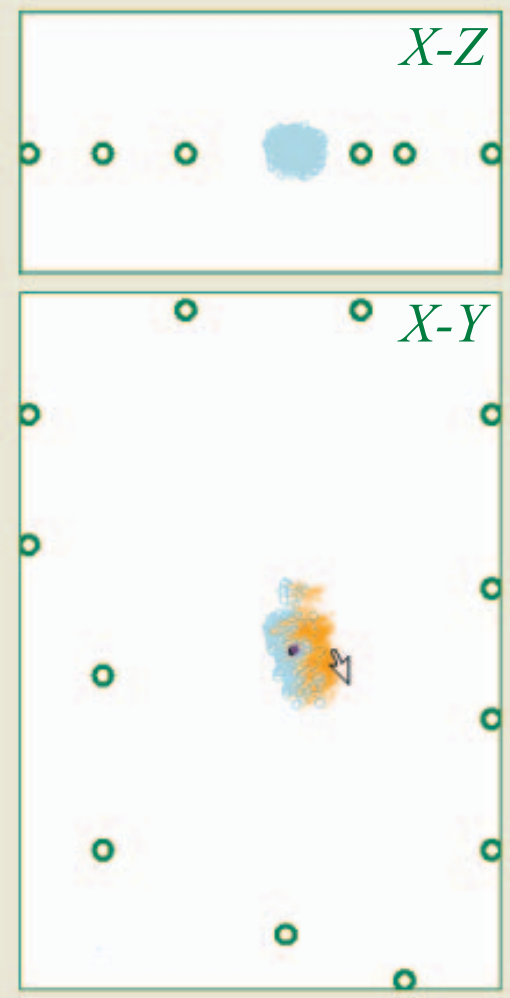

図 7.8: 結果：2 つのマーカ観測後のサンプルの状態

※サンプルの中心の座標

$(x, y, z) = (233, 383, 154)$

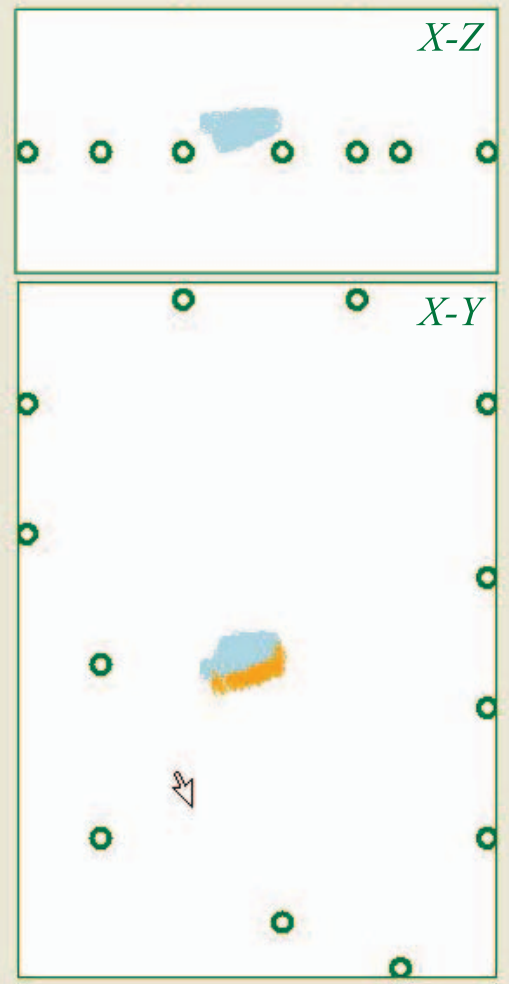

図 7.9: 結果：3 つのマーカ観測後のサンプルの状態

※サンプルの中心の座標

$(x, y, z) = (283, 367, 162)$

※点（水色）はサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す.

