飛行ロボット搭載用一軸距離センサの開発と
その応用に関する研究

伊藤　恒平

システム情報工学研究科

筑波大学

２０１５年　３月

目次

第 1 章　緒言 7

1.1　小型飛行ロボット 7

1.1.1　近年の飛行ロボットの活躍 7

1.1.2　近年の飛行ロボットの研究動向 10

1.1.3　飛行ロボットの小型化 11

1.2　レーザ距離センサの軽量化 13

1.2.1　軽量距離センサの調査 13

1.2.2　軽量一軸距離センサ実現法の検討 15

1.2.3　二次元距離センサの一軸化に伴う熱による誤差の低減 15

1.3　一軸距離センサによる自己位置推定 16

1.4　本論文の構成 17

第 2 章　レーザ距離センサの小型軽量化 19

2.1　まえがき 19

2.2　レーザ距離センサの測定原理 19

2.3　二次元距離センサ 20

2.3.1　回転ミラーによるスキャン機構 21

2.3.2　マルチエコー測定機能 22

2.4　軽量レーザ距離センサの実現方法 23

2.5　レーザ距離センサの熱による誤差 24

2.6　一軸レーザ距離センサにおける熱による変動の低減方法 25

2.6.1　一軸距離計における基準板の観測 25

2.6.2　基準板測定を用いた熱による変動の除去 26

2.7　一軸レーザ距離センサの性能確認 26

2.7.1　温度による測定出力の変動 27

2.7.2 測定値の算出 28
2.8 受光強度による補正 30
2.9 センサ回転による二次元測定 33
2.10 回転テーブルによる連続回転下での測定 35
2.11 あとがき 38

第 3 章 一軸距離センサを用いた飛行ロボットの自己位置推定 40
3.1 まえがき 40
3.2 一軸レーザ距離センサによる周辺環境の認識 42
3.2.1 実験装置 42
3.2.2 実験方法 45
3.2.3 実験結果 48
3.2.4 考察 55
3.3 自己位置推定 56
3.4 モンテカルロ自己位置推定 58
3.5 移動モデル 61
3.6 観測モデル 61
3.7 シミュレーション 62
3.7.1 シミュレーションの目的 63
3.7.2 シミュレーションの結果 63
3.8 飛行ロボットによる自己位置推定実験 68
3.8.1 実験装置 69
3.8.2 実験方法 69
3.8.3 実験結果 69
3.8.4 考察 73
3.9 あとがき 74

第 4 章 結言 75
4.1 本研究のまとめと成果 75
4.2 今後の課題 76

謝辞 78

参考文献 80

付録 A　プロペラデータ　82

付録 B　飛行ロボットの慣性演算　84

付録 C　飛行実験動画サムネイル画像　88

付録 D　モンテカルロ位置推定シミュレーションソースコード　91

付録 E　モンテカルロ位置推定飛行実験データ処理ソースコード　103

図目次

1.1 クアッドロータのプロペラ配置 12
1.2 小型飛行ロボットの自己位置推定及び地図作成 14
1.3 プラットフォームの回転による測域 17

2.1 三角測量方式 20
2.2 位相差検出方式 20
2.3 パルス伝播方式 21
2.4 二次元距離センサ内部模式図 21
2.5 複数物体からのマルチエコーの検出 22
2.6 発光受光シーケンスのタイミングチャート 24
2.7 基準板を観測する方法案 25
2.8 基準板の配置 27
2.9 基準板を設置した様子 28
2.10 温度特性実験装置 29
2.11 2m の目標測定出力と温度の時間推移 29
2.12 基準板と目標の測定出力の差分の時間推移 30
2.13 実験の方法 30
2.14 2000mm の測定目標の測定結果を平滑化した結果 31
2.15 実際の距離と測定結果 31
2.16 受光強度と測定誤差の関係 33
2.17 センサ回転による擬似二次元測定 34
2.18 測定の状況 34
2.19 センサ回転測定結果 35
2.20 ターンテーブルによる回転装置 36
2.21 計測環境 36

2.22 スキャン式センサとの比較 38

3.1 一軸距離センサによる自己位置推定 41
3.2 クアッドロータヘリコプタ 43
3.3 クアッドロータヘリコプタシステム 43
3.4 飛行制御用マイコン 44
3.5 誘導制御用ボードコンピュータ 44
3.6 慣性計測装置 45
3.7 飛行実験環境 45
3.8 クアッドロータヘリコプタの制御 46
3.9 地図と計測データの重ね合わせ（1） 49
3.10 地図と計測データの重ね合わせ（2） 50
3.11 地図と計測データの重ね合わせ（3） 51
3.12 地図と計測データの重ね合わせ（4） 52
3.13 地図と計測データの重ね合わせ（5） 53
3.14 地図と計測データの重ね合わせ（6） 54
3.15 飛行ロボットの飛行軌跡 55
3.16 自己位置推定 57
3.17 初期分布 64
3.18 1 回目のリサンプル後 65
3.19 0.04 秒後のリサンプル後 66
3.20 0.12 秒後のリサンプル後 67
3.21 2.5 秒後のリサンプル後 68
3.22 飛行試験の様子 70
3.23 初期状態 71
3.24 0.02 秒後のリサンプル後 72
3.25 2.5 秒後のリサンプル後 73

B.1 Coordinate system 85

表目次

1.1 小型飛行ロボットの部品と重量リスト 14
1.2 距離センサ 15

2.1 距離と測定値 32
2.2 3DM-GX3 のジャイロの主な仕様 37

A.1 APC 5.5× 4.5 プロペラデータ 82

C.1 飛行実験動画のサムネイル画像 88

第 1 章

緒言

1.1　小型飛行ロボット

1.1.1　近年の飛行ロボットの活躍

近年，阪神淡路大震災や東日本大震災に見られるような大災害や，悪天候による風水被害，火山の噴火などの自然災害が繰り返し発生している．更に自然災害と共に，福島第一原子力発電所の放射能漏れに見られる震災に伴う二次災害など，人々の暮らしを破壊し，尊い人命を奪うような出来事が繰り返されている．

このような際の災害現場では，建物や道路が崩れ，通常の移動手段では辿りつけない場合や，放射能汚染によって人が近づけなくなってしまっている場合がある．そのような場合に，無人ロボットによる，現場の情報取得，被災者の救助や救援活動等に関する情報の収集が期待されているところである．

ロボットの災害現場での活躍については，阪神淡路大震災をきっかけとしたレスキューロボットについての研究が盛んである [1]．レスキューロボットの研究の主な成果としては，クローラ型の地上移動ロボットが成果を上げており福島第一原子力発電所の内部調査などで活躍を見せている [2]．前述のクローラ型も含め地上を移動する無人ロボットは，Unmmand Ground vehicle(UGV) と呼ばれている．その他，災害現場や危険地帯で活躍が期待される無人ロボットには空中，海上，海中を移動する無人ロボットをそれぞれ，Unmmand Aerial vehicle(UAV)，Unmmand Sea vehicle(USV) と呼称されている．

その中で，飛行ロボットについての研究が活況を呈している．飛行ロボットについては米軍による軍事利用が先行しており，我が国においては農薬散布などの民生利用で発展してきた．近年になって，複数のロータをモータで回転することで飛行するマルチコプタと

呼ばれる回転翼機が登場したことも研究を加速していると考えられる．

従来から回転翼方式の飛行ロボットは存在した．従来型の回転翼方式の飛行ロボットには，シングルロータのヘリコプタとタンデムロータのヘリコプタがある．シングルロータのヘリコプタは，大きなメインロータで揚力と推進力を生み出し，尾尻にあるテールロータと呼ばれる小さなロータでメインロータの発生する回転モーメントを打ち消す構造になっている．タンデムロータヘリコプタはメインロータが2つあり，互いにモーメントを打ち消す構造となっている．従来型のヘリコプタ [3] はロータブレードのピッチを変えることで複雑な運動を実現している．回転するロータブレードのピッチを可変するためにはブレードと連結されたピッチリンクと呼ばれるリンクがブレードとスオッシュプレートと呼ばれる板を連結している．スオッシュプレートはロータと共に回転する部分と，胴体に対して回転しない部分の 2 つに分かれている．スオッシュプレートはロータの回転軸に対して上下の移動と傾けができるようになっている．スウオッシュプレートの上下移動はリンクに接続されたブレードのピッチが一斉に変わることを意味する．また，スオッシュプレートを傾かせることにより，ブレードのピッチ角が場所により変わり，1 回転に一回の割合で周期的に変化する．これは，例えば，ブレードが前に来た時はピッチが小さく，後ろに来たときはピッチが大きいということが起こるということである．以上のように，従来型のヘリコプタは機構が複雑なため，研究者が手軽に研究に適した規模の飛行ロボットを用意することが出来なかった．そのため，研究を開始することが難しかったが，マルチロータの飛行ロボットはモータでロータを回転させ，その回転数を電気的に制御するだけで比較的簡単に飛行を実現することができるため，多くの研究者がマルチコプタを用いた様々な研究に取り組んでいる．

従来型のヘリコプタやマルチコプタ，固定翼の無人機を含めた飛行ロボットの用途を軍事用と民生用に区分し，その用途を簡単に以下にまとめる．

【軍事用飛行ロボット】

大きくは以下のような種類に分類できる

標的機　標的機は軍用機や高射部隊が訓練時に目標として使う無人機である．

攻撃機　主にミサイルを装備して，目標を攻撃する無人機である．

偵察機　各種情報取得を任務とし，偵察目標付近の情報を収集するために可視光カメラや赤外線カメラなどの各種センサを装備している無人機である．

輸送機　極めて危険の大きい最前線の兵員に，武器弾薬など任務達成に必要な物資を輸送する任務に使用される無人機である．

【民生用飛行ロボット】

民生用の無人機については，現在様々な用途が模索されている段階であるので明確な分類はまだ時期尚早のきらいがあるが，現在使用されている用途や企業が実験段階にあるものを以下に挙げる．

農薬散布　日本では農薬散布目的での無人機の使用が多い．

架線工事　山の尾根沿いなどの架線工事のためにケーブルを空中架線するために用いられる．

写真撮影　自然環境の調査や，オリンピックに代表されるスポーツ競技の撮影，映画やテレビなどの作品のための撮影など写真や動画などの撮影目的に使用される．

災害時の情報収集　写真撮影とほぼ同じような使用目的であるが，近年では大きな自然災害が続けて発生しており，その二次災害への対応も含めて，救援活動や救出活動などの意思決定のためにはいち早く，現地の状況を知ることが大事であり，そのような場所に各種センサを搭載した飛行ロボットを派遣して状況を調査することが行われ始めており，今後も期待されている．

荷物配送　通信販売業者や運送業者が，小型の物資の輸送のために飛行ロボットを利用することを検討していることが報道されるようになってきた．現時点では完全な実用化には至っていないが，試験運用を開始する企業も出始めている．技術的な課題の他にも，安全面や関連法規の整備などまだ課題は多いと考えられるが，この分野も今後発展が見込まれる．

警備・監視　近年，工場や商業施設の警備や監視などに飛行ロボットを使用するサービスを提供する企業の発表があったが，すでに警備や監視のために利用されている UGV との連携も模索されている．また，建設現場の資材置き場の警備や監視などの用途に向けた検討もなされている．

犯罪捜査　使用事例としては，武器を所持している犯罪者が在宅しているか確認するために，無人航空機を使用し，在宅を確認した後に警察が突入するといった事案がある．

遺跡調査　遺跡を空から空撮し三次元復元を行って作成した遺跡の立体データを用いることにより，研究を加速させている．

科学的観測　火山の河口付近の観察，近年新たに隆起した離島の観測，森林の植生調査など科学的観測に用いられてきた．

大型建造物の点検　トンネルや橋梁・ダム・高層ビル・送電線等の大型構造物の点検に活用されている．

以上に挙げた活用例の多くは，飛行ロボットに搭載されたカメラで撮影したデータを活用するという意味では共通しているが，現在は多くの研究者や企業が飛行ロボットの活用法について模索している時期であるので，今後の応用に役立てるため，具体的な活用事例を俯瞰し，分類を行った．

1.1.2　近年の飛行ロボットの研究動向

このように，飛行ロボットの活用が模索されており，飛行ロボットの研究 [4] については様々なものがある．2000 年代に入り，マルチコプターの研究が盛んになるにつれて飛行ロボットの飛行制御，自己位置推定，地図作成等様々な研究が活発になされるようになっており，自律飛行の実用化にも目処がついてきた．

本研究では，主に GPS を使用できない屋内環境下において小型飛行ロボットを活用するための検討を行った．屋内の非 GPS 環境下における飛行ロボットの研究については多くの研究がなされている [5]．小型飛行ロボットは小型であるが故にペイロードも小さいため，載せられるセンサの性能も低い．また，地上を走行するロボットのように車輪の回転量から移動量を推定することができず，慣性計測装置などから得られる加速度や角速度を積分することにより姿勢を推定する必要がある．位置の推定は加速度の 2 階積分になるためノイズ等の積分による誤差が大きくなり，そのままでは推定値として使用するのは困難である．このような条件のもとで，小型飛行ロボットの自己位置推定と地図作成をする多くの研究が存在している [6]．多くの研究では小型飛行ロボットは MEMS のジャイロ，加速度計と二次元レーザ距離センサを搭載しており，拡張カルマンフィルタや二次元スキャン式レーザ距離センサによるスキャンマッチングによる自己位置推定と地図作成を同時に行う Simultaneous Localization And Mapping（SLAM）アルゴリズム [7] の研究が活発に行われている．また，SLAM には拡張カルマンフィルタを用いるものの他に，最近ではパーティクルフィルタを用いたものも研究されている．また，二次元距離センサだけではなく，ステレオカメラや単眼カメラを搭載して姿勢や位置を割り出す研究も存在する [8][9]．

更には，三次元距離センサによる立体地図の作成も可能になってきている [10]．その中には，二次元距離センサを搭載した機体が三次元の周辺情報を取得するために，飛行ロボット自身が回転運動をすることにより行う研究が見られる [11]．

このように，従来の研究では，飛行ロボットや，地上を走行するロボットも二次元距離センサによってスキャンを行って，周辺の環境を認識して自己位置推定を行っていることが確認できた．

1.1.3　飛行ロボットの小型化

飛行ロボットは多方面での活用が期待されており，その特性から，特に GPS を使用できない屋内での使用も想定される．屋内は狭い上に障害物などが多く，複雑な空間であることが予想できるため，飛行ロボットは止まることが許されない固定翼機よりも，その場でホバリングと呼ばれる空中停止や，離着陸するための平面があれば容易に離着陸できる回転翼機が適していると考えられる．

自己位置推定機能を付与するために，様々なセンサを搭載した飛行ロボットは，小さいものでも全長全幅が 50cm 程度 [12] の大きさとなっている．しかし，約 50cm の大きさは，飛行ロボットが屋内を自由に飛行するためにはまだ大きすぎると考えられる．日本国内の建物の開口部の規格の標準値で最も小さいものは JIS により 40cm と規定されている [13]．災害などで倒壊しかけた建物の中での捜索などを考えた場合に，天井からのワイヤや配管などのぶら下がりや，什器類が倒れているなど，通常の空間よりも飛行可能空間が更に狭くなっていることも想定されるため，飛行ロボットが屋内を飛行するためには，規格値の 40cm よりも更に小さいことが必要である．

そこで，本研究では飛行ロボットのサイズの目標値を全幅・全長について 30cm 以下と考え，検討を行った．なお，現在入手可能な飛行ロボットの中にも，大きさが 50cm を下回る飛行ロボットも存在しているが，そのような機体は，飛行するために必要な最低限の機能のみが実装されているものの，環境を計測するためのセンサが重量の制限により搭載されておらず，飛行ロボットの研究やサービスへの応用を考えることが不可能である．

まず，小型飛行ロボットのおおよその重量の見積もりをすることで，小型飛行ロボットの今後の課題を明らかにするために，現時点で入手可能な部品や材料を用いた場合にどの程度の重量が実現可能なのかを検討した．検討にあたり，飛行ロボットの基本仕様は，基本構造を 4 枚ロータのクアッドロータ機とし，全幅・全長を 30cm として，できるだけ運搬重量を大きくするもの，とした．飛行ロボットの性能を決定するのはロータである．ロータはプロペラとも呼ばれている．30cm の大きさを実現のためには図 1.1 に示すような一般的なプロペラ配置とすると，プロペラの直径は 15cm 以下である必要がある．

マルチロータに使用するプロペラは模型用メーカから製造販売されているものから選択でき，多くのプロペラがインチ単位で販売されている．プロペラの発生する推力はプロペ

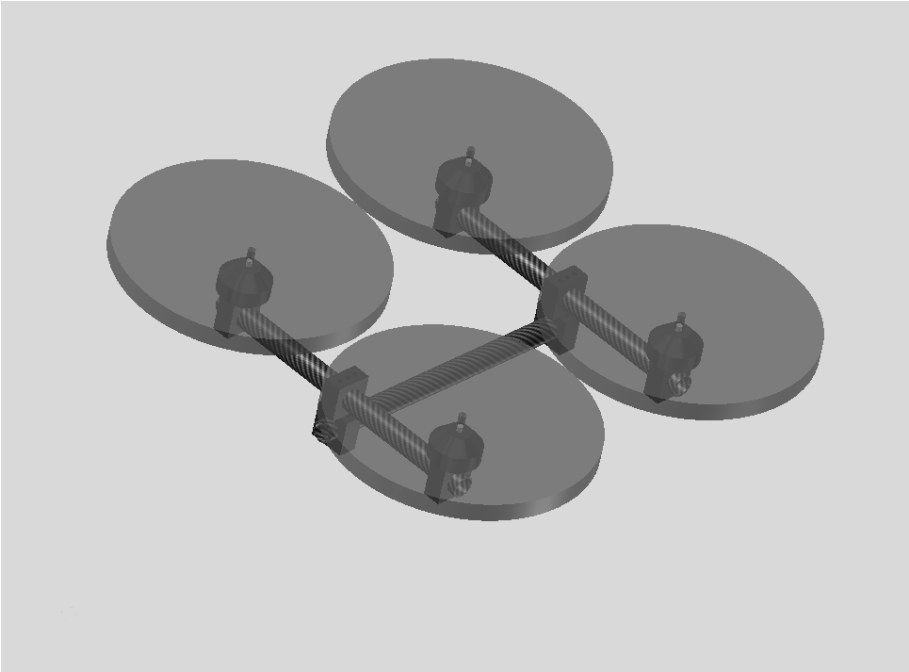


図 1.1　クアッドロータのプロペラ配置

ラの直径が大きいほど大きいため，15cm 以下の直径で最も大きい物を選択しなけらばならない．プロペラの選択にあたっては，販売されているプロペラの基本性能見積がメーカから公開されている APC Propeller から選択した．様々なプロペラが用意されている中で，15cm 以下で最大のプロペラは 5.5 インチ (14cm) である．

プロペラの性能データには付録 A に示すようにプロペラの回転数と進む速さに応じて発生するトルク，発生する推力等のデータが示されている．このプロペラを 4 枚使用した場合について，搭載可能重量と航続時間を見積もった．回転数とトルクの積からプロペラ吸収パワーと呼ばれる，プロペラを回転させるのに必要な動力を計算することができる．また，屋内を飛行する飛行ロボットは低速で飛行し，ホバリングを頻繁に行いながら飛行することも予想できるため，ホバリング状態でどの程度の時間飛行可能か見積もりをした．プロペラ吸収パワーから消費電力が計算できるため，電池の容量が与えられれば飛行可能時間が計算できる．したがって，使用する電池の容量を仮定する必要がある．

本検討では市販されている電池の中から容量が 3700mAh，電池重量が 289g の電池を選択するものと仮定した．5.5 インチのプロペラを使用した場合，約 800gf の重量で 10 分間飛行することが可能なことがわかった．なお，見積に際しては，電力変換効率は 1 としてある．

飛行するために，最低限必要な部品とその個数及び重量配分見積もりを表 1.1 に示した．本検討から，最低限飛行するための重量は 692gf となり，10 分間ホバリングが可能な重量は 800gf であるので，残りは 108gf となる．従って，この重量配分以内のセンサを搭載する必要があることがわかった．

無線操縦ではなく，自律的に飛行ロボットが行動するためには，最低限，自己位置推定能力が必要だと考えるが，オートパイロットに組み込まれている加速度計とジャイロによる慣性演算では時間と共に増える位置誤差を修正することはできないため，距離を計測できるセンサが必要である．

また，自己位置推定の演算を実施するために必要な計算装置も必要だが，Intel の Edison のように x86 アキーテクチャを備えた約 5gf 程の重量しか無い超小型コンピュータも登場しており，計算能力については今後の発展に期待できるところである．

以上から，距離センサについては 100gf 程度ものを搭載する必要がある．しかし，センサについての重量配分が大変厳しいことから，バッテリは容量が小さく軽量なものを選択することとした．飛行時間を犠牲にすることで，より重量の大きなセンサを搭載することが可能となるが，現時点では飛行ロボットの応用分野が明確になっていないため，できるだけ飛行時間を長くする方向での，機体設定を選択した．

また，プロペラやモータなどの推進装置及び制御装置の性能向上により，より重い物の搭載が可能になるため，飛行ロボットの小型化のためには研究すべき分野であり，目標の飛行ロボットの小型化のためには，両方の面からの研究が必要である．

本研究では，搭載機器の軽量化について検討を行い，その中で，飛行ロボットの自己位置推定に不可欠な距離センサの軽量化について検討した．

1.2 レーザ距離センサの軽量化

1.2.1 軽量距離センサの調査

飛行ロボットの小型化には，プロペラやモータの推進装置の性能向上と併せて，搭載物の軽量化が必要である．本研究では，搭載物の軽量化の中で，飛行ロボットに必要不可欠なレーザ距離センサの軽量化について取り上げた．レーザ距離センサには，一方向の距離を計測するセンサ，多方向の距離をスキャンする二次元距離センサ，三次元的に多方向の距離情報を取得できる三次元距離センサなどが存在する．本論文では，一方向の距離を計測するセンサを一軸距離センサと呼ぶ．飛行ロボットの自律飛行に関する研究では，図 1.2 に示すような自己位置推定や地図の作成のために，多くの機体で二次元距離センサが

表 1.1　小型飛行ロボットの部品と重量リスト

部品名称	個数	重量 (gf)
電池	1	289
モータ	4	120
プロペラ	4	36
オートパイロット	1	38
無線受信機	1	9
機体フレーム	1	150
その他	1	50
	合計	692

搭載されている．

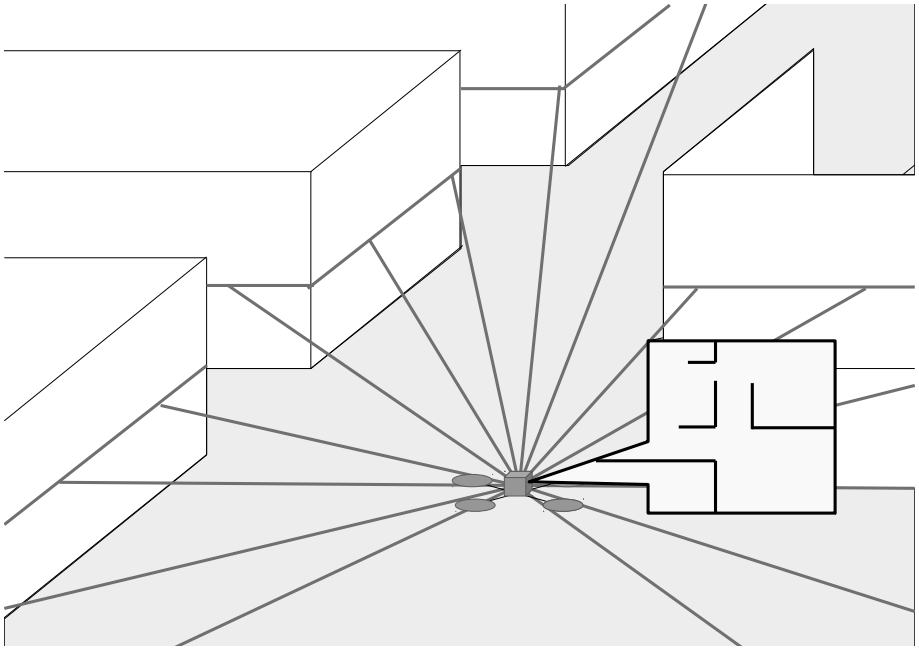


図 1.2　小型飛行ロボットの自己位置推定及び地図作成

飛行ロボットを小型化する際に，表 1.2 に示すように，軽量の二次元距離センサを調査したところ，最軽量のセンサが 130gf 程度であることがわかった [14]．また，二次元距離

センサが周辺をスキャンするために回転機構を有していることから，回転機構を必要としない一軸距離センサは，更に軽量であろうと予想し，調査したが，実際には二次元距離センサと同程度の測定距離を持つセンサは二次元距離センサと同程度か，更に重いものまであった．二次元距離センサの最軽量のものでも 130gf と重く，一軸距離センサにおいても，条件内で見つけられなかったので，センサを軽量化することができないか検討を行った．

表 1.2　距離センサ

メーカ	名称	重量	距離	分解能	方式	接続
Hokuyo	UTM-30LX	210gf	30m	1mm	二次元	USB
Hokuyo	UST-20LX	130gf	30m	1mm	二次元	USB
SICK	TiM300	150gf	4m	1mm	二次元	USB
Nikon	COOLSHOT	165gf	500m	0.5m	一軸	No interface
NEWCONOPTIK	LRF Mod 2	180gf	2500m	1m	一軸	RS232C
Sharp	GP2Y0A21YK	20gf	0.8m	1mm	一軸	Analog
Parallax	Laser Range Finder	50gf	2.4m	1mm	一軸	Serial
SICK	DT60	202gf	5.3m	1.5mm	一軸	Analog

1.2.2　軽量一軸距離センサ実現法の検討

二次元距離センサは，レーザを周囲に照射するための回転機構を内部に有し，重量，寸法共に小型化に支障があるものと考える．一方で，構造を単純化できる一軸距離センサについては，市販されているものは二次元距離センサと比較して重量において際立って軽量だということは無いことが判ったが，軽量化の観点からすると，より軽量なものにできる余地は二次元センサよりもあると考えられる．そこで，本研究では軽量な一次元距離センサを実現することとした．

但し，センサ設計のプロセスでは，仕様決定，詳細設計，制作，試験運用という過程を経ることはせず，二次元距離センサは小型飛行ロボットに搭載するには重いが，近年，軽量化が進んでいる二次元距離センサの光学系，信号処理系を流用することで，開発期間を短縮し，実験等に供することとした．

1.2.3　二次元距離センサの一軸化に伴う熱による誤差の低減

二次元距離センサの光学系や信号処理系の一軸距離センサへの流用で，短期間に二次元距離センサの測定距離と同等の距離を測定できる一軸距離センサが実現できると考えたが，改造には幾つかの処置を行った．レーザ距離計は内部回路が通電により発熱し，その熱によって回路定数が変化するために，同じ距離を測定した場合であっても測定結果が変

わるという特性を持っている．この特性を補正するため，距離既知の物体の計測を実施してその結果と比較した上で誤差分を補正して，出力として得られる仕組みとなっている．二次元レーザ距離計の場合は，レーザ光線は回転ミラーにより全周に振られていて，センサ内部の後方に距離既知の板が置かれており，その板を毎回の計測で測定することで，他の測定結果を補正する仕組みとなっている．回転ミラーを持たない一軸距離センサは測定と同時に距離既知の場所の測定を行って距離を補正することができない．そのため，本論文では種々の補正方式を考えたが最終的にマルチエコー取得機能と呼ばれる，複数の物体からの反射一つ一つの距離を取得できる機能を用いて，距離の補正を実現することを試みた．

なお，本研究では，距離センサの軽量化を目的としているが，軽量化に必要な熱による誤差低減についての手法の実証に関しては，既存のセンサの発光受光回路と光学系を利用することで実験が可能であるため，軽量化を実際に施したセンサは使用していない．実験にあたっては，既存のセンサの回転機構を回転できないように機械的に固定し，内部基準板による補正機能のソフトウェアが動作しないように改造したセンサを採用することとし実験を行った．

1.3　一軸距離センサによる自己位置推定

軽量な一軸距離センサを実現し，飛行ロボットに搭載することができた場合について，従来の飛行ロボットでは二次元距離センサを使用し一度で得られる広範囲の周囲の状況を一軸距離センサでは 1 方向の距離のみしか取得できないため，ロボットの自己位置推定や地図作成においては工夫が必要である．

本研究では回転翼式の飛行ロボットを使用する．この場合は，図 1.3 に示すように飛行ロボットが空中で停止して，その場回転などが行えるため，飛行ロボットそのものを回転させたり，位置を移動して変えることにより周辺情報の取得を行う方法を提案する．その際に，飛行ロボットはホバリング時もわずかに位置がずれたり，回転中に回転中心がずれることがある．この場合は同じ場所で取得したとする仮定が成立する二次元距離センサとは同様に扱えないため，モンテカルロ位置推定のアルゴリズムの中に距離測定結果を取り込んで自己位置推定を行うことを提案し，シミュレーション及び飛行実験によって実証した．

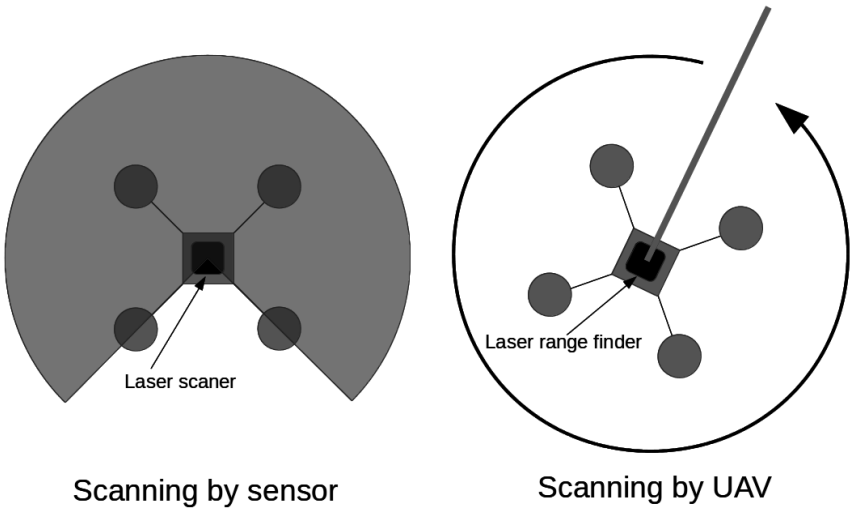


図 1.3　プラットフォームの回転による測域

1.4　本論文の構成

本論文の構成は次のようになっている．

第 1 章では飛行ロボットの研究の現状を述べ，飛行ロボットの小型化についての問題提起を行った．そして，本研究ではレーザ距離センサの小型化の手法を提案し，これを実証することを目的とすることを述べ，最後に提案する一軸センサによる自己位置推定を本研究の到達目標にすることを述べた．

第 2 章では軽量なレーザ距離センサを実現するために，距離センサの原理及び構造を理解し，軽量化の方法の検討について述べた．

検討から，二次元距離センサを応用した一軸距離センサを実現することが最も軽量な距離センサを実現することであることが明らかになった．しかし，二次元距離センサを応用した一軸距離センサの実現においては，二次元距離センサが回転機構を有するが故に実現できていた，誤差の補正を行うことができないため，誤差補正についての検討について述べた．

更に，提案した一軸距離センサ実現手法についての有効性を確認する実験について述べ，実験から新たに見つかった問題と，その対処方法について述べた．

第 3 章では，提案する一軸距離センサはスキャンセンサではないため，一度の計測ではロボットの周辺の状況をつかめないが，飛行ロボットならば，その場での旋回も容易にで