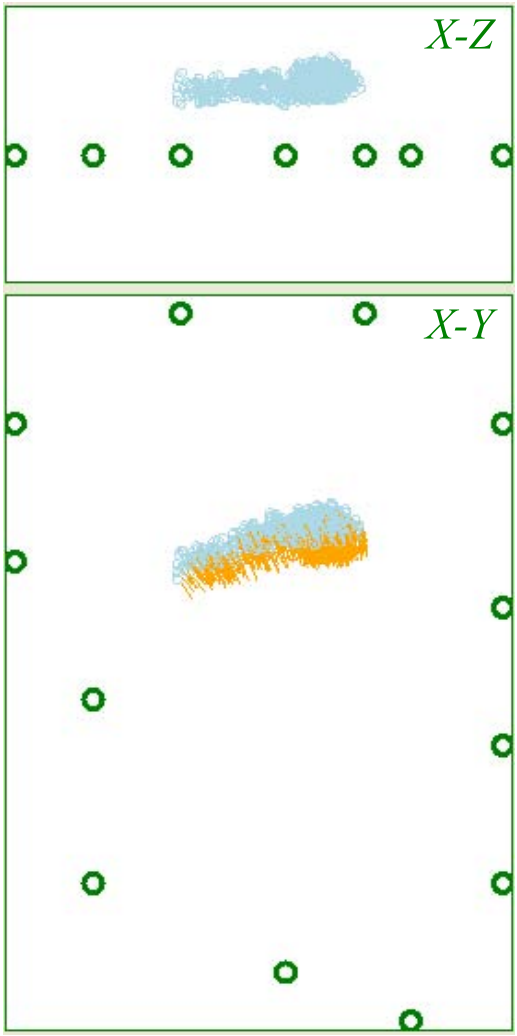

図 7.10: 結果：移動時のサンプルの状態（初期値）

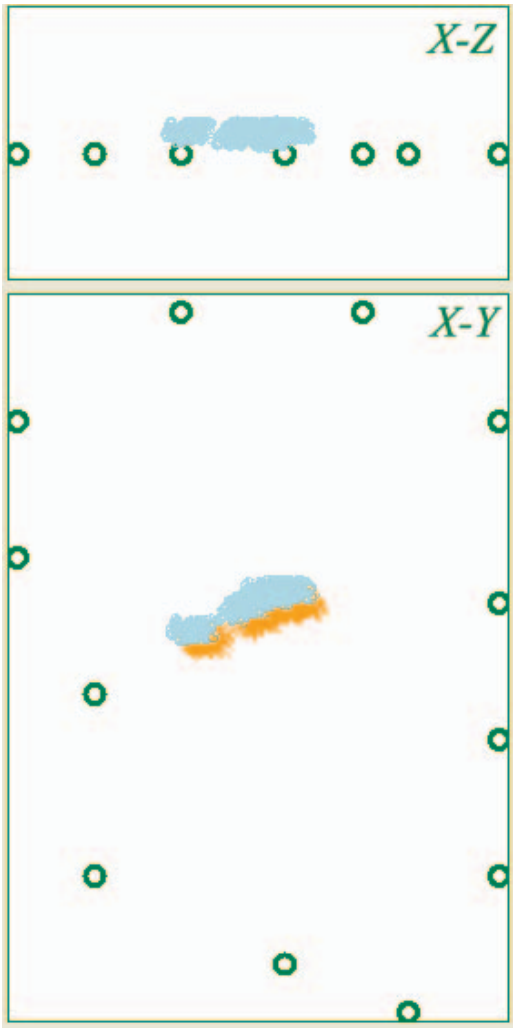

図 7.11: 結果：移動時のサンプルの状態（移動命令 7 回実行後）

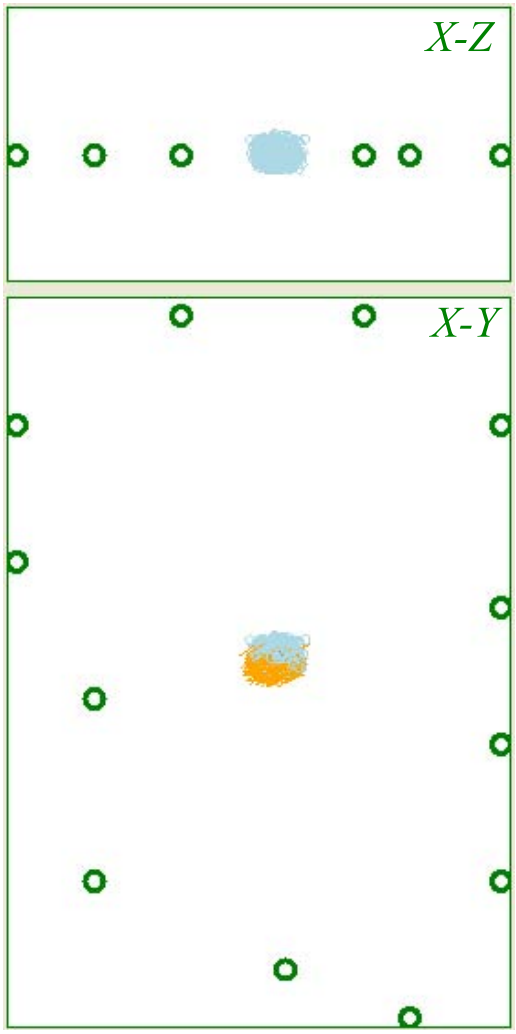

図 7.12: 結果：移動時のサンプルの状態（移動命令 9 回実行後）

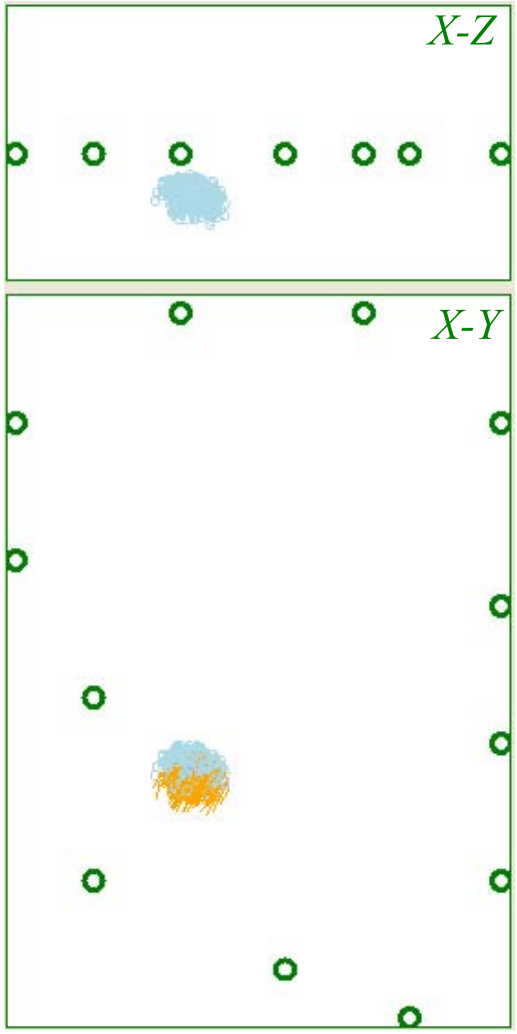

図 7.13: 結果：移動時のサンプルの状態（移動命令 14 回実行後）

※点（水色）はサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す.

7.3 協調位置推定の評価実験

2 台の飛行ロボット（A，B とする）を用いて，A が B を観測することによって，お互いの存在確率がどのように変化するかを，測定する．

7.3.1 2 台とも静止している場合

先ず A，B の両者が静止している場合について測定する．飛行ロボットの配置は，図 7.14 に示す通りである．

それぞれの飛行ロボットの自己位置推定が，正しい位置を見積もれた場合と，間違った位置を見積もってしまった場合とで，どのような変化が見られるかを測定するため，以下の 3 項目の場合にわけて実験を行った．

- A も B も自己位置推定を正しく行った場合（図 7.14 参照）．
- B が間違った自己位置推定をしていた場合（図 7.15 参照）．
- A が間違った自己位置推定をしていた場合（図 7.16 参照）．

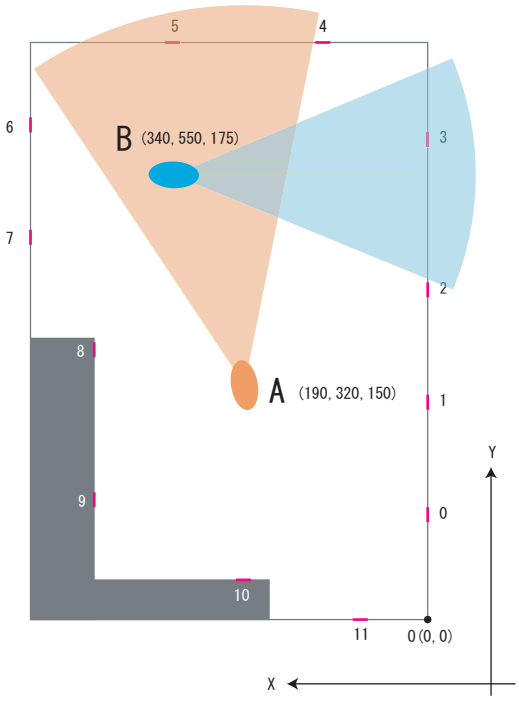

図 7.14: 2 台とも静止している場合

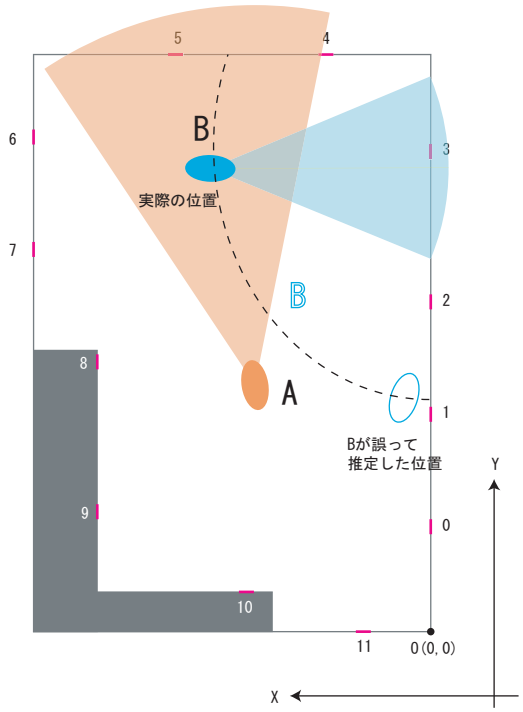

図 7.15: B が間違った自己位置推定をしている場合

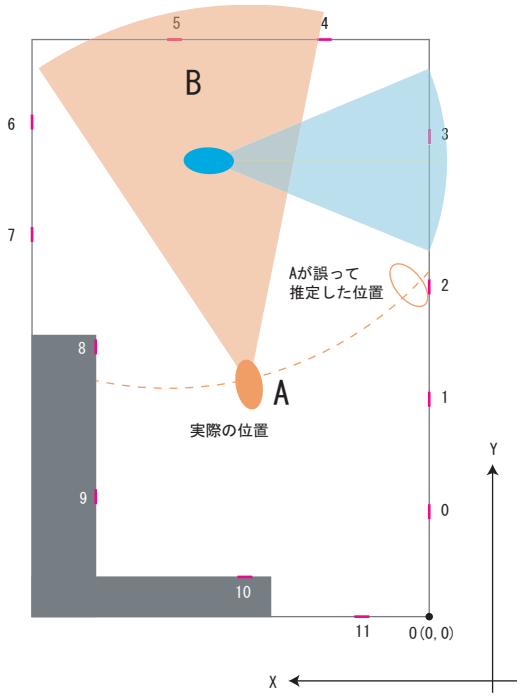

図 7.16: A が間違った自己位置推定をしている場合

7.3.2 2 台が静止 or 移動する場合

静止時の実験と同様に，2 台の飛行ロボットを使う．それぞれの飛行ロボットは，静止または移動をする，この設定において，それぞれの自己位置推定と協調位置推定により，サンプルがどのように変化するかを測定する．設定は以下の通りである．