きるため，飛行ロボットの回転や移動によって，ロボットの自己位置推定を実現できないか検討し，従来から研究されているモンテカルロ位置推定が適用できることを述べた．続いて，ここまで実証してきた一軸レーザ距離センサとモンテカルロ位置推定を一軸距離センサ搭載のロボットに応用した自己位置推定について実際の飛行実験により実証した．飛行実験の前には，シミュレーションプログラムによりアルゴリズムの検討を行った．

第 4 章では，本研究のまとめと，将来の展望について述べる．

第 2 章

レーザ距離センサの小型軽量化

2.1　まえがき

本章は，軽量なレーザ距離センサを実現するために，二次元距離センサの仕組みを一軸距離センサへ流用する際に問題となる i) 熱による測定値の変動，ii）実験によって確認された受光強度による測定値の変動について，それらの対処法について述べ，実用化の方法に関し実験による有効性を確認した結果を述べた．

2.2　レーザ距離センサの測定原理

小型化を検討した，レーザ距離センサの測距原理について概要を述べる．レーザ距離センサの測定原理には大きく 3 つの方式がある．一つは，図 2.1 に示す三角測量方式で，測定物から反射したレーザは距離に応じて受光素子に当たる場所に違いを生じるので，その違いを読み取ることで，距離を算出する方式である．

次の 2 つは，どちらも光が目標物まで行って返ってくるまでの時間を計測する方式で Time of flight（以下 TOF) と呼ばれる方式である．図 2.2 に示す方式は位相検出方式と呼ばれ， 被測定物に投光して反射して返ってくるレーザ光の波形は光が空間を旅をした時間に比例して位相がずれる．この位相のズレを計測することによって経過時間を計測して光速を掛けることで距離を算出する．図 2.3 に示すのはパルス伝播方式と呼ばれ，パルス光を発光して，物体に反射してパルス光が返ってくるまでの経過時間を直接計測する．

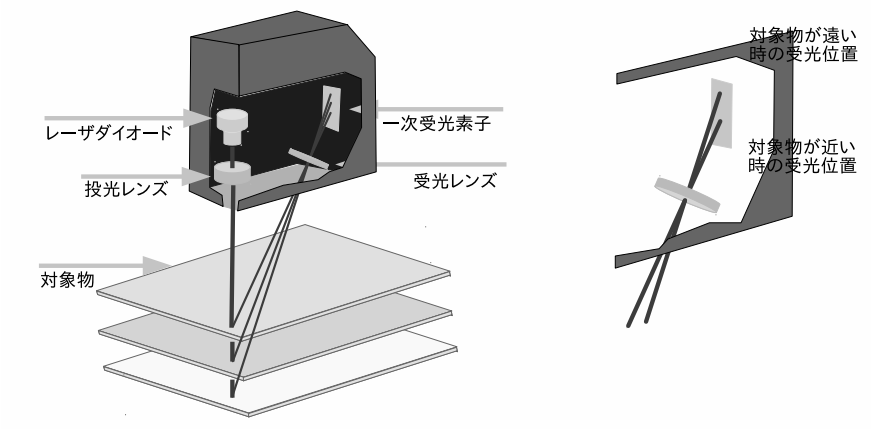


図 2.1　三角測量方式

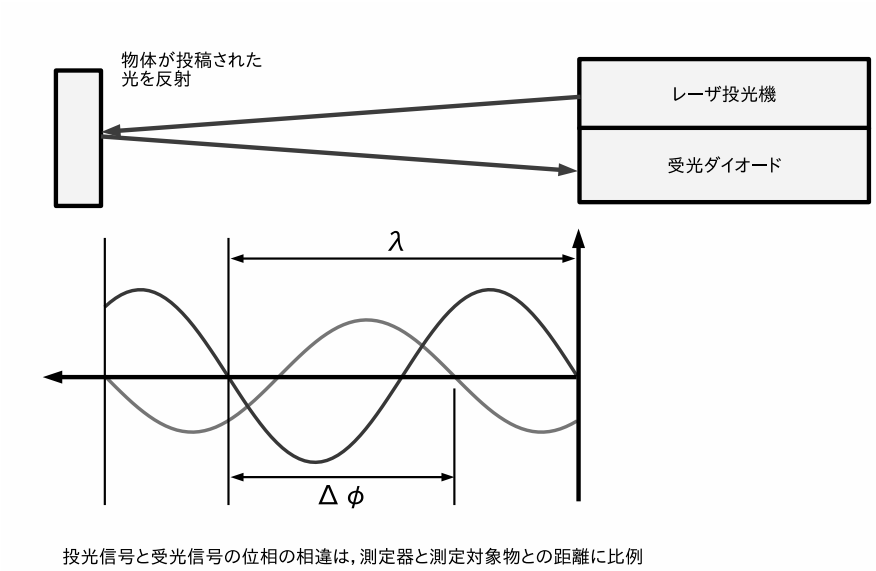


図 2.2　位相差検出方式

2.3　二次元距離センサ

小型軽量化を検討するために，従来から飛行ロボットが使用している二次元距離センサについて，測定原理を述べる．本研究で使用している北陽電機株式会社製の UTM-30LX-EW（以下 UTM）については，パルス伝播方式を採用している [15].

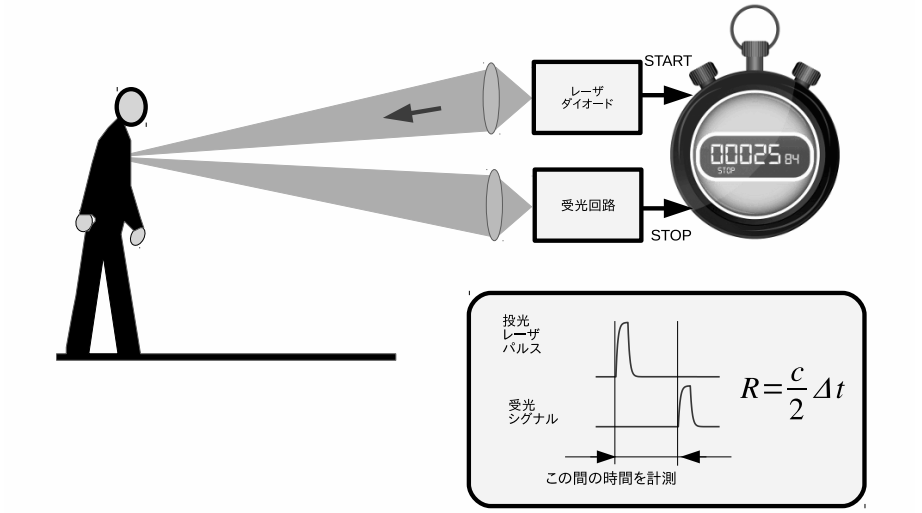


図 2.3　パルス伝播方式

2.3.1　回転ミラーによるスキャン機構

UTM の内部の主要構造は図 2.4 のような構造になっており，ミラーが回転することによりレーザ光を各方向に投光する．ミラーの回転速度に対して光速が圧倒的に高速なため，ほぼ同時に反射光を受光するとみなせる．

現在市販されている二次元距離センサは，すべてミラーをモータによって回転する構造になっており，機械的な機構を有しており，そのための重量を軽減できないため，重量においては軽量化の限界があると考えられる．

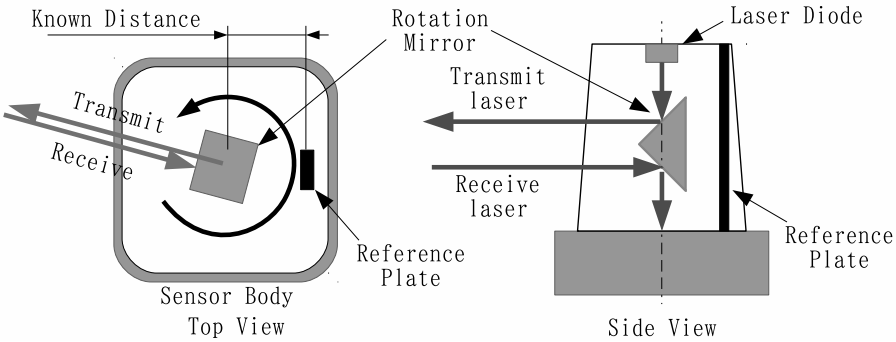


図 2.4　二次元距離センサ内部模式図

2.3.2　マルチエコー測定機能

本研究における二次元距離センサの重要な機能として，マルチエコー測定機能がある．エコーとは物体からの反射光のことである．従来のレーザ距離計は，距離計から最も近い物体までの距離を測定する．これは，測定したい，対象物の前に雨粒やセンサのカバーの汚れ等の物体があった場合はそれらを測定してしまい対象物までの距離が測定できないことを意味する．使用環境によっては，雨や悪天候による泥はね等によるセンサ受光面の汚れが避けられない場合もあり，それらに対応できることが求められた．雨粒や汚れがレーザビームを完全に遮らない場合は，図 2.5 に示すように，ビームの一部は雨粒等をすり抜けて，更に遠方にある被測定物に当たる．センサの受光部は雨粒からのエコーも，被測定物からのエコーも同様に受光できるが，従来は一番最初に取得されたエコーのみの距離を出力していたため，雨や汚れによって測定不能になることがあった．そこで，複数のエコーについて，距離を算出し出力するようにした機能がマルチエコー機能である．これによって雨粒や汚れがあっても，それらより遠方の物体までの距離の測定が可能となった．

なお，出力される複数の測定結果のうち，どれが被測定物までの距離を示すかは使用者の判断に委ねられている．

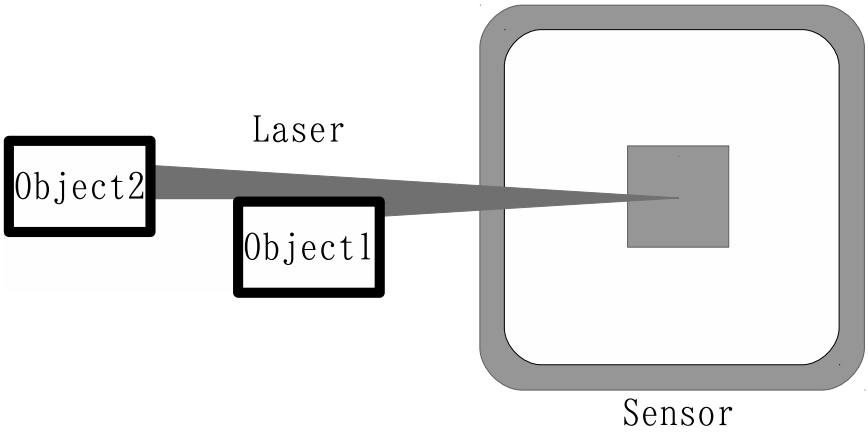


図 2.5　複数物体からのマルチエコーの検出

2.4　軽量レーザ距離センサの実現方法

ここまで見てきたように，レーザを使用した距離センサには様々な方式や形態があるが，周辺の距離を 1 回の計測で複数方向計測可能な二次元距離センサは，内部機構上最も重量が重くなる．一方，二次元距離センサよりも内部機構が単純になるはずの一軸距離センサは，二次元距離センサと比較しても，それほど軽くはないことも判ってきた．

以上から，軽量な距離センサを小型の飛行ロボットに搭載する場合の選択肢は 2 つあると考えた．一つは，従来から最も使用されている二次元距離センサを更に軽量化して，搭載する方法であり，二つ目は，原理上はもっと軽くなるはずの一軸距離センサを使用する案である．

一つ目の方法が早期に実現できるならば，それを実行するのが最も簡単で問題の無い方法であるが，内部に回転機構を持った二次元距離センサの軽量化は，機械的な機構を如何に軽量・小型に設計するかにかってくるため，開発期間とコストに響いてくることが考えられ，それほど早い時期には実現できないと考えている．

二つ目の方法は，二次元距離センサが機械部分を含んで 130gf を実現していることから，二次元距離センサの距離測定に必要な，レーザ発光及び受光部，レンズなどの光学系，発光受光信号を制御し，処理をして距離を算出する部分を活用し，回転機構等の一軸距離センサでは不要な部分を取り除いた距離センサを開発して，軽量な一軸センサを使用する案である．後者は，既に出来上がっているものの流用であるので，比較的短時間で開発が終わると考えられる．

開発を短期間で終わらせ，実用化の検討ができることを考慮すると，後者の案が有力と考えられるものの，飛行ロボットに距離センサを載せる目的に適合するかの検討が必要である．レーザ距離センサは，自己位置推定の目的で使用されることが多い，周辺の地図を持ち，二次元距離センサを用いて計測した周辺の形状と地図とを比較して，自己位置を割り出す方法が一般的になってきている．一軸距離センサで得られる一方向の距離測定結果からでは自己位置を特定できないと考える．しかし，機体の回転や運動と組み合わせることで，自己位置推定が可能になるのではないかと考えられるため，その方法も含めて飛行ロボットに搭載するセンサとして一軸距離センサを検討した．

2.5　レーザ距離センサの熱による誤差

図 2.6 に本センサの発光指令から受光までのタイミングチャートを示した．本センサは，発光指令パルスが出された時刻を 0 秒として時刻の基準とする．発光指令パルスから実際にレーザパルスが送信されるまでには必ず d の遅れが生じる．図の場合は 2 つの物体からの 2 つのエコーを受信する例であるが，第 1 エコーが受信されるまでの時間 T_1 と第 2 エコーが受信されるまでの時間 T_2 を計測する．したがって，実際に光が飛行した時間はそれぞれの計測結果から d を引いた時間となる．

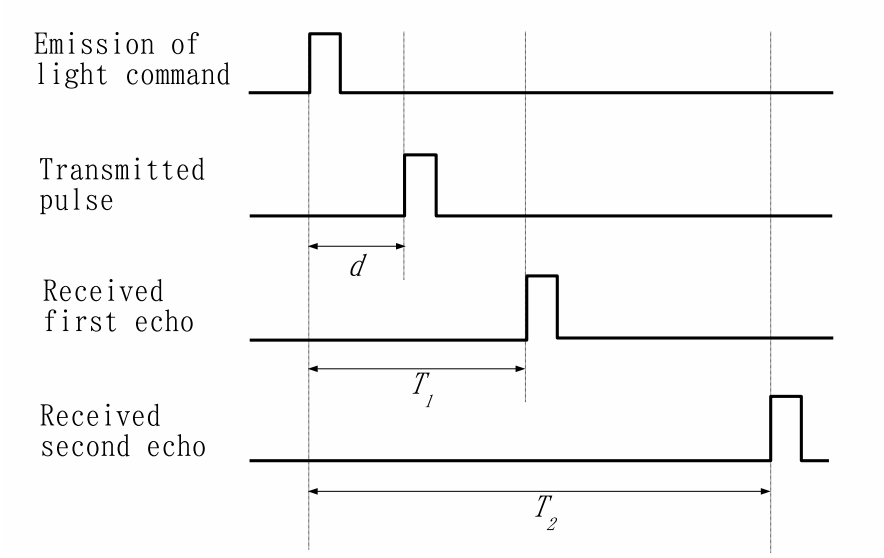


図 2.6　発光受光シーケンスのタイミングチャート

この d はセンサ内部の状態やプログラムの実行速度等の複数の内部要因によって変化する．種々考えられる内部要因の中で，特に大きな変化の要因は制御回路の温度である．レーザー距離計の投光回路及び受光回路はアナログ回路である．通常，電気回路は通電により発熱するが，アナログ回路中の電子部品の性能は熱により変化する．そのため，回路の温度が変化すると，レーザ光の発光に要する時間に影響する．その結果，距離測定結果に変動が生じる．レーザ距離計は，通常何らかの方法で熱を主な原因とする内部要因による変動を考慮し，補正する仕組みが備わっている [16]．例えば，筆者がスキャン機構を取り外して一軸センサにすることを検討した UTM では，図 2.4 に示す位置に距離の基準板が設けられている．センサはスキャンごとに基準板を測定し，基準板までの距離は既知であるので，その測定結果を用いて補正を行い，熱による変動を除去している．

2.6　一軸レーザ距離センサにおける熱による変動の低減方法

2.6.1　一軸距離計における基準板の観測

筆者が用いる一軸距離センサは，二次元距離センサの回転機構を固定しているため熱による変動を除去することができない．この変動は一回の測定で得られるエコーにすべて等しく含まれているために，距離が既知の物体を測定した結果と被測定物の測定結果の引き算により取り除くことができる．したがって，如何にすれば距離既知の基準板と被測定物を測定することが可能になるのかが課題であった．方法としては図 2.7 に示すように，大きく 2 つが考えられる．A) 定期的に距離既知の物体の観測を可能とすること．B) 適当な位置にビームを完全に遮らないように基準板を設定し，基準板と被測定物の両方をマルチエコー機能を用いて測定することである．A) については，マルチコプターに搭載するのであればプロペラを観測することで目的が達成できると考えたが，プロペラは飛行の際に，発生する推力の大小によって曲げモーメントが生じて曲がり，位置が変わることが判ったため，基準にできなかった．その他には，回転する機材を設置する案もあるが，これでは軽量化と逆行するので不採用となった．したがって，基準板を新たに外部に設置するという軽易な処置で目的を達成できる可能性の高い B) 案を採用した．

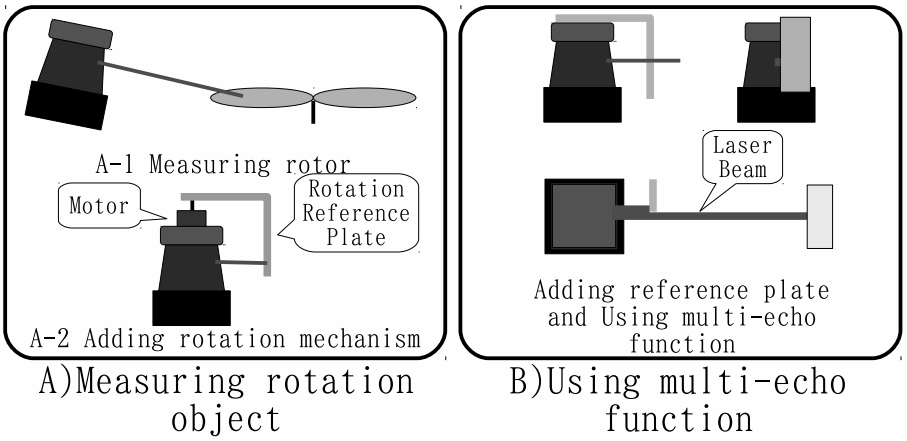


図 2.7　基準板を観測する方法案

2.6.2　基準板測定を用いた熱による変動の除去

被測定物までの距離を l_1 とし，基準板までの距離を l_2 とする．l_1 の値は未知であるが，l_2 の値は既知である．一回の測定における被測定物からのエコーの測定結果が l_{o1}，基準板からのエコーの測定結果を l_{o2} とする．測定出力は，すべてのエコーに共通に含まれる発光遅れによる変動量 Δl とノイズや測定物の反射率の違いなどの要因による測定誤差 e_1,e_2 が含まれていて測定出力は，それらを用いて以下の 2 式となると，仮定する．

$$l_{o1} = l_1 + \Delta l + e_1 \tag{2.1}$$

$$l_{o2} = l_2 + \Delta l + e_2 \tag{2.2}$$

Δl がどのような値になるのか推定することは困難であるが以下のように式 (2.1) から式 (2.2) を引くことによって，取り除くことができる．

$$l_{o1} - l_{o2} = l_1 - l_2 + e_1 - e_2 \tag{2.3}$$

式 (2.3) を整理し式 (2.4) のようにすると，測定値の差分に l_2 を加えたものは，被測定物までの距離 l_1 に誤差 $e_1 - e_2$ を加えたものに等しく右辺は被測定物の測定値を表していると考えられる．

$$l_1 + (e_1 - e_2) = l_{o1} - l_{o2} + l_2 \tag{2.4}$$

以上述べたように， 2 点の測定において一方の距離が既知であれば，ノイズ等による偶然誤差は残るが，熱による系統的な誤差を除去し測定が可能となる．具体的には，図 2.8 に示すようにセンサ前方の距離が既知の位置にビームの一部を遮蔽する板を設け基準板とする．これにより， 被測定物からのエコーと基準板からのエコーの両方が得られ， 式 (2.4) 右辺から被測定物までの距離を測定することが可能となる．したがって，R を測定結果とすると，式 (2.5) から求められる．

$$R = l_{o1} - l_{o2} + l_2 \tag{2.5}$$

2.7　一軸レーザ距離センサの性能確認

センサの発熱よる測定値の変動や，提案する手法の有効性について，実験により確認する．

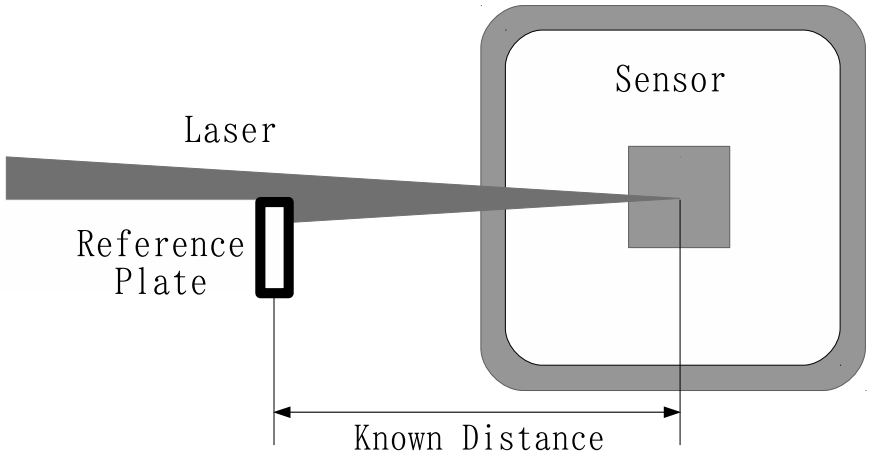


図 2.8　基準板の配置

2.7.1　温度による測定出力の変動

実験に用いるセンサは図 2.9 に示すように，センサ中心から 30mm の位置に基準板を設けた．基準板の材質はアルミニウム合金であるが，ビームが当たる面には，受光素子が強い反射光により飽和してしまうと測定誤差が発生するため 400 番の耐水ペーパを貼り付け，反射率を低下させている．図 2.10(a) のように，センサの近傍に熱電対温度計を設置し，図 2.10(b) に示す厚さ 5cm のスタイロフォーム断熱材で作られた断熱箱に入れる．断熱箱は測定のための約 4cm × 3cm の窓が開けられている．また，断熱箱の右側方には，熱風を入れるため 0.5cm × 10cm の隙間が設けられている．

測定対象はセンサから 2m の位置をメジャーにより測定し，そこに設置した．測定対象の表面についても 400 番の耐水ペーパを貼り付けている．測定については，測定開始からストップウォッチで時間を測定し，約 70 秒間，加熱はせず放置し，約 70 秒後から 150 秒後まで，断熱箱の右側方の隙間からヒートガンを用いて熱風を送り，強制的に断熱箱内部の温度を上昇させた．

図 2.11 は測定結果を示している．縦軸の第 1 軸は距離を示し，第 2 軸は温度を示している．UTM の測定値は mm 単位の整数で返される．この実験結果は，2m の位置の測定対象から返されるエコーによる結果であるが，このデータには発光遅延等による影響が除去されずに入っているために，出力結果は大きく 2m とは異なった値を示している．ヒー

トガンで加熱する前から，断熱箱内の温度が上昇しているのは，センサの発熱によるものである．またヒートガンで熱風を送り，断熱箱内の温度を強制的に上昇させた場合も，温度の上昇に合わせて距離測定値が上昇している．測定の全期間を通じてみると，温度と距離測定結果には相関が認められる．したがって，本実験結果より，温度によらない変動も認められるものの，温度が変動の大きな要因であることは確認できた．

なお， センサは 3ms 間隔で測定を繰り返しているが， 測定値のばらつきの影響を抑えて見やすくするため，図 2.11 では約 1 秒分（333 個）のデータの移動平均をとった値を表示している．

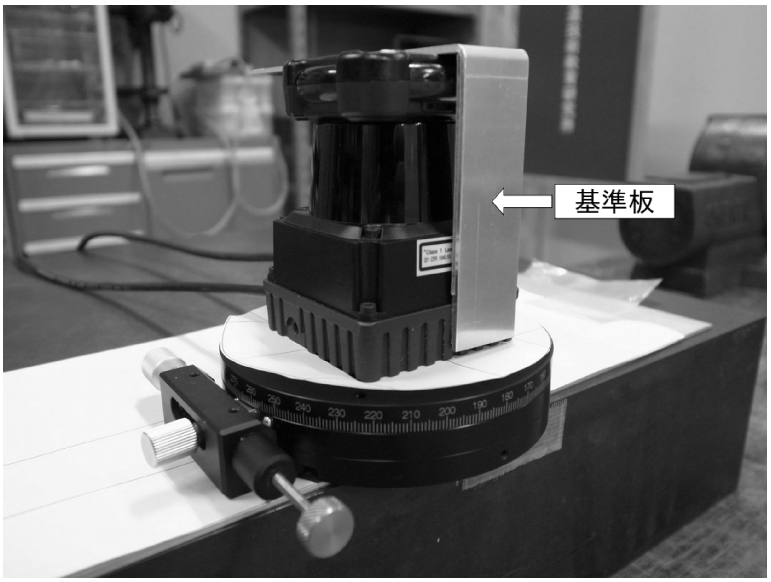


図 2.9　基準板を設置した様子

図 2.12 は本実験の際の，基準板からのエコーと 2m の目標からのエコーの測定値の差分値について示したものである．若干の変動が認められるが，基準板と被測定物からの測定値の差分により温度に起因する測定値の変動に関しては，除去できていることが確認できた．

2.7.2　測定値の算出

図 2.13 に示すように段ボール紙の測定目標をセンサより 1000mm から 4000mm まで 500mm 間隔で 7 箇所に設置し，それぞれを約 15 秒測定する実験を行った．予め目標の位置はセンサ位置から水糸を用いて中心線を割り出し，位置をメジャーで測量し，被計測物

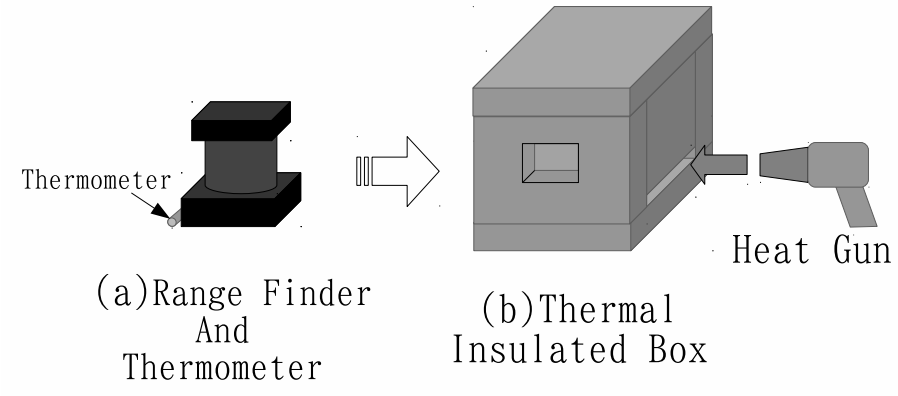


図 2.10　温度特性実験装置

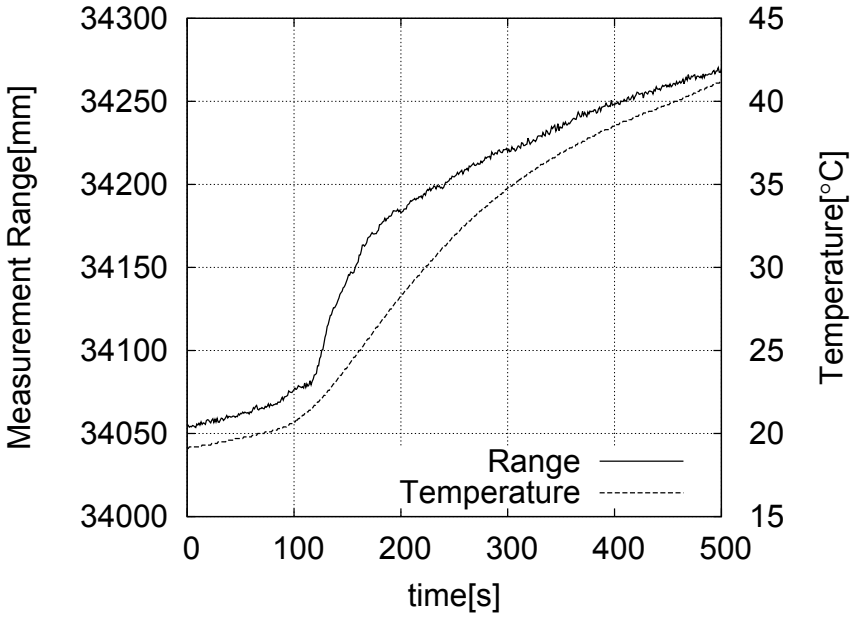


図 2.11　2m の目標測定出力と温度の時間推移

の中央が中心線上に直角になるように直角定規を用いて設置した．図 2.14 は，式 (2.5) に基づいて 2000mm においた測定目標からのエコーから同時に測定された基準板のエコーの測定値を差し引き，基準板の距離 30mm を加えた値を横軸時間として表したものである．なお，測定値のばらつきを抑えて見やすくするために，約 0.1 秒分（33 個）のデータの移動平均による平滑化を施している．

各距離における測定値の平均値・最小値・最大値及び誤差を表 2.1 に示す．この結果を

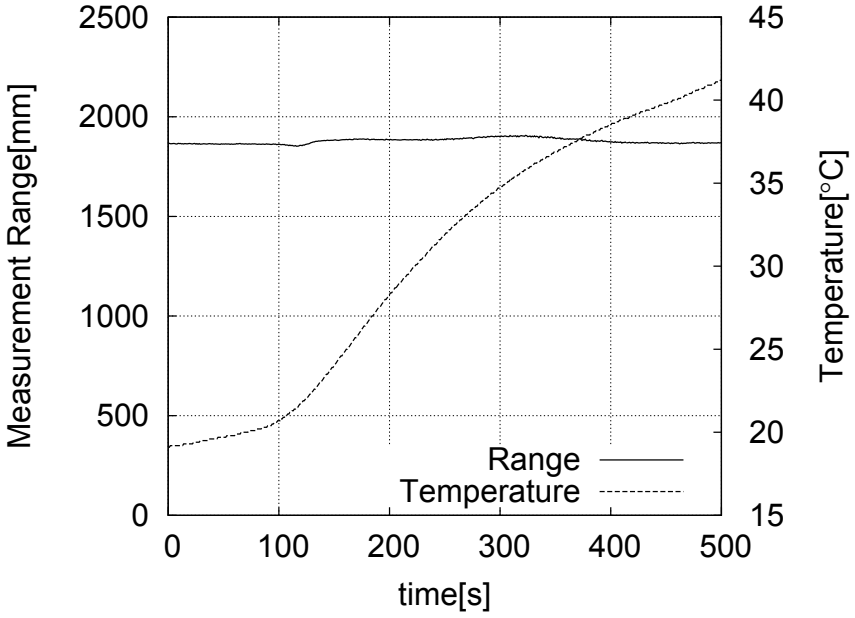


図 2.12　基準板と目標の測定出力の差分の時間推移

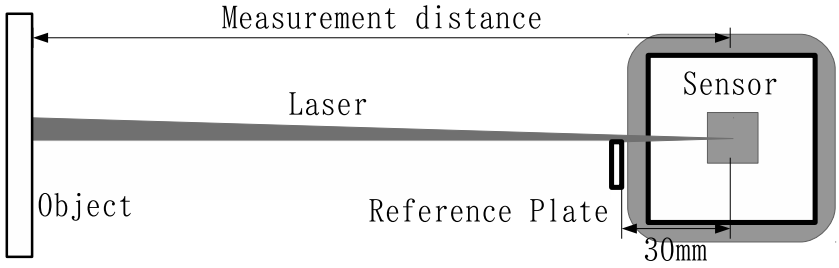


図 2.13　実験の方法

グラフにプロットすると図 2.15 のようになり，実際の距離と測定値が一部ずれていることが確認できる．

2.8　受光強度による補正

UTM の出力には各エコーに対応する計測物までの距離を表す出力の他に，受光したエコーの強度に関係する値を取得することができる．この値は，受光素子の出力値をアナログデジタル変換したもので受光強度に対して相対的な値となる．以後，この値を受光強度と呼ぶ．この値はメーカからは物理量との関係が明らかにされていないため単位について