- A が静止，B が移動（図 7.17 参照）
- A が移動，B が静止（図 7.18 参照）
- A，B 両者とも移動（図 7.19 参照）

7.3.3 結果

A，B 両者が正しく自己位置推定が行われた場合の協調位置推定の結果を，図 7.20〜7.22 に示す．図 7.20 は初期状態なので，環境中にサンプルが散乱している．図 7.21 では，マーカを観測することで，A，B のそれぞれの飛行ロボットの位置は，円環状の存在確率分布に成る．さらに，飛行ロボット A が飛行ロボット B を観測することで，図 7.22 のように正しい位置が求められた．

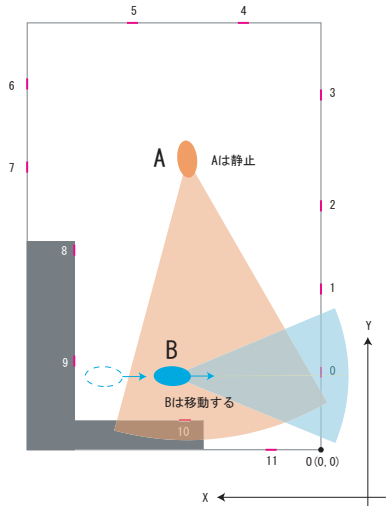

図 7.17: A が静止，B が移動の場合

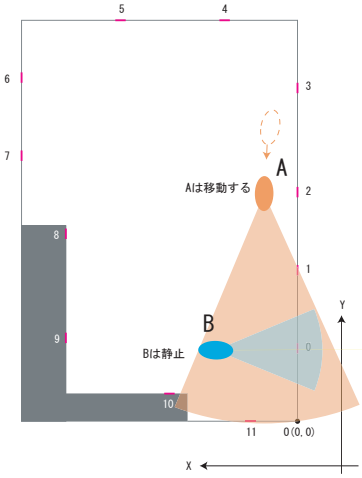

図 7.18: A が移動，B が静止の場合

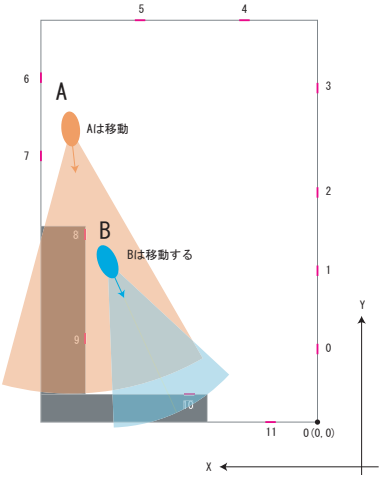

図 7.19: A,B 両者移動の場合

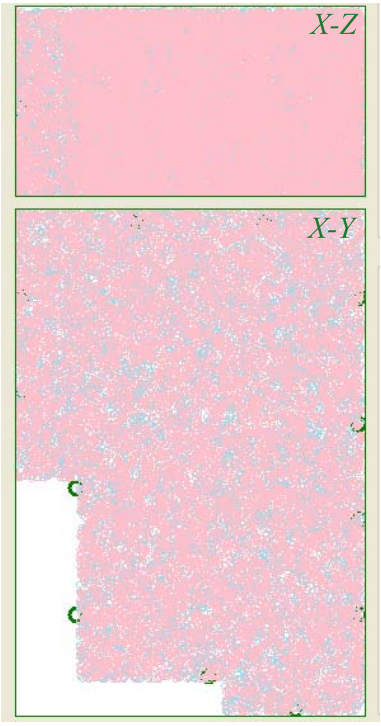

図 7.20: 初期状態

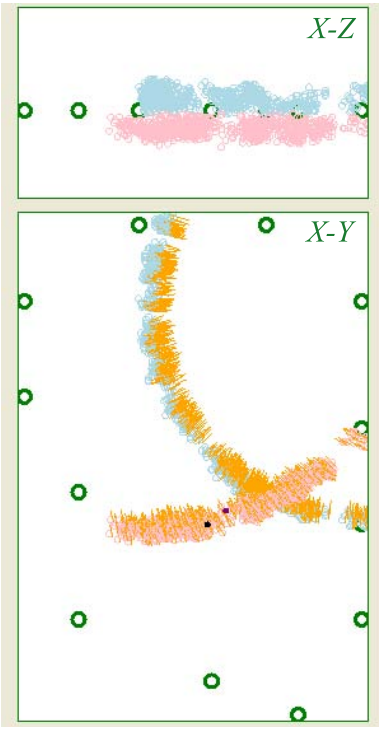

図 7.21: A,B が環境中のマーカを観測した後のサンプルの状態

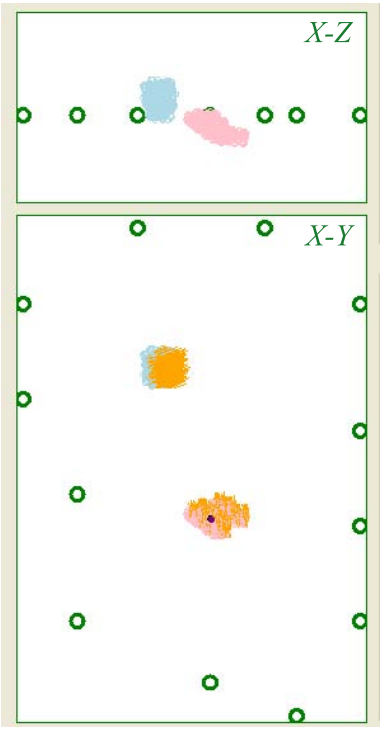

図 7.22: A が B のマーカを観測した後のサンプルの状態

※赤い点は飛行ロボット A のサンプル，青い点は飛行ロボット B のサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す.

B が誤った位置推定をした場合

次に，B の自己位置推定が誤っていた場合についての結果を，図 7.23〜図 7.25 に示す．初期状態の図 7.23 では，B の位置は正しく求められていないが，A の観測情報を得ることで，A，B 両者の位置が正しく求められることがわかる．