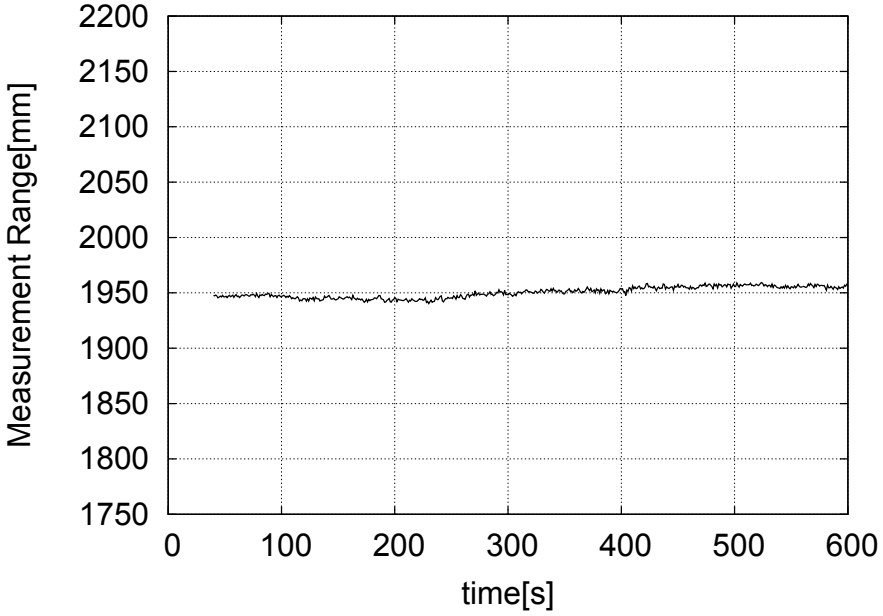


図 2.14　2000mm の測定目標の測定結果を平滑化した結果

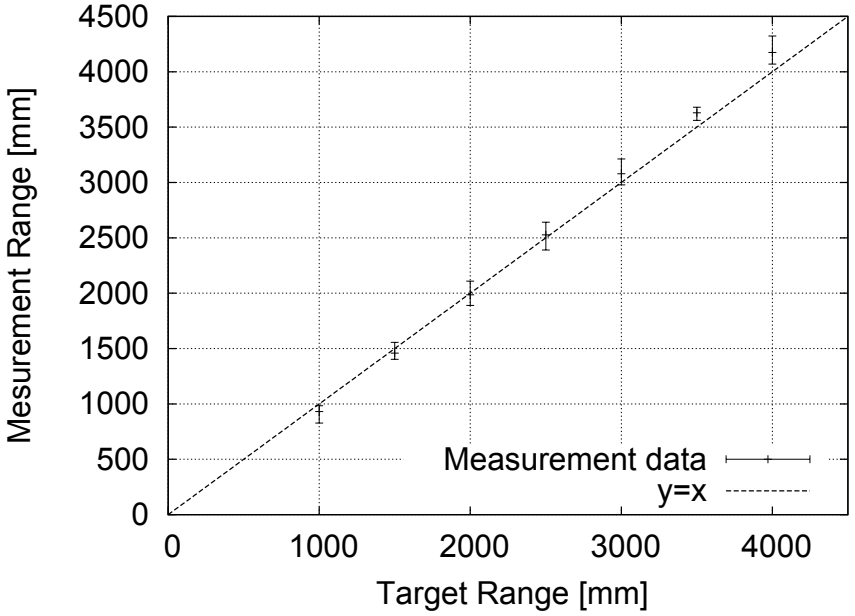


図 2.15　実際の距離と測定結果

表 2.1　距離と測定値

距離	測定値 (mm)			
(mm)	平均値	最小値	最大値	誤差
1000	880	781	929	120
1500	1377	1324	1468	123
2000	1873	1783	1989	127
2500	2384	2254	2491	116
3000	2904	2809	3030	96
3500	3423	3358	3470	77
4000	3936	3837	4076	44

は具体的に示すことができない．

実験結果では，目標の距離と測定結果には誤差が確認されたため，受光強度と誤差に関連が無いか調べることとした．受光強度は測定対象の表面の状態に大きく左右されるため，測定対象の材質をケント紙，段ボール，紙やすりなどに変えて，1m から 4m まで 0.5m づつ距離を変えて測定し，受光強度を横軸，距離の誤差を縦軸にしてグラフ化したものが図 2.16 である．示されている直線は，計測データから求めた回帰直線である．回帰直線の方程式は以下のようになり，受光強度と誤差との関係を示すものである．

$$e = \alpha I + \beta \tag{2.6}$$

式 (2.6) において e は測定誤差，I は受光強度を表す．

実験結果から $\alpha = -0.09548$,$\beta = 106.48$ となった．測定値 R と目標までの距離を l_1 とすると誤差 e は次式となるので

$$e = R - l_1 \tag{2.7}$$

受光強度 I から誤差 e が推定できるとすると

$$l_1 = R - e \tag{2.8}$$

ここで式 (2.5) と式 (2.6) を R と e に代入して整理すると

$$l_1 = (l_{o1} - l_{o2}) + l_2 - (\alpha I + \beta) \tag{2.9}$$

となり，式 (2.9) 左辺と基準板の距離 l_2 と，受光強度と誤差の回帰直線のパラメータ α，β 及び，センサのエコー l_{o1}，l_{o2}，受光強度 I より，目標の距離を算出することができる．

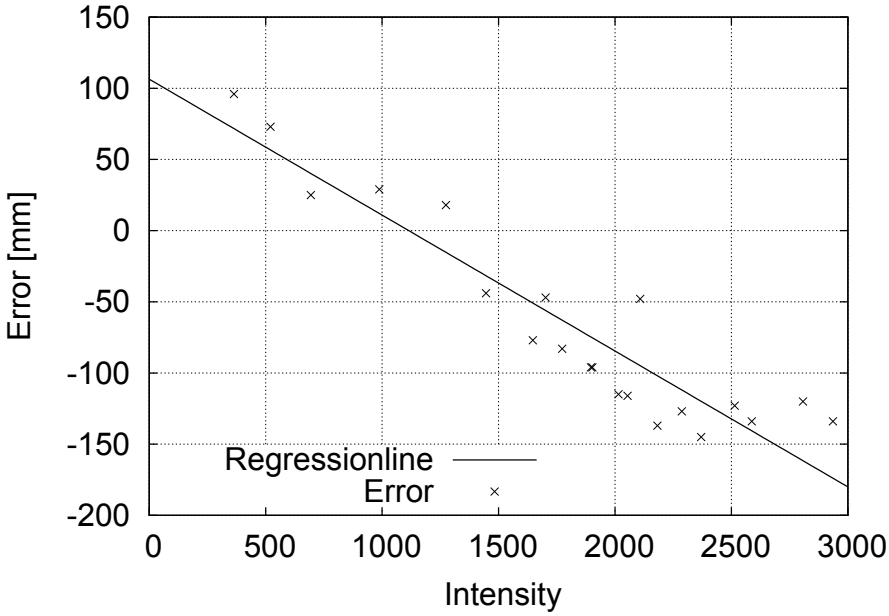


図 2.16　受光強度と測定誤差の関係

2.9　センサ回転による二次元測定

新たな実験式の有効性を確認するために，実際の使用条件を模擬して，センサを回転させて擬似的なスキャン測定を実施した．

測定状況は図 2.17 と図 2.18 に示すとおり，3 個の箱を奥行き方向にずらして配置し，正面を 90°として左右に 80°～100°までセンサを精密調整機構付きのターンテーブルに載せ，0.5°ずつ回転させて，それぞれの方向における距離を測定した．測定値は約 1 秒間の値の平均とした．測定距離 R と測定方向 θ から，測定点の座標の算出は次式によって行った．

$$x = Rcos\theta \tag{2.10}$$

$$y = Rsin\theta \tag{2.11}$$

式 (2.10), 式 (2.11) において，x 及び y は直交座標系の x 座標，y 座標を表す．

測定結果を図 2.19 に示す．

段ボール箱の外形に沿うように測定点が確認でき，測定が可能であることが示された．また，段ボールの面に垂直方向の測定誤差の最大値と最小値は +78mm から-15mm で

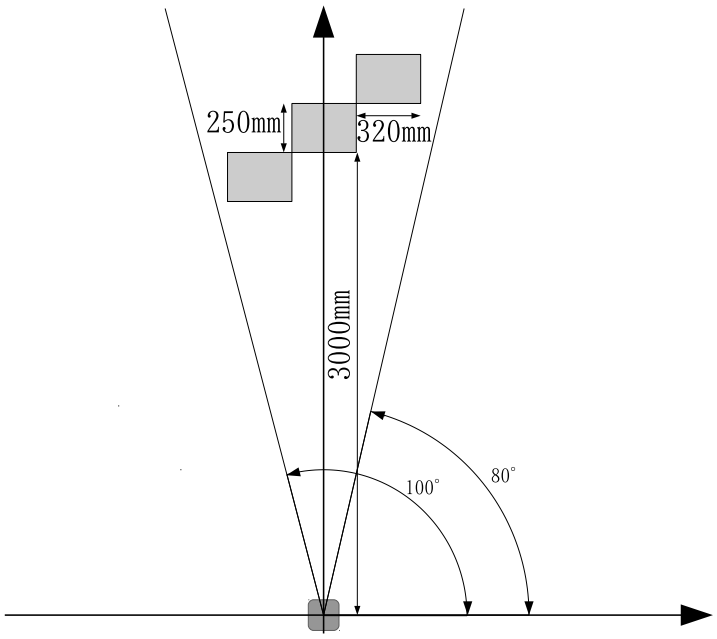


図 2.17　センサ回転による擬似二次元測定

図 2.18　測定の状況

あった．この結果から，基準板を設置してマルチエコー機能を用いた測定では，センサの距離出力と受信強度を実験式に適用することにより，目標からのエコーだけでは熱の影響による誤差が大きく計測器としては成立し得なかったものが，受光強度による変動を考慮した形で測定することができることが示された．

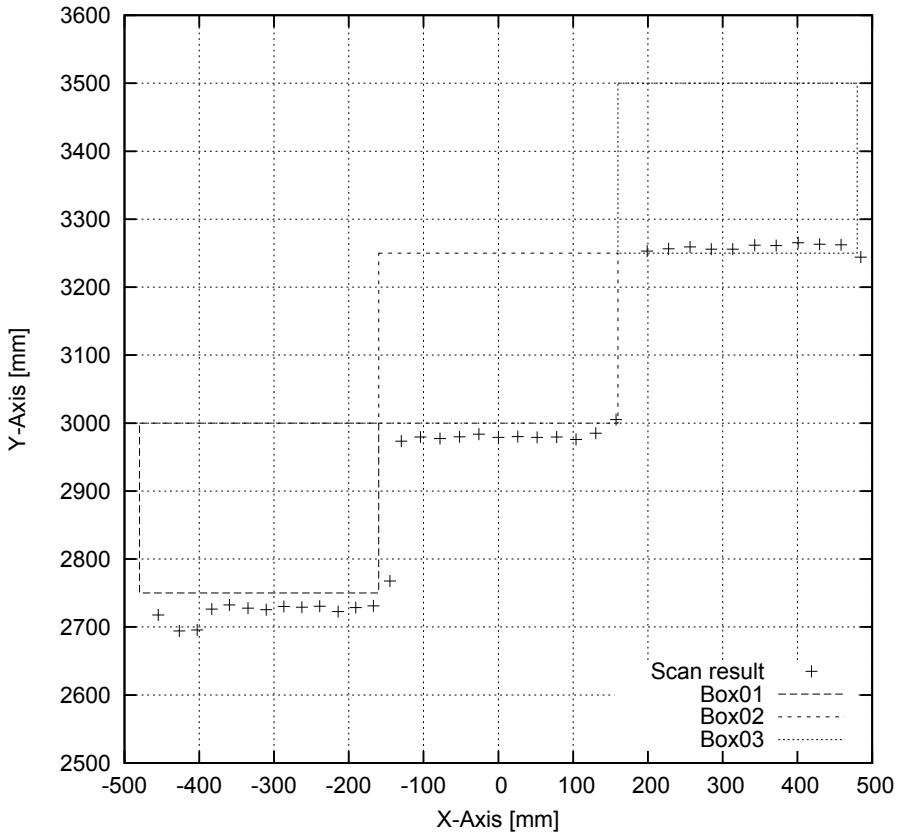


図 2.19　センサ回転測定結果

2.10　回転テーブルによる連続回転下での測定

筆者は，本センサを飛行ロボットに搭載して，飛行ロボットと共に本センサが回転することで周囲の状況を取得し自己位置推定に用いたいと考えた．そのため，本センサにより周辺状況を取得した際に，周囲の状況と一致するかどうかの確認を行った．

図 2.20 に示すように，本センサを飛行ロボットに取り付け，飛行ロボットごと回転テーブルに固定して載せ，飛行ロボットの動力を用いて約 1Hz 程度の早さで連続的に回転し測定する実験を行った．計測のために用いた環境は，図 2.21 に示すようなコンクリートの列柱と壁により四方が囲まれた環境である．

その際，センサの方向を取得するために運用場面を想定して UAV に搭載する Microstrain 社の Attitude Heading Reference System (AHRS) である 3DM-GX3 のジャイロ

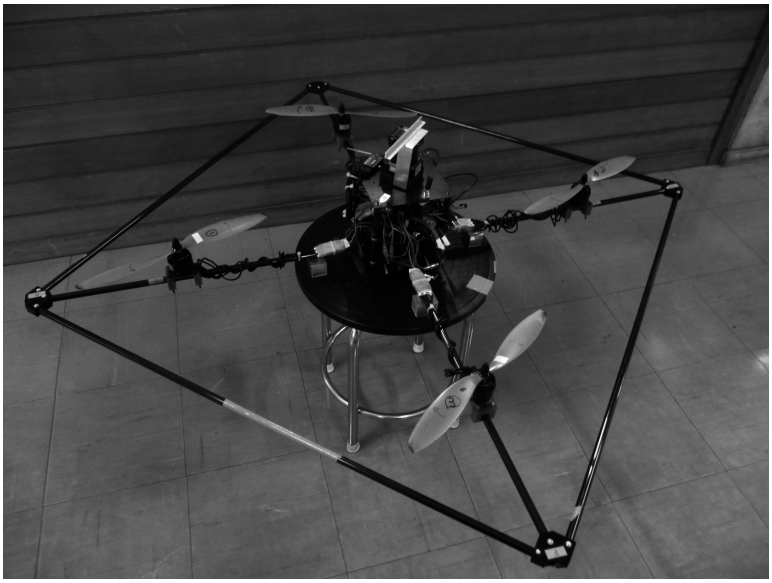


図 2.20　ターンテーブルによる回転装置

図 2.21　計測環境

出力を台形積分して用いた．3DM-GX3 のジャイロの仕様を Table 2.2. に示す．

表 2.2　3DM-GX3 のジャイロの主な仕様

Measurement range	± 300° /sec
Non-linearity	± 0.03 % fs
In-run bias stability	18° /hr
Initial bias error	± 0.25° /sec
Scale factor stability	± 0.05 %
Noise density	0.03° /sec/$\sqrt{\mathrm{Hz}}$
Alignment error	± 0.05°
User adjustable bandwidth	440 Hz max
Sampling rate	30 kHz