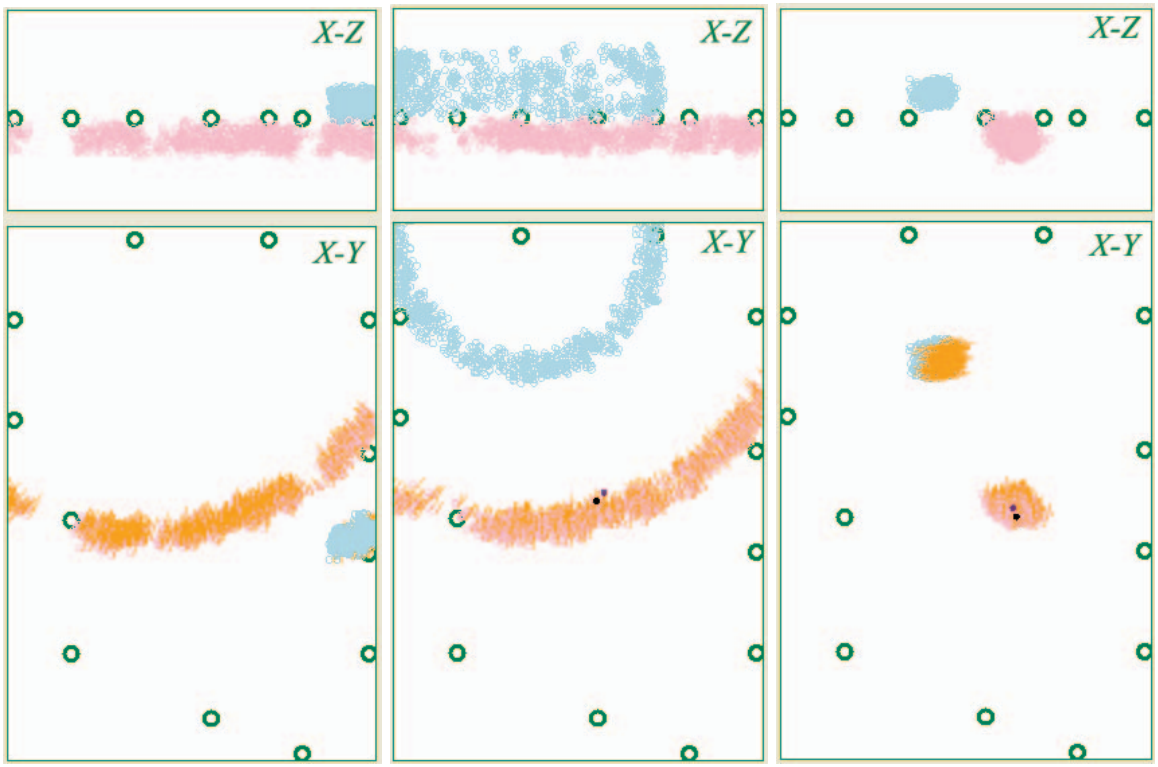

図 7.23: 初期状態 (B が誤った位置を推定)

図 7.24: A の観測により B の存在確率分布を修正（B の観測データがない場合）

図 7.25: A の観測により B の存在確率分布を修正（B の観測データがある場合）

※赤い点は飛行ロボット A のサンプル，青い点は飛行ロボット B のサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す．

A が誤った位置推定をした場合

A の自己位置推定が誤っていた場合についての結果を，図 7.26，図 7.27 に示す．初期状態の図 7.26 では，A の位置は正しく求められていないが，A が B を観測することで A の位置が訂正され，A，B 両者の位置が正しく求められることがわかる．

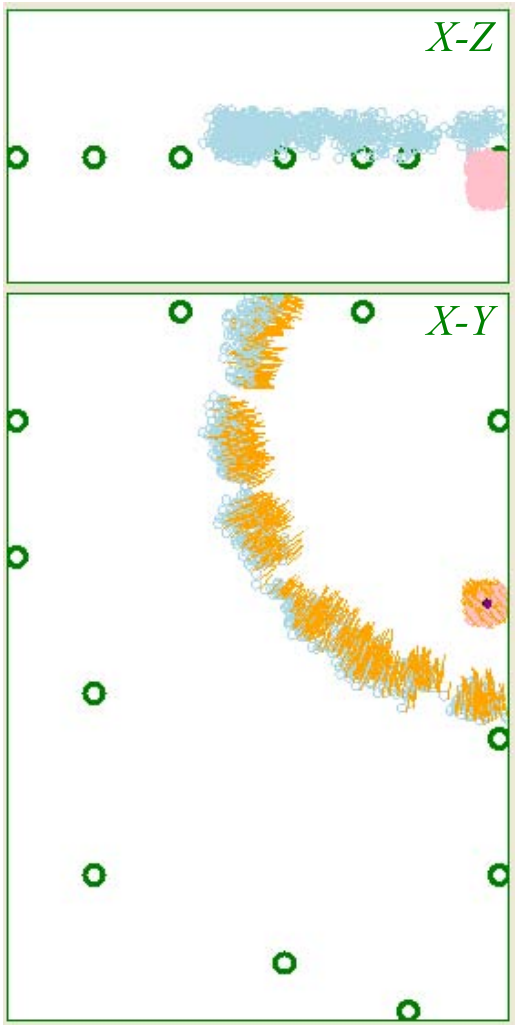

図 7.26: 初期状態 (A が誤った位置を推定)

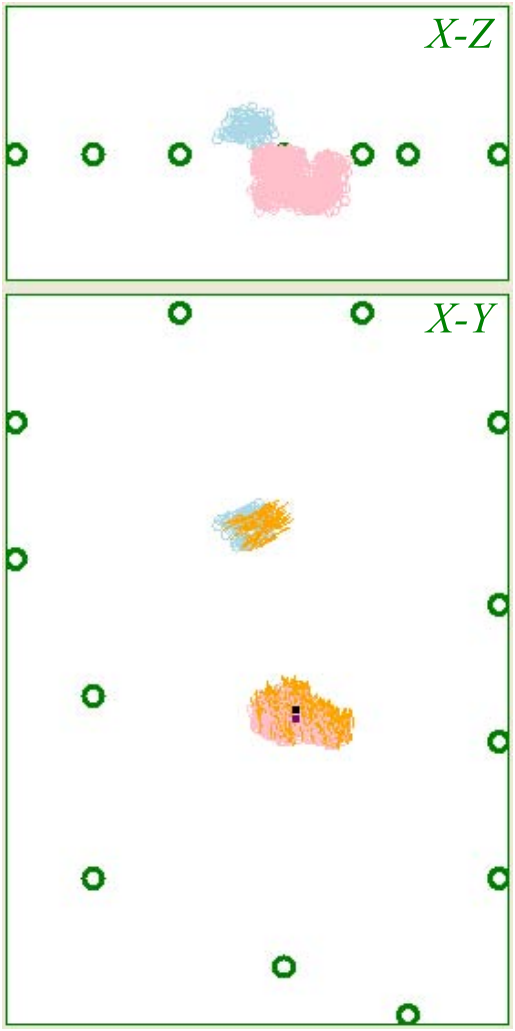

図 7.27: A の観測により B の存在確率分布が修正

※赤い点は飛行ロボット A のサンプル，青い点は飛行ロボット B のサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す.

飛行ロボット A が移動している場合

飛行ロボット A が移動，飛行ロボット B が静止している状況でのサンプルの推移を，図 7.28〜7.30 に示す．図 7.29 では，飛行ロボット A が飛行ロボット B を観測することで，両者のサンプルがすべて破棄され，自己位置推定からやり直しされているのがわかる．これは正しい相対位置情報と，飛行ロボット A の位置情報が，得られなかったと考えられる．また，図 7.30 では，飛行ロボット A の位置が実際の位置と全くちがう位置に推定されてしまった．この時にも，上記と同様の情報が，正しく伝達されなかったためと考えられる．

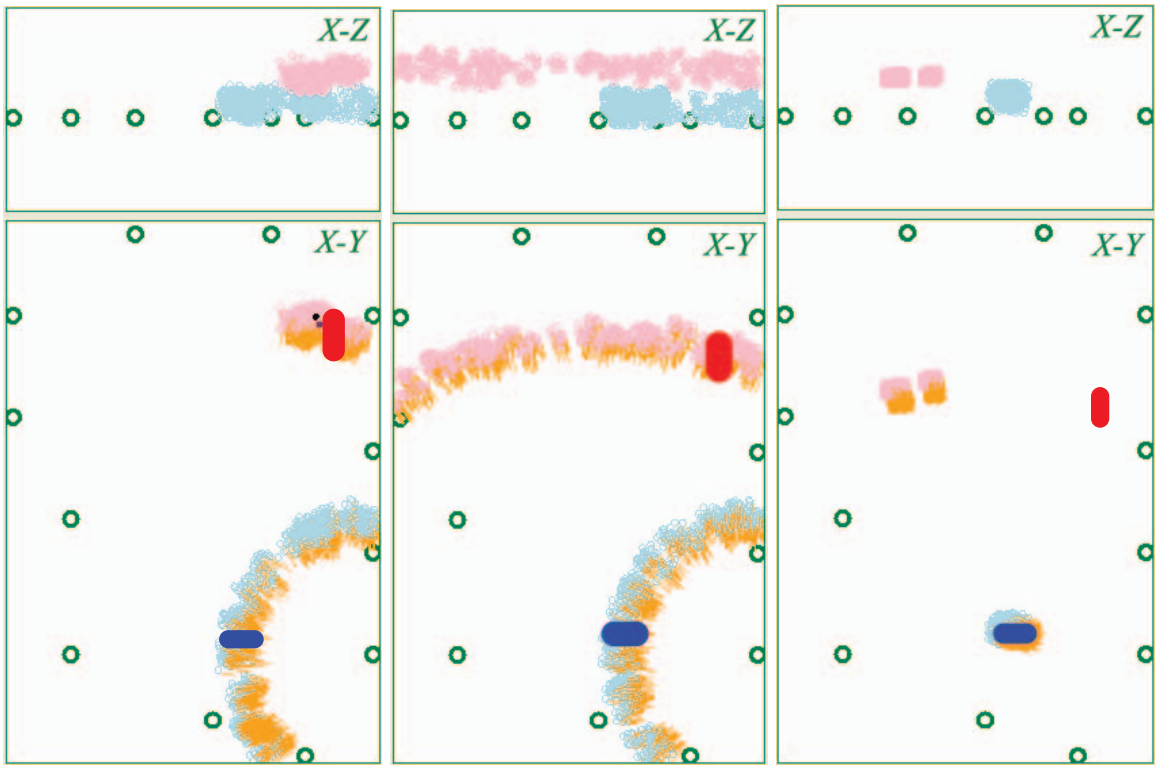

図 7.28: A が移動している場合（初期状態）

図 7.29: A が移動している場合（移動命令 6 回実行後）
※相対位置データに誤りがあり，一度両方のサンプルを初期化して自己位置推定からやり直す．

図 7.30: A が移動している場合（移動命令 9 回実行後）

※赤い点は飛行ロボット A のサンプル，青い点は飛行ロボット B のサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す．

飛行ロボット B が移動している場合

飛行ロボット B が移動，飛行ロボット A が静止している場合の実験結果を，図 7.31〜図 7.33 に示す．図 7.31 と図 7.32 では，自己位置推定のみによる存在確率分布であるが，図 7.33 では，飛行ロ

ボット A が飛行ロボット B を観測して，正しい位置を推定することができた．

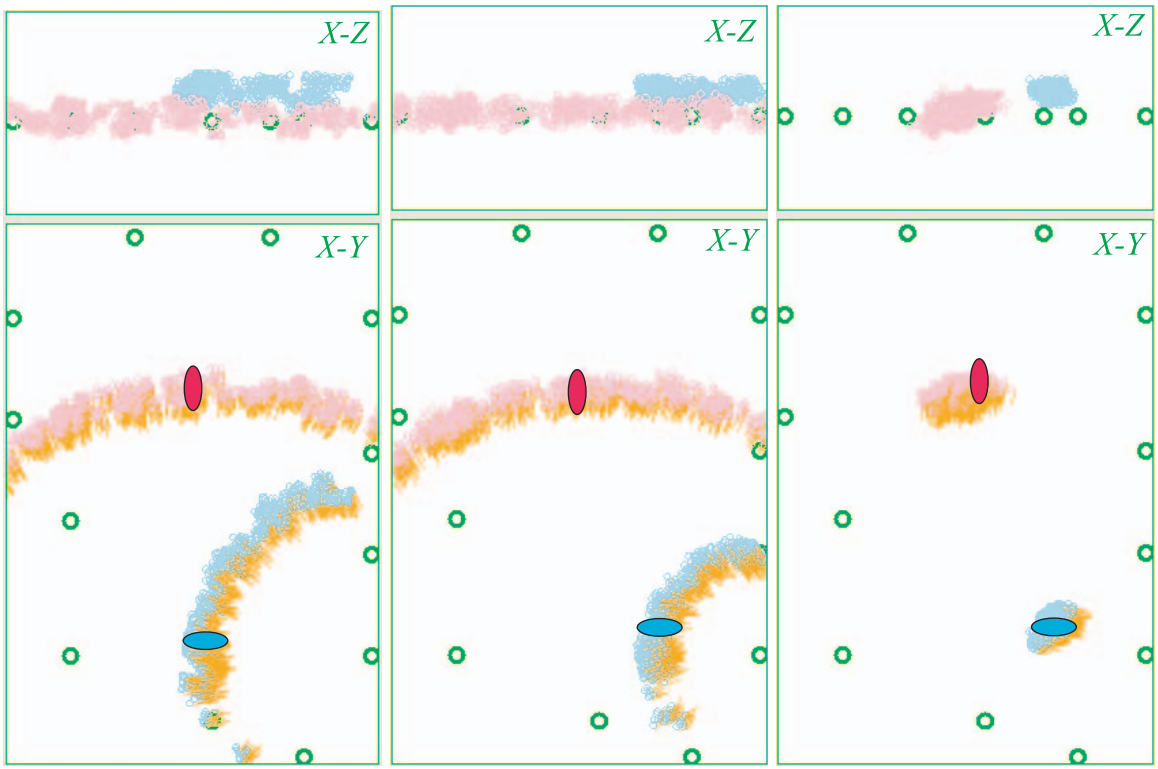

図 7.31: B が移動している場合（初期状態）

図 7.32: B が移動している場合（移動命令 13 回実行後）

図 7.33: B が移動している場合（移動命令 15 回実行後）

※赤い点は飛行ロボット A のサンプル，青い点は飛行ロボット B のサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す．

A,B 両者が移動している場合

飛行ロボット A，B 両者が移動している場合での実験結果を，図 7.34〜図 7.37 に示す．図 7.34 から図 7.35 の間では，協調位置推定によって，互いの位置が正しく求められた．しかし図 7.35 から図 7.36 の間では，協調位置推定によって，逆に飛行ロボット B の位置が見失われてしまった．そして，図 7.37 では，改めて飛行ロボット B が自己位置推定からスタートしている．「飛行ロボット A が移動，B が静止」の実験と同様で，相対位置関係と各飛行ロボットの自己位置の分布に，大きな誤りがあったためと考えられる．