本センサの出力と 3DM-GX3 の方向データからセンサが測定した物体の位置を二次元グラフにプロットすると図 2.22 のようになった．図 2.22 において原点がセンサを設置した位置である．黒い点群が本センサを用いて回転テーブルで連続的に回転させて得られた周囲状況である．灰色の点で示された点群は比較のために二次元距離センサ UTM-30LX を用いて測定した結果である．UTM-30LX の測定結果と比して，センサをその場で回転することで周辺の状況を取得することが可能であることが確認できた．測定誤差については，本センサと基準にした UTM-30LX の計測では，計測角度間隔が一致していないので，必ずしも同一の方向を観測しておらず，点と点の対応により誤差を求めることはできない．そのため，点群が形成する線分同士の距離を誤差としてグラフから読み取ると本計測においては最小値-100mm，最大値 60mm 程であった．

グラフ上で灰色の点群が存在するにも関わらず，黒い点群が付近に存在しないところがあることが確認できた．また，黒い点群の密度が疎のところも確認できた．点群が確認できないところは距離が 9m から 10m の範囲にある地物を計測できておらず，黒い点が疎のところは，データを確認すると受光強度が小さいところであった．本実験データ中の受光強度の最小値は 34，最大値は 2615 であった．黒い点群が疎のところはほぼ，受光強度の値が 300 未満であり，300 を切るような計測データを得られた前後の部分の計測は物体が存在していても計測値を得られておらず，計測を失敗する確率が高くなることが確認できた．受光強度が小さくなるのは距離が遠い場合と，レーザが当たる角度が計測面に対して小さい時であった．

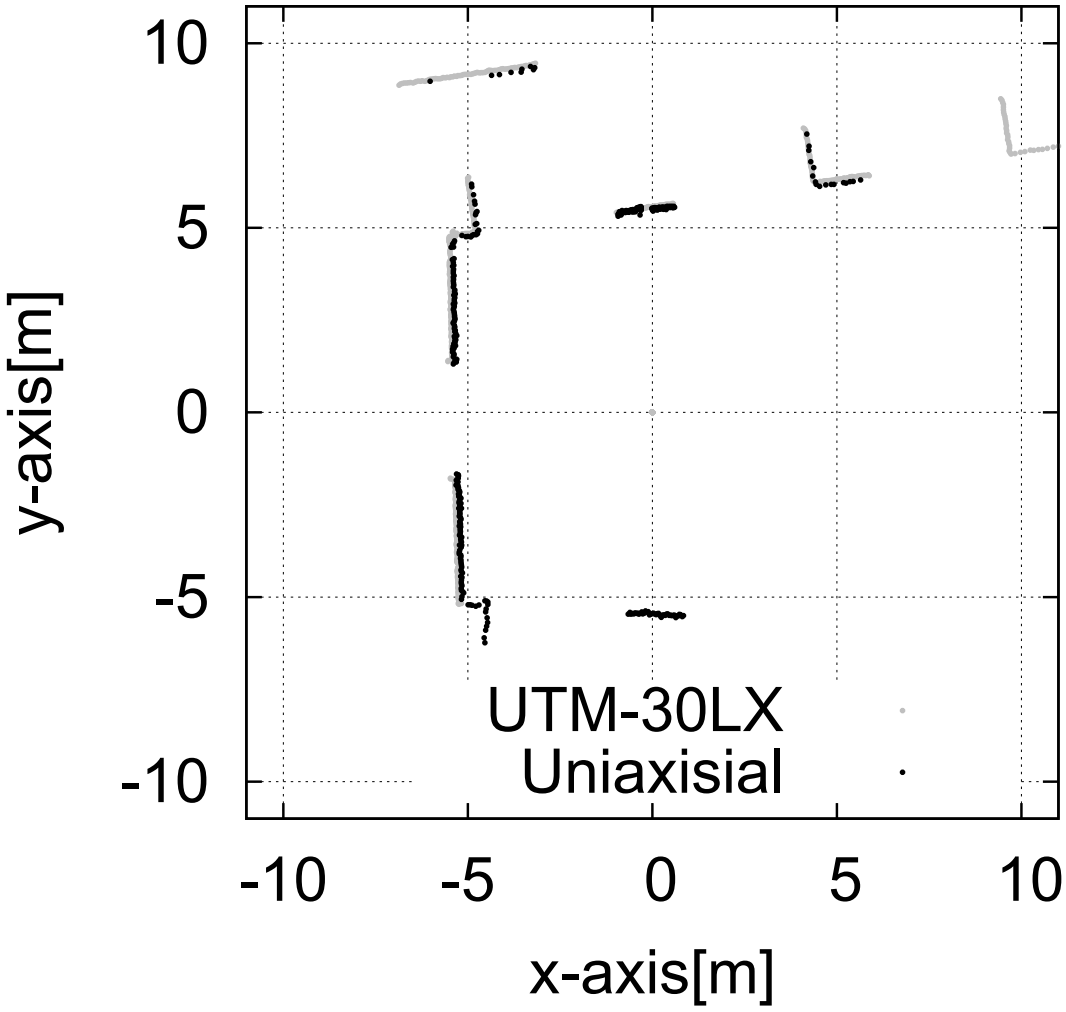


図 2.22　スキャン式センサとの比較

2.11　あとがき

小型飛行ロボットの自律飛行を実現するために，搭載機器の小型化からのアプローチを検討する際に，最も重要なセンサである距離センサの軽量化が第一だと考え，距離センサの小型化について検討を行った．現在では小型軽量の二次元距離計の開発が進み，相当な

小型化が実現しているため，二次元距離センサに用いられている技術で小型の一軸距離センサを開発することで，軽量・超小型の一軸距離計を実現できると考えた．

製品の距離センサは様々な原因による誤差の低減対策が組み込まれ，最終的には機器固有の特性に合わせて調整を行って出荷される．誤差の低減対策の中で，回路の発熱による誤差の問題は，全ての製品に共通に起こることが判った．既存の二次元距離計は回転する機構があるため，測定ごとに基準板を読むことができるが，回転機構が働かない場合は基準板を読むことができず，外気温や回路の熱によって大幅に測定値に誤差が発生してしまう．そこで，外部にレーザビームを一部だけ遮る基準板を設けて，反射光をマルチエコー機能を用いて読み取ることで，被測定物の距離と基準板の距離の両方を取得して，差分を取り誤差を軽減する手法について提案し，距離測定実験，静的な回転実験，動的な回転実験を行い，有効性を確認した．その中で，熱による誤差の他に，受光強度による誤差があることも確認し，受光強度による誤差についても低減することができた．

次章では飛行ロボットに搭載した時に，ロボットの動揺が測定にどのような影響を与えるのかを確認しながら，本センサを二次元距離計の代替として使用した自己位置推定について述べる．

第 3 章

一軸距離センサを用いた飛行ロボットの自己位置推定

3.1　まえがき

本章では，一軸距離センサを搭載した飛行ロボットに予め地図を持たせて，一軸距離センサから得られた距離データを活用して，自己位置推定を行う手法について述べる．

距離を計測するセンサが一軸距離センサの場合，飛行ロボットは，一方向の先にある物体までの距離を計測出来るだけで，一度の計測で周囲の形状を取得することはできない．そのため，図 3.1 に示すように，飛行ロボットが地図内の何処かに存在するとして，方向も未知とすると，一軸距離センサから得られた距離情報が得られる可能性のある場所は広い領域に渡って存在する．これは，距離センサが一度に複数の方向を計測することができれば，計測した複数の距離情報から，飛行ロボットが存在する可能性のある領域が限定できることを意味する．二次元距離センサは多いものでは 1000 方向以上の周辺距離を測定できる．したがって，一度の計測により地図情報の誤差と，距離センサ誤差の範囲内に収まる精度で自己位置を推定できる．

このように，一度の計測で一方向の距離の情報しか取れない場合においては，一度の計測で，自己位置推定をしても，場所を絞り込むことができない．しかしながら，回転翼の飛行ロボットの特性を活かし，ホバリングをし，定点において回転することができれば，一軸距離センサを用いて，二次元距離センサと同様の自己位置推定ができる．回転翼飛行ロボットは原理的には，その場で停止しながら，Z 軸周りの回転が可能である．しかし，飛行ロボットの電装品は一つの種類を一個しか積んでいない場合もあり，対称配置が難しいため，飛行ロボットの重心が回転軸を通っていないこともある．そのような場合は，一

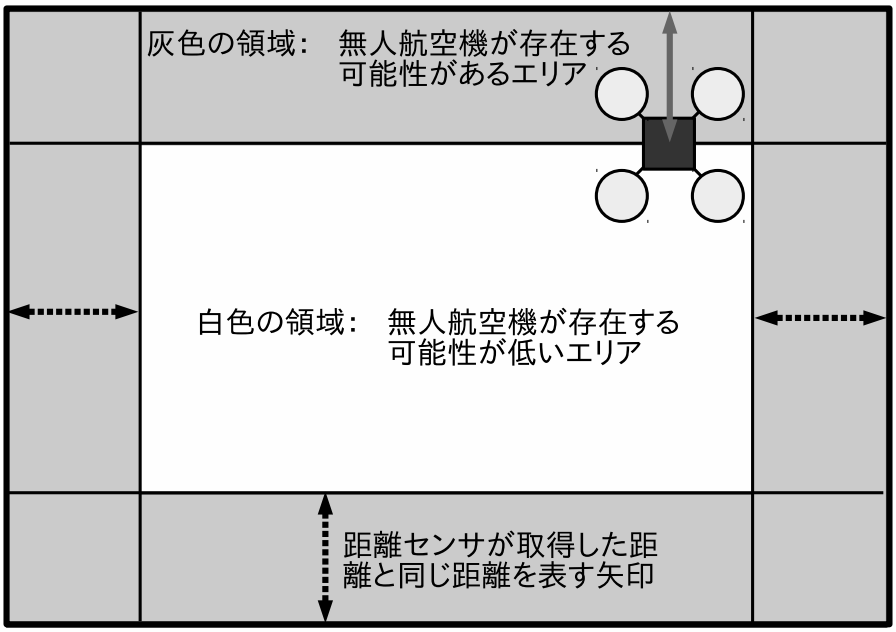


図 3.1　一軸距離センサによる自己位置推定

本の軸に貫かれたように回転せず，周囲に円弧を描きながら移動しつつ回転する場合も多く見られる．機械的なアンバランスがあるとしても，制御によりその場回転も可能になると考えられる．

したがって，一軸距離センサを搭載した飛行ロボットによって自己位置推定をする方法として，回転しながら定点に留まる位置制御を施した上で，回転しながら複数の方向の距離データを取得し，ジャイロのデータから方向を求め，得られた複数方向の距離データと地図データから自己位置推定をする方式が考案できる．一方で，位置制御を施さず回転し，周囲の距離データを取得する方式も考えられる．この場合，できるだけ機械的にバランスを調整し，少しの移動を伴うことは許容したうえで行う．前者の場合は，位置制御をするために，位置を知るためのセンサを積む必要がある，前提として一軸距離センサを一つだけ積むことにしているため，距離センサを複数付けて位置を知ることはできない．また，小型軽量化を目指した時に，他のセンサを更に増やして位置の取得をすることも考えにくい．したがって，飛行ロボットの小型化を追求した場合に，位置制御が理想的にできることを期待しないほうがより現実的であると考えた．よって，本研究では，位置制御を施さない飛行ロボットに一軸距離センサを搭載して，回転して周囲の距離を取得して自己位置推定を行う手法について検討を行った．

検討については，飛行ロボットのその場回転の飛行実験を行い，その際の距離情報を，

地図情報と重ね合わせを行い，自己位置が推定できるか確認した．そして，一軸距離センサを用いた自己位置推定では一度の計測で自己位置を推定することは不可能だが，回転しながら取得した複数の距離データから推定できる自己位置の候補領域を計測ごとに重ねあわせていくことにより，自己位置が絞り込めると考えた．これは，多くの無人機の自己位置推定に用いられているモンテカルロ位置推定（Monte Carlo Localization，以下 MCL）[17] の手法のようにベイズ更新を繰り返していくことで，推定精度を向上する手法と同様であると考えて，一軸距離センサを搭載した飛行ロボットの自己位置推定に MCL を適用し，有効性を確認した．

3.2　一軸レーザ距離センサによる周辺環境の認識

実験用の回転機構を固定しスキャン機能を止めた UTM-30LX をクアッドロータの実験用飛行ロボットに搭載し，ホバリング状態から Z 軸周りに旋回させて周辺の距離データを取得し，地図データと重ね合わせを行い，それによって自己位置が推定可能かどうか確認した．

3.2.1　実験装置

実験装置の概要を以下に述べる．

実験用飛行ロボット

今回開発した一軸距離センサを搭載して実験に用いる飛行ロボットは，図 3.2 に示す研究室で設計製作した 4 つの直径 14 インチのプロペラをもつクアッドロータヘリコプタである．機体の大きさは周辺物との衝突による墜落を防止するガードを含めた大きさが縦横 0.95 m高さ 0.3m 程である．4 枚のプロペラは互いに反動トルクを打ち消すために 2 枚づつ反対方向に回転するようになっている．実験用の機体のシステムは図 3.3 のようになっている．機体は 3D Robotics 社製のオープンハードウエアの arduino マイコンボードである APM2.6 によって姿勢制御される．姿勢制御のソフトウエアはオープンソースソフトウェアを使用している．APM2.6 にはモータの制御のための Electoric Speed Contorller（ESC）が接続されている．ESC には PWM 信号が入力されてモータを制御する．ESC には高トルクのブラシレスモータが接続され，14 インチのプロペラを回転している．機体の重量は電池込みで 3.0kg 程であり，この機体の最大推力は 4.8kg 重程であるので，1kg 以上のペイロードを積むことが可能である．

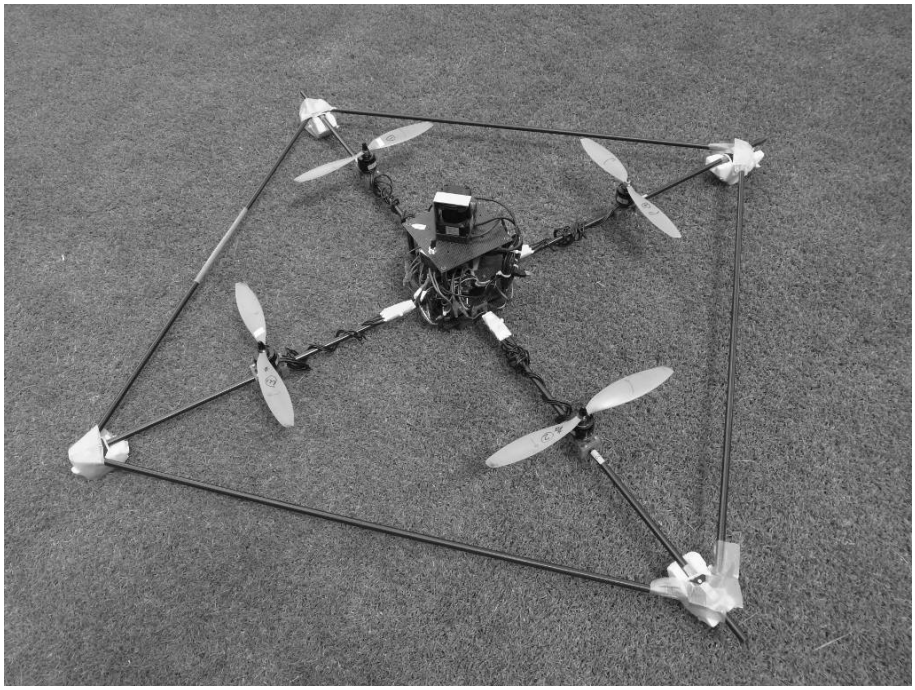


図 3.2　クアッドロータヘリコプタ

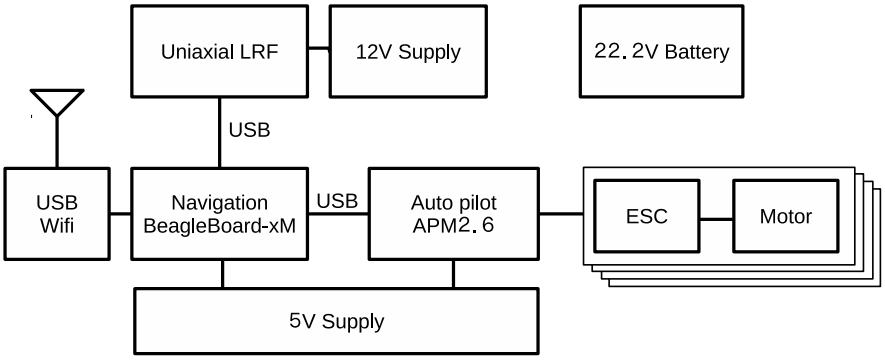


図 3.3　クアッドロータヘリコプタシステム

姿勢・高度制御用マイコンボード

クアッドロータヘリコプタの特性は不安定であるので，制御によって安定化を図る必要がある．そのため，角速度と角速度の積分値を角度としてフィードバックを行い，安定化制御を行う．安定化制御用に図 3.4 に示す，APM2.6 を使用している．APM2.6 にはジャイロ，加速度計，気圧計が搭載され，磁気コンパス，超音波による高度計が接続され