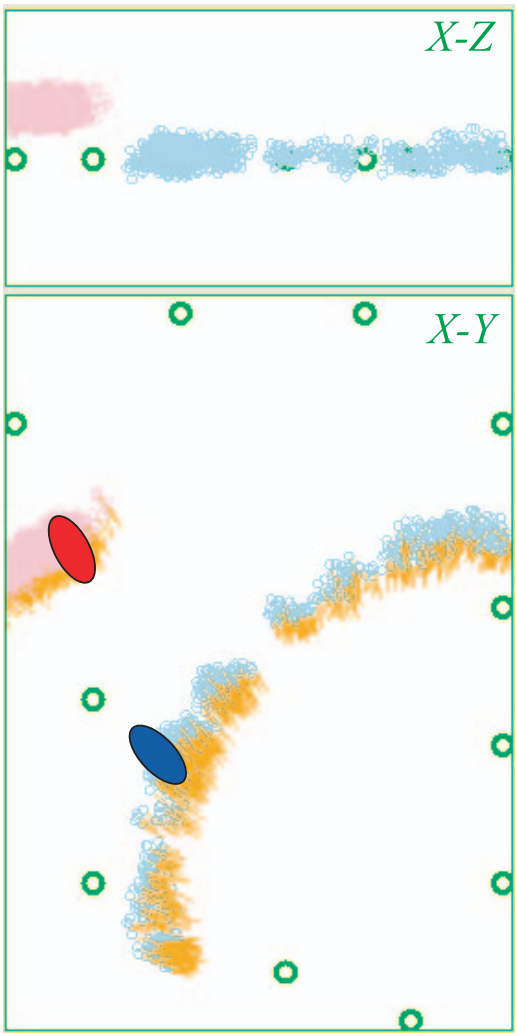

図 7.34: A,B 両者に移動している場合（初期状態）

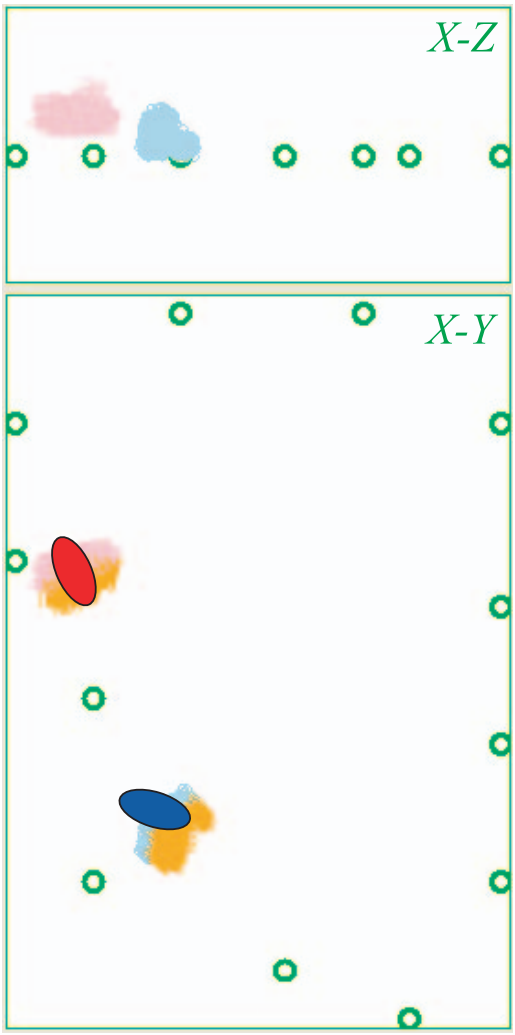

図 7.35: A,B 両者が移動している場合（移動命令 9 回実行後）

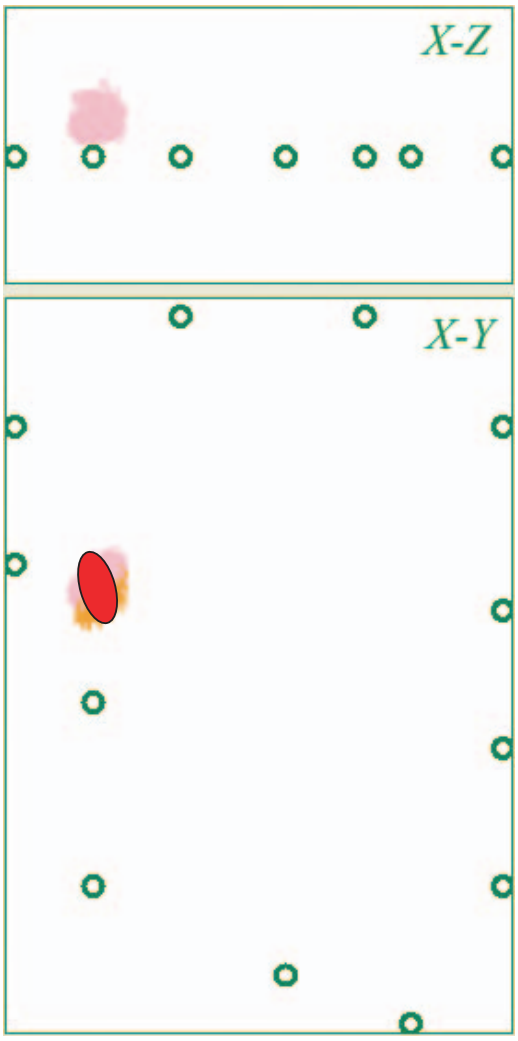

図 7.36: A,B 両者に移動している場合（移動命令 11 回実行後）

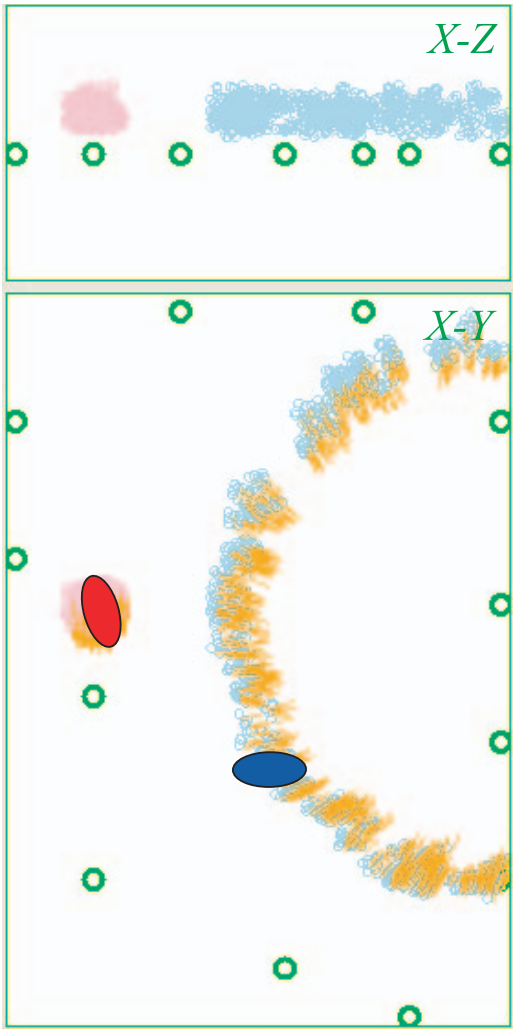

図 7.37: A,B 両者が移動している場合（移動命令 14 回実行後）

※赤い点は飛行ロボット A のサンプル，青い点は飛行ロボット B のサンプルを表し，点から伸びる線はサンプル（飛行ロボット）の方向を表す.

第8章　考察

8.1　自己位置推定に関する考察

自己位置推定では，観測されるマーカが 1 つの場合は，landmark（マーカ）を中心とした円環状にサンプルを分布させることができた．そして，観測される landmark が 2 個，3 個 … と増加すると，円環状の分布はさらに収束し飛行ロボットの位置を正しく求めることができた．この時，サンプルの XY 座標における分布は，飛行ロボットの向きに対して，垂直方向に伸びた分布になっている．これは，図 8.1 を用いて説明できる．飛行ロボットが一度に複数の landmark を観測する場合，飛行ロボットと landmark までの距離は，十分離れている．また，landmark は飛行ロボットの方向と垂直に並び，視野角に収まるように，一定の間隔で，密集している．そのため，それぞれの landmark の観測によってできる“ 観測の円環 ”は，垂直に交差せず，重なりが大きくなるように交差する．これによって，飛行ロボットの分布は，飛行ロボットの向きの垂直方向に広がった分布になるのである．

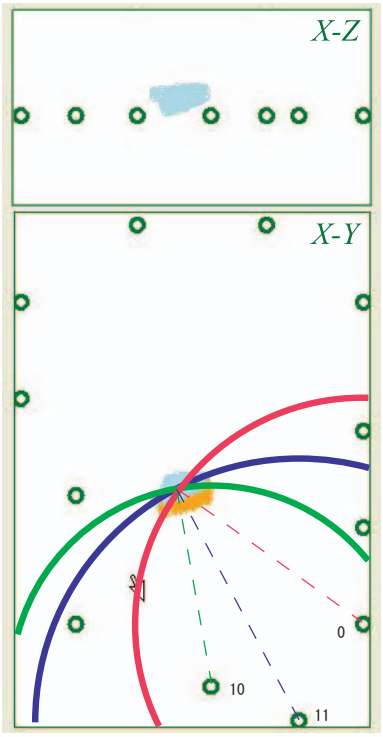

図 8.1: 観測する landmark(マーカ) とサンプルの分布

飛行ロボットが移動していた場合は，移動に合わせてそのサンプルも同時に推移させることで，自己位置推定を行った．landmark の観測が十分に行われれば，その位置推定能力は，静止時において

複数個の landmark を観測した時と同等の結果が得られる．しかし，観測データがないまま一定時間移動を続けると，次の landmark を観測した時に，正しい位置からずれた位置を，自己位置と推定してしまう事がたびたび起きた．この原因は，行動命令の誤差が蓄積されることで，起きてしまう．運動方程式により内部パラメータを用いて，飛行ロボットの移動成分を計算すると，外部環境の変化（特に空調など）によって，飛行ロボットの移動に大きな影響を及ぼしてしまう．これを解決するためには，外部センサを利用して現在の速度や角速度を求めて，外部環境の変化も考慮した移動成分を，推定する必要がある．飛行ロボットの場合は，センサを増やすことは望ましくないため，画像データからオプティカルフローなどによって，速度等を計算する必要がある．

8.2 協調行動に関する考察

協調位置推定では，各飛行ロボットの位置情報と，その相対的な位置関係の情報を統合することで，それぞれの広がった存在確率分布を，正しい位置に収束させることができた．

また各飛行ロボットが誤った位置推定をしてしまった場合に，協調位置推定を行うことで，正しい位置に復帰できることが確認された．評価実験において，飛行ロボット A が自己位置推定を誤った場合についての結果を，解析してみる．飛行ロボット A は，図 8.2 のように，landmark5 を観測して自己位置推定を行っているが，移動に伴う誤差によって，実際の位置から離れた地点を自己位置としてしまった．このとき，飛行ロボット B を観測することで，飛行ロボット A の観測を基にした飛行ロボット B の位置は，図 8.2 の破線で表される位置に，求められる．この飛行ロボット A によって求められた飛行ロボット B の位置は，実際の位置とは外れているが，landmark5 と飛行ロボット B との距離は，実際の位置の場合と変わらないので，飛行ロボット A がつくる飛行ロボット B の存在確率分布は．正しく求められている．よって，この分布を用いて飛行ロボット B の存在確率分布を更新すれば，飛行ロボット B の位置が正しく求められるのである．さらに，飛行ロボット B の位置が定まったことで，飛行ロボット A の位置が誤っていることも分かり，飛行ロボット A の位置を，計算しなおすことが出来る．

同様に，飛行ロボット B が自己位置推定を誤っていた場合，飛行ロボット A が飛行ロボット B を観測することで，飛行ロボット B の位置を正しく求めることができる．図 8.3 のように，飛行ロボット B は landmark3 を観測して自己位置推定をするが，その位置が誤った位置になっている．この時，飛行ロボット A は観測データによって，飛行ロボット B の存在確率分布を作る．飛行ロボット B の存在確率分布は，飛行ロボット A が作った分布とは重ならないため，飛行ロボット B の確率分布は全てゼロになり，初期状態の一様分布に戻る．そして，改めて飛行ロボット B が landmark3 を観測した情報と，飛行ロボット A が飛行ロボット B を観測した情報を用いて，自己位置推定，協調位置推定を行うことで，正しい飛行ロボット B の位置が求まるのである．

移動時における協調位置推定は，実験結果からもわかるように高い頻度で，位置推定に失敗している．この原因の１つは，自己位置推定の場合と同様で，サンプルの遷移に問題がある．また別の原因として考えられるのは，飛行ロボットと同士の相対位置を，正しく扱えていないことも考えられる．特に他の飛行ロボットを観測する飛行ロボット（実験では飛行ロボット A）が移動する場合，相対位置の変化は大きい．自己位置推定と協調位置推定の間に，時間差が生じた場合，各飛行ロボットの自己位置と観測される相対距離に，差が生じているのではないかと考えられる．この問題も，サンプルの遷移に起因する部分（協調位置推定時に，正しい位置にサンプルが遷移していない）があるので，行動の推定の精度を上げる必要がある．

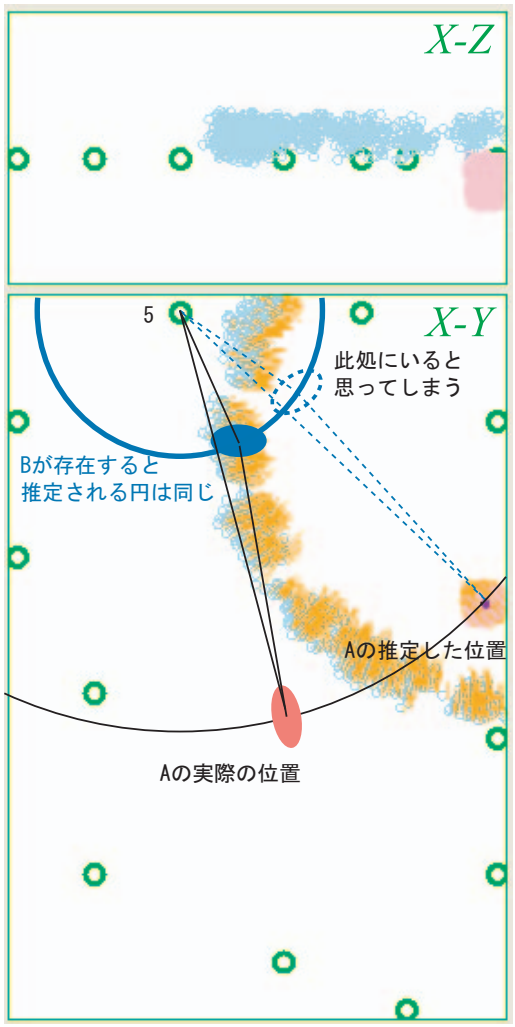

図 8.2: A が誤った位置を推定をした場合に正しく位置推定ができる原理

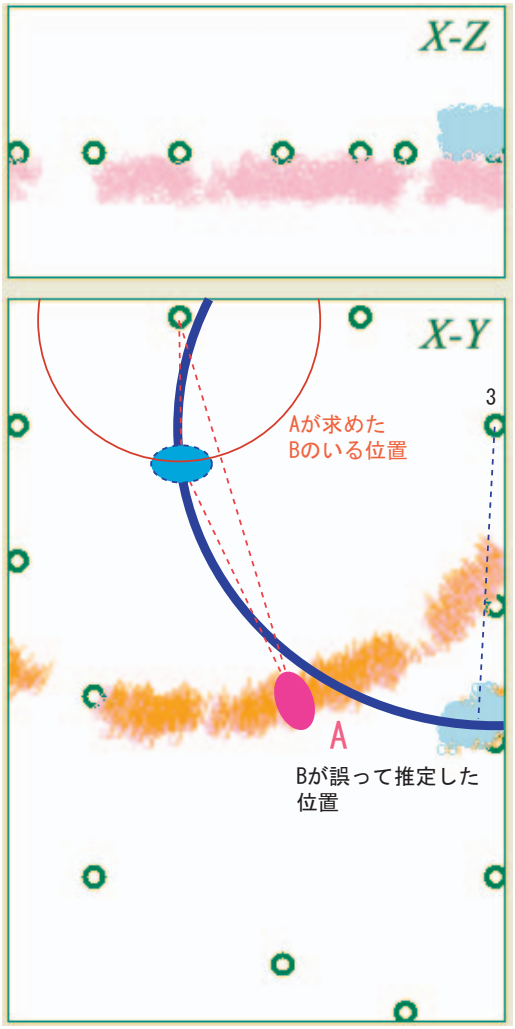

図 8.3: B が誤った位置推定をした場合に正しく位置推定ができる原理

第9章　結論

本研究では，飛行ロボットが地上ロボットにはないさまざまな長所を持っていることに着目し，飛行ロボットの自律移動に必要な位置推定について議論をした．ロボットの位置推定は，ロボットが経路計画や目標物の追従，環境中の物体の衝突回避などをする上で重要な課題のひとつである．本研究では，屋内用飛行ロボットのための位置推定システムを構築するために，群飛行ロボットによる協調位置推定システムを提案した．飛行ロボットは，ペイロードの制約から位置推定に必要な十分なセンサを搭載することができない．この問題を，複数台の飛行ロボットがお互いの情報を共有することで，位置推定の向上を図ることを提案した．そして飛行ロボットには探査や監視といった応用を踏まえて飛行船を採用し，センサにはカメラを用いた．位置推定法には Monte Carlo Localization を採用し，その観測データに円環状の確率分布を用いた．円環状の確率分布を組み合わせることで，確率分布を収束させることができた．この位置推定法による評価実験の結果は，飛行ロボットの静止時において良好な結果を示し，ある飛行ロボットが誤って推定した位置を別の飛行ロボットが観測することで正しい位置に復帰させることができた．しかし，飛行ロボットの移動時においては，移動による誤差によって正しく推定されない場合が度々起きたので，さらに改良する必要があると考えられる．