ている．制御則は PID 制御が用いられている．

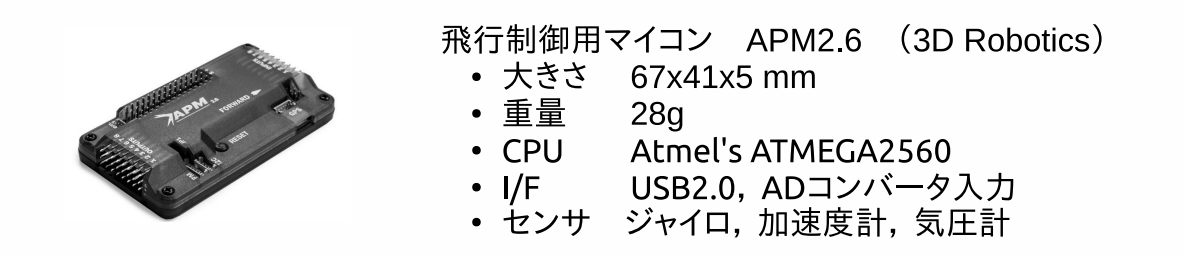


図 3.4　飛行制御用マイコン

ナビゲーション用組み込みボードコンピュータ

実験用飛行ロボットは一軸レーザ距離センサや慣性計測装置からのデータを保存することや，将来的には各種センサからのデータを用いた自律飛行ができることが要求される．そのため，前述の APM2.6 では性能が劣るため，図 3.5 に示す，組み込みボードコンピュータの BeagleBoard-xM を搭載している．この組み込みボードコンピュータに Linux OS の Ubuntu14.04 をインストールしてあり，無人機オペレーションシステム ROS をインストールすることでセンサからの情報収集，自律飛行時のナビゲーションなどを実行する．APM2.6 はこのボードに USB インターフェースにより接続され，相互の通信を行う．

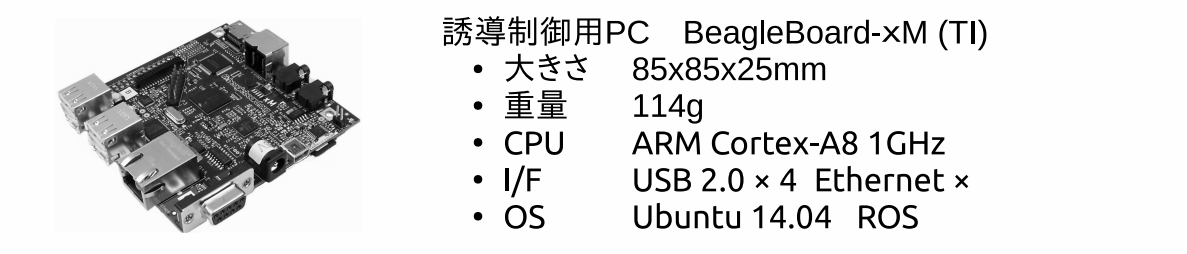


図 3.5　誘導制御用ボードコンピュータ

慣性計測装置

自己位置推定用の慣性計測装置として前章の実験でも使用した Microstrain 社の Attitude Heading Reference System (AHRS) である 3DM-GX3 を搭載している．角度，角速度，加速度を取得でき，BeagleBoard-xM と USB インターフェースで接続される．

慣性計測装置　3DM-GX3-25

- 大きさ　44x24x11mm
- 重量　18g
- I/F　USB2.0

図 3.6　慣性計測装置

3.2.2　実験方法

実験は，予め決めておいた場所で飛行ロボットをホバリング状態から Z 軸を回転軸として時計回りに回転させ，一軸距離センサにより距離情報を取得する．

実験環境

図 3.7 に示す，コンクリートの数本の柱と壁に囲まれている環境で飛行実験を行った．

図 3.7　飛行実験環境

クアッドロータヘリコプタの運動と操縦

クアッドロータヘリコプタの飛行原理は図 3.8 に示すように，ロール角を制御したい場合は，左右のプロペラの回転数に差をつけることにより回転モーメントが発生して，ロール軸まわりに回転を生じる．他も，同様の原理により行う．

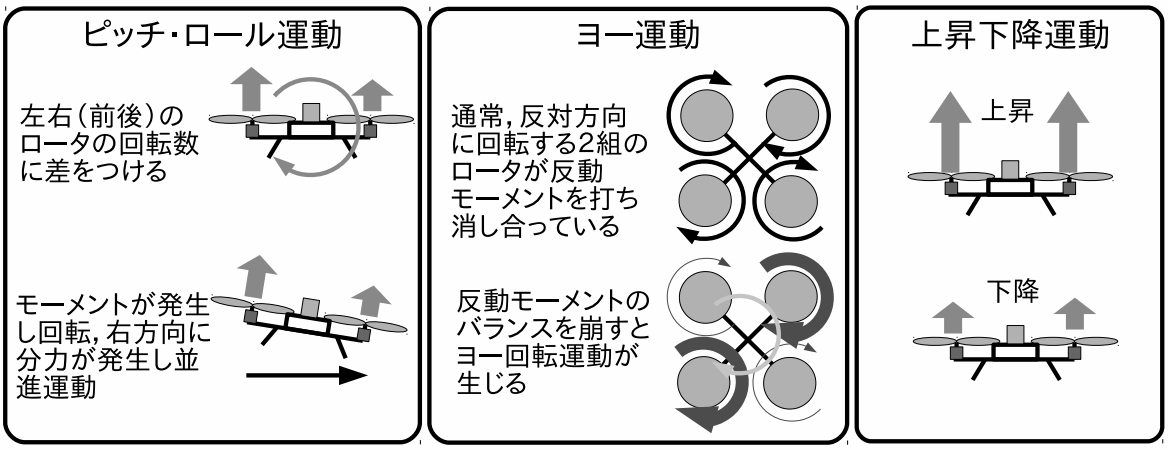


図 3.8　クアッドロータヘリコプタの制御

本研究で用いられる飛行ロボットのコントローラの制御用プログラムは，3D Robotics 社製の Arducopter2.9 が使用されている．この制御プログラムは指定された緯度経度を順に通過していく等，様々な飛行モードを持つが，基本的には姿勢安定化制御モードがあり，その上で機体の誘導制御プログラムが動作する．本研究では機体の高度を一定に保つ高度制御モードを使用し，他の制御モードは使用しない．機体の位置等は目視で手動操縦をした．制御プログラムはソースコードが公開されているフリーソフトの形態を採っているので，目的に合わせてプログラムを改良することができるが，本研究ではプログラムを修正する必要はなかった．

姿勢安定化制御モードはピッチ，ロール系とヨー系で別々の制御則が適用されている．ピッチ・ロール系の制御は，各軸周りの回転角速度をフィードバックして目標角速度に一致させる内側のループと各軸回りの角度をフィードバックして目標角度に一致させる外側のループからなっている．内側の角速度制御系は PID 制御になっており，外側の角度制御系は P 制御として構成されている．目標角度は送信機の操縦スティックの傾き角度で指定する．

ヨー系の制御については外側の角度制御系と内側の角速度制御系によって構成されているが，ピッチ・ロール系の制御とは違い，入力はヨー角速度になっている．違いは，入力を積分して外側の角度制御系の目標値として渡す部分と，入力が分岐して角度制御系の入力としてフィードフォワードとして直接加える構造となっている．

また，各ゲインの調整が可能であるが，本研究ではデフォルトの値を使用し，調整は行っていない．

推力については通常はオープンループの制御で，送信機のスロットル調整用スティックの倒れ角に比例した推力指令が各モータの回転指令に直接加算される制御になっている．一般的には，スロットル調整用スティックを最大値の倒れ角に対して 50% に倒した際に機体がホバリングするように調整されるが，オープンループであるため，電池の消費により操縦に対する応答は変化する．そのため，高度等を手動操縦で一定に保つことは難しく，高度制御モードが用意されている．

高度制御モードは気圧センサや超音波センサによって高度を取得し高度制御を行う．制御則は，加速度制御のフィードバックループの外側に，速度制御のフィードバックループがあり，更にその外側に高度制御のフィードバックループがある構造となっている．

飛行の手順

飛行までの手順は以下のとおりである．

1. 予め決めておいた場所に飛行ロボットを設置し，電源を入れる．
2. Beagleboard-xM の無線 LAN が無線ルータと接続したことを確認する．
3. 操作用端末としてノート PC を立ち上げ，端末プログラムを用いて，Beageboard-xM に SSH を用いて接続する．
4. roscore を実行する．
5. IMU のデータを取得する ROS のノードを実行する．
6. 一軸距離センサーのデータを取得する ROS のノードを実行する．
7. ROS のコマンド rosbag により，IMU の出力と一軸距離センサの出力をロギングする．
8. 離陸しホバリング状態にする．
9. ホバリングが安定したことを確認できたら，ヨー回転指令を与えて，機体を時計回りに回転させる．
10. 最低 1 周したら，回転を停止する．
11. 別の場所に移動し，ホバリング，旋回を繰り返し行う．
12. データ取得が終了したならば, rosbag を停止しする．rosbag はデータをバッファーから，SD カードに記憶するのに時間がかかるため，記録終了を確実に確認して次の手順に進む．
13. 他の ROS のノードもすべて停止する．
14. roscore を停止する．
15. 電源を切る

3.2.3　実験結果

以下に，実験結果を示す．

以下のグラフは，2 箇所で回転運動を 1 周した際に得られたデータである．図 3.9 から図 3.14 において，灰色の線が予め二次元距離センサを用いて取得した実験環境の地図を表し，黒線は一軸距離センサから得られた距離情報とジャイロから得られた角度情報を元に描いた周辺状況である．これらのグラフでは，周辺状況を平行移動，回転の試行錯誤を繰り返し，地図と重ねたものである．図の中で十字マークは，飛行ロボットの位置を表す．飛行ロボットは飛行中に大きく 1m から 2m と大きく移動してしまう場合があり，地図と周辺状況のデータは前章のターンテーブルを用いた回転測定実験の結果のようにはすべての部分が重なり合わない．したがって，それぞれの図中にある (1)〜(6) は取得された周辺状況データの何処を地図と重ねたかを表す．図 3.9 から図 3.11 は離陸した場所で時計回りに一周旋回した際の周辺状況データで，同じデータであるが地図と重ねあわせる場所を 3 箇所変えたものである．旋回中取得したデータは (1) が最初に取得された部分で，続いて（2)，最後が同じところに戻って (3) である．次に，場所を移動してデータを取ったものが図 3.12 から図 3.14 であり，(4) から (6) の順にデータが取得されている．

図 3.15 は図 3.9 から図 3.11 までの回転運動中の飛行ロボットの軌跡を，同様の重ねあわせ手法を用いて，更に多くの特徴のある端点についての飛行ロボットの位置を求め，移動軌跡として描いたものである．途中点の間隔が開いていしまっているところは，左側の空間にセンサが取得できる特徴のある端点がないため，位置を推定できない部分である．