今後の課題としては，先ず移動における協調位置推定を確実に成功させる必要がある．このためには飛行ロボットの速度や加速度を外部環境の変化から読み取り，移動時のサンプル遷移の誤差を減少させる必要がある．また，本研究では協調位置推定において，他者が自己を観測した時の情報 (Positive Sight) により位置推定の訂正を行った．この他にも，他者が自己を観測しなかった時の情報 (Negative Sight) も有意な情報であるので，これらを利用して位置推定を促進させることが出来ると考えられる．

さらに発展した問題としては，位置推定結果に基づいた自律移動システムを実現させることである．これは移動計画に基づいた位置と実際の位置を比較することで，行動命令に修正を与えることが可能であると考えられる．また場合によっては，位置推定を容易にするような行動命令も与えることで全体としてシステムの向上が図れると考えられる．

謝辞

本研究を進めるにあたって，多くの方のお世話になりました．代表的な方々を挙げさせて頂きます．

研究の遂行並びに本論文を作成するにあたり、懇切丁寧な御指導と御高配を賜わりました杉本雅則助教授に心から感謝を致します．また同じ研究プロジェクトメンバーである新田亮氏，屋比久保史氏には，実験や調査などを協力して頂き，大変感謝しています．研究室の同期である竹内雄一郎氏，三浦宗介氏，宮原耕介，矢谷浩司氏には，日常生活も含めて様々な面でお世話になりました．有難うございます．杉本研究室の後輩達にもお世話になりました．感謝しています．

最後に，家族，友人，名前の挙がっていない方々をも含めた皆様方に心より感謝致します．

参考文献

[1] James Bruce and Manuela Veloso. Real-time randomized path planning for robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[2] Alexander Van de Rostyne. Pixel gallery. http://www.scarlet.be/pixel/.

[3] R.S. Fearing, K.H. Chiang, M. Dickinson, D.L. Pick, M. Sitti, and J. Yan. Wing transmission for a micromechanical flying insect. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1509–1516, 2000.

[4] M Fiala. Artoolkit applied to panoramic vision for robotic navigation. In *Proceedings of the Vision Interface*, pp. 119–127, 2003.

[5] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 2000.

[6] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI/IAAI*, pp. 343–349, 1999.

[7] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, Vol. 11, pp. 391–427, 1999.

[8] Robert Grabowski, Luis Navarro-Serment, Chris Paredis, and Pradeep Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots - Special Issue on Heterogeneous Multirobot Systems*, 1999.

[9] Andrew Howard, Maja J Mataric, and Gaurav S. Sukhatme. Localization for mobile robot teams using maximum likelihood estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 434–459, 2002.

[10] Andrew Howard, Maja J Mataric, and Gaurav S Sukhatme. Putting the 'i' in 'team': An ego-centric approach to cooperative localization. In *Proceedings of the IEEE International Conference on Robotics an Automation (ICRA03)*, pp. 868–892, 2003.

[11] Fumiya Iida and Dimitrios Lambrinos. Navigation in an autonomous flying robot by using a biologically inspired visual odometer. In *Sensor Fusion and Decentralized Control in Robotic System III, Photonics East, Proceeding of SPIE*, pp. 86–97, 2000.

[12] P. Jensfelt and S. Kristensen. Active global localisation for a mobile robot using multiple hypothesis tracking, 1999.

[13] H. Kato and M Billinghurst. Artoolkit. http://www.hitl.washington.edu/artoolkit/.

[14] 加藤博一, Mark Billinghurst, 浅野浩一, 橘啓八郎. マーカ追跡に基づく拡張現実感システムとそのキャリブレーション. 日本バーチャルリアリティ学会論文集, Vol. 4, No. 4, 1999.

[15] Jongwoo Kim and James P. Ostrowski. Motion planning of aerial robot using rapidly-exploring random trees with dynamic constraints. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2200–2205, 2003.

[16] Ilan Kroo. The mesicopter: A meso-scale flight vehicle. http://aero.stanford.edu/mesicopter/.

[17] J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 692–698, 2001.

[18] R. Kurazume and S. Hirose. An experimental study of a cooperative positioning system. *Autonomous Robots*, Vol. 8, No. 1, pp. 43–52, 2000.

[19] R. Kurazume, S. Hirose, S. Nagata, and N. Sashida. Study on cooperative positioning system -basic principle and measurement experiment-. In *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1421–1426, 1996.

[20] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 473–479, 1999.

[21] James M. McMichael and Michael S. Francis. Micro air vehicles - toward a new dimension in flight. http://www.darpa.mil/tto/mav/mav_auvsi.html.

[22] J.D. Nicoud and J.C. Zufferey. Toward indoor flying robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 787–792, 2002.

[23] 西村絢子, 川村秀憲, 山本雅人, 大内東. 追跡タスクにおける自律飛行船ロボットの学習制御. エンターテイメント コンピューティング 2003 IPSJ Symposium Series Vol.2003, No.1, 情報処理学会, 2003.

[24] S. Nagata R. Kurazume and S. Hirose. Cooperative positioning with multiple robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1250–1257.

[25] S.I. Roumeliotis and G.A. Bekey. Distributed multi-robot localization. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, pp. 781–795, 2002.

[26] S. Saripalli, J.F. Montgomery, and G.S. Sukhatme. Visually-guided landing of an autonomous aerial vehicle. pp. 371–380, 2002.

[27] Roumeliotis S.I. and Bekey G.A. Collective localization: A distributed kalman filter approach to localization of groups of mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2958–2965, 2000.

[28] 株式会社タカラ. ドリームフォース 02 スカイシップ. http://www.takaratoys.co.jp/skyship/item.html.

[29] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, Vol. 128, No. 1-2, pp. 99–141, 2001.

[30] Jason Welsby and Chris Melhuish. Autonomous minimalist following in three dimensions:a study with small-scale dirigibles. In *Towards Intelligent Mobile Robots*, 2001.

[31] 柳沢紀子, 川村秀憲, 山本雅人, 大内東. 屋内用バルーン型ロボットの設計とモデル化. エンターテイメント コンピューティング 2003 IPSJ Symposium Series Vol.2003, No.1, 情報処理学会, 2003.

[32] Hong Zhang and James P. Ostrowski. Visual servoing with dynamics: control of an unmanned blimp. *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 618–623, 1999.

[33] Jean-Christophe Zufferey, Dario Floreano, Matthijs van Leeuwen, and Tancredi Merenda. Evolving vision-based flying robots. In *Proceedings of Biologically Motivated Computer Vision Second International Workshop, BMCV 2002*, pp. 592–600, 2002.

[34] Sjoerd van der Zwaan. Vision based station keeping and docking for floating robots, MSc Thesis, Lisbon, May 2001. http://www.isr.ist.utl.pt/labs/vislab/thesis/.

[35] Sjoerd van der Zwaan, Alexandre Bernardino, and Josè Santos-Victor. Vision based station keeping and docking for an aerial blimp. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems-IROS'2000*, pp. 614–619, 2000.