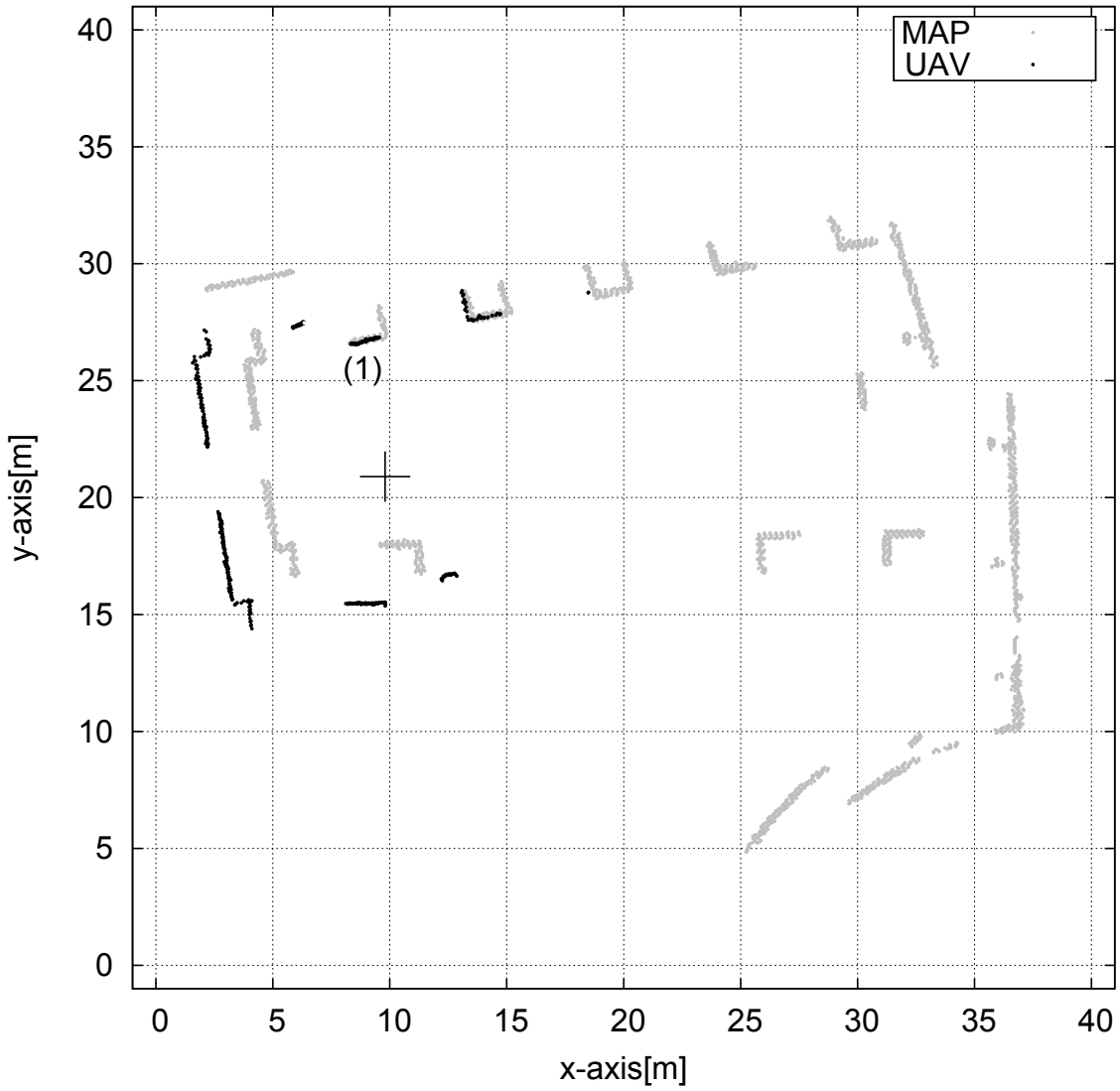


図 3.9　地図と計測データの重ね合わせ（1）

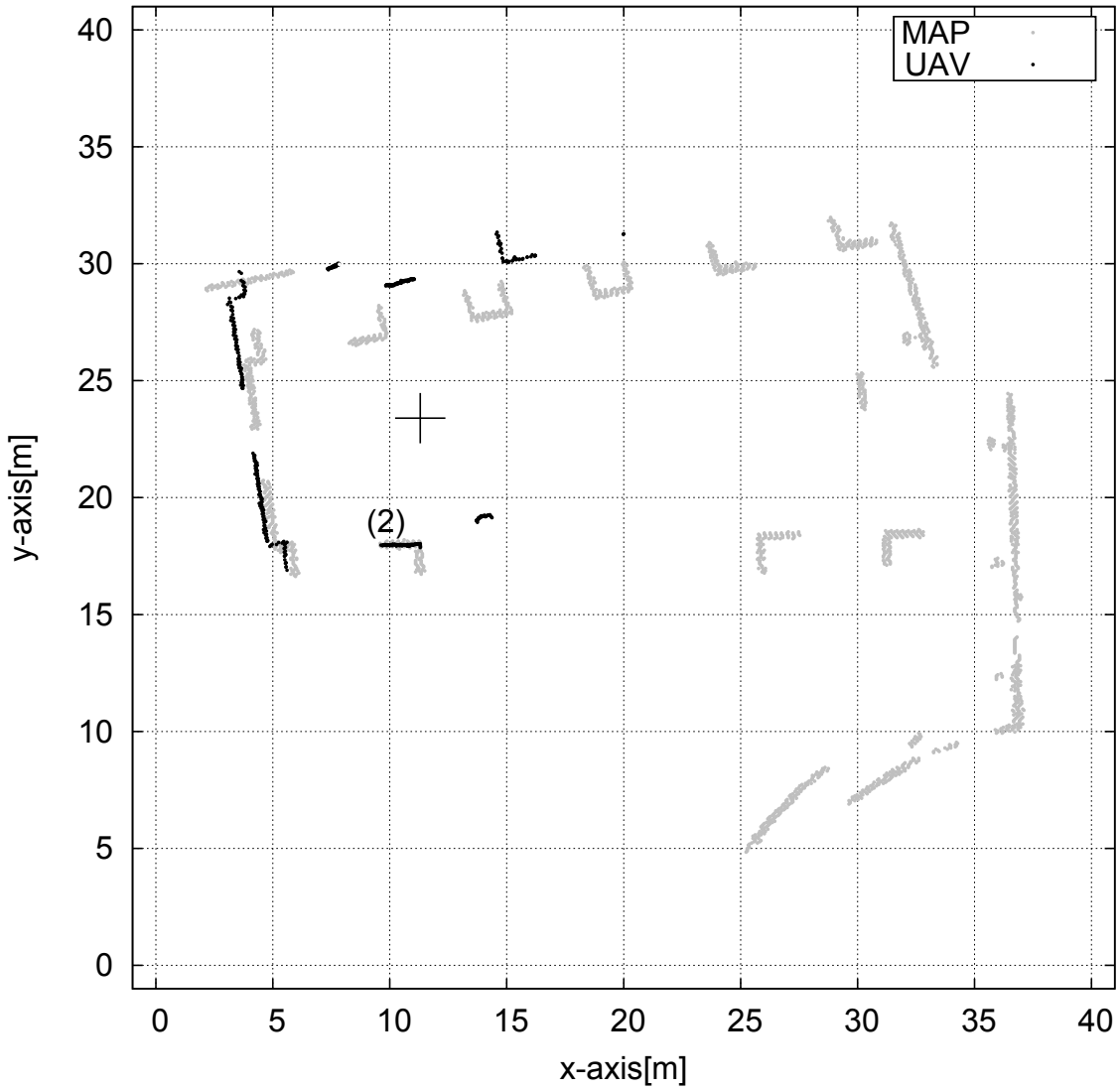


図 3.10　地図と計測データの重ね合わせ（２）

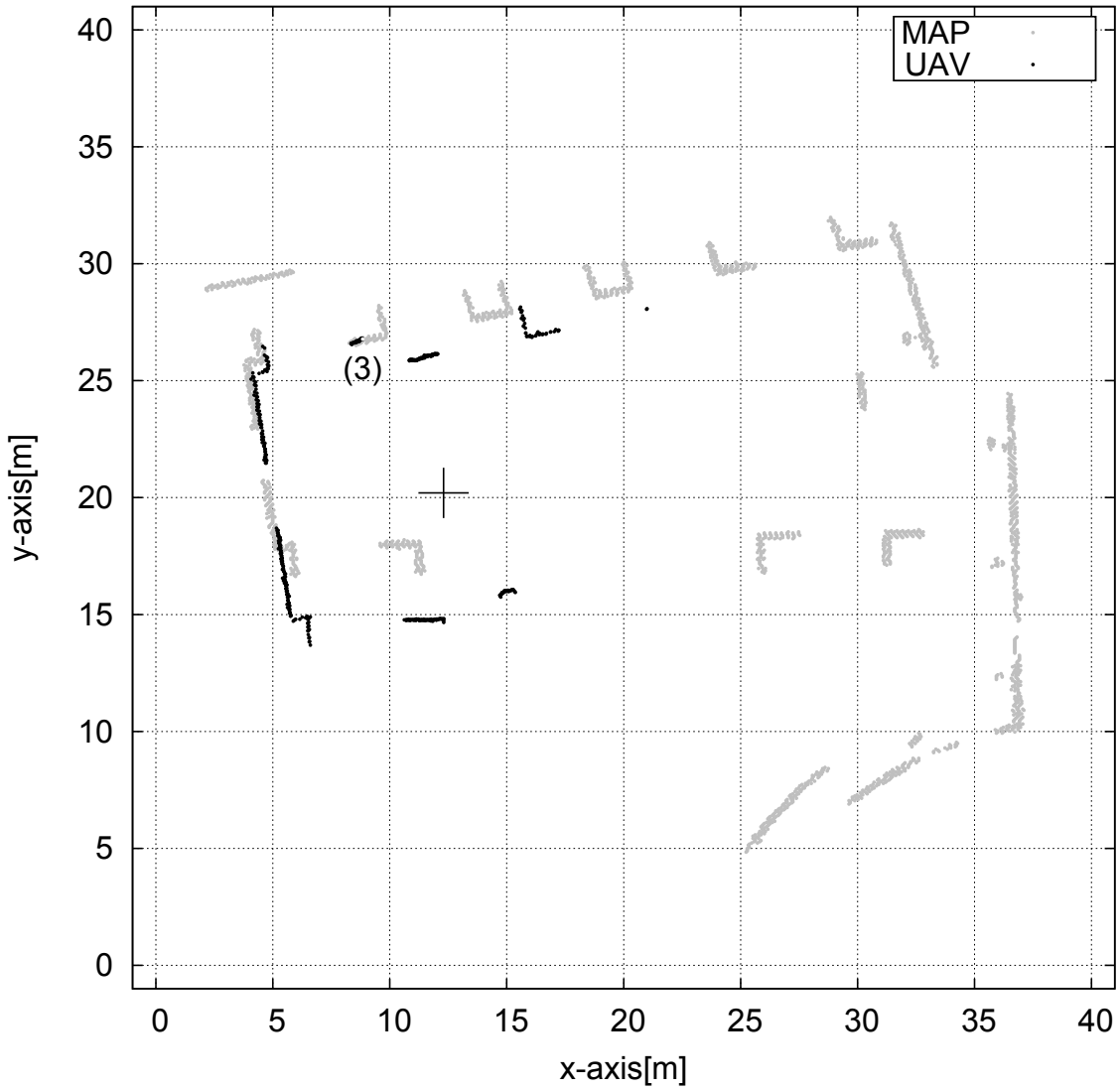


図 3.11　地図と計測データの重ね合わせ（3）

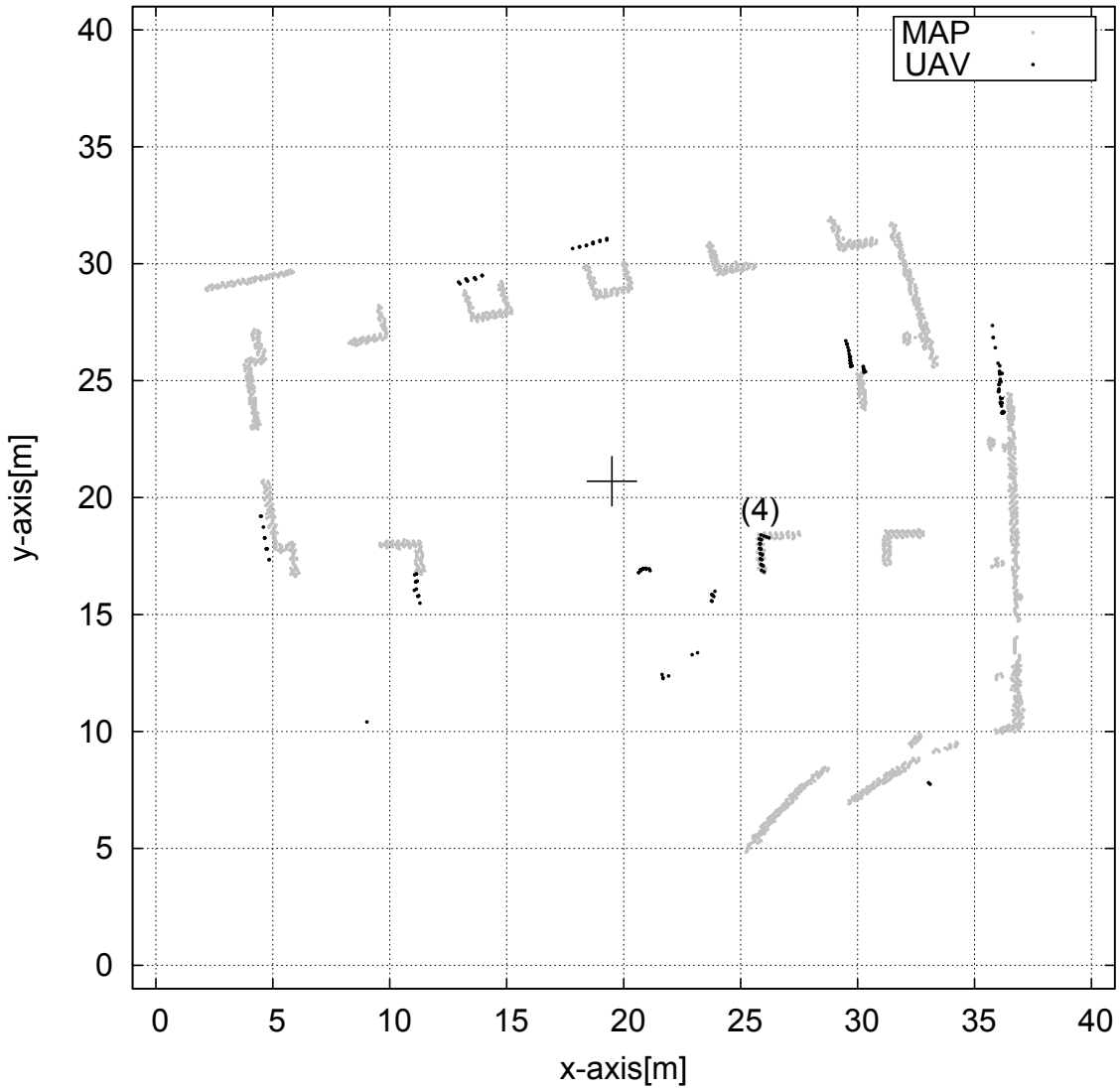


図 3.12　地図と計測データの重ね合わせ（4）

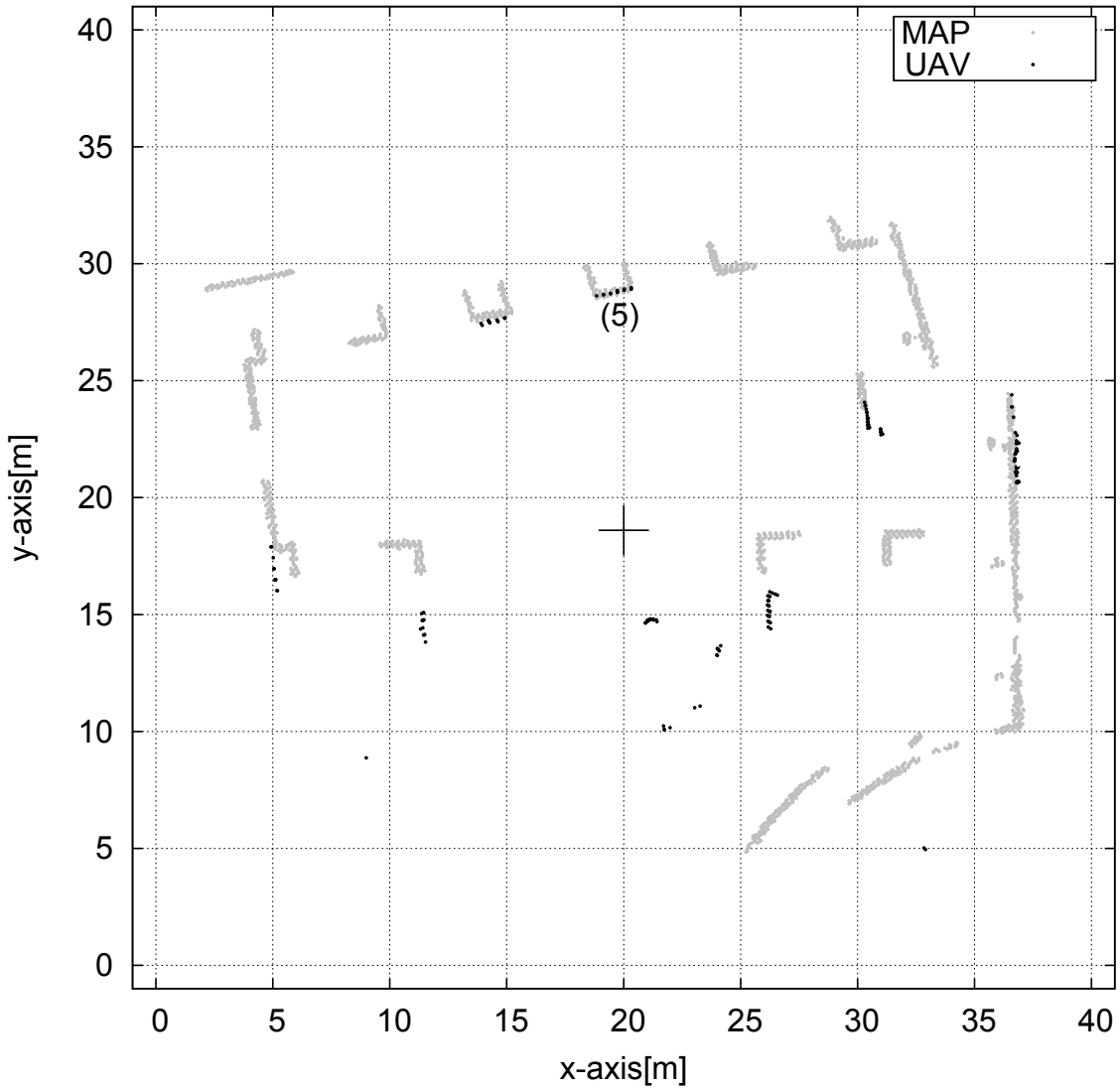


図 3.13　地図と計測データの重ね合わせ（５）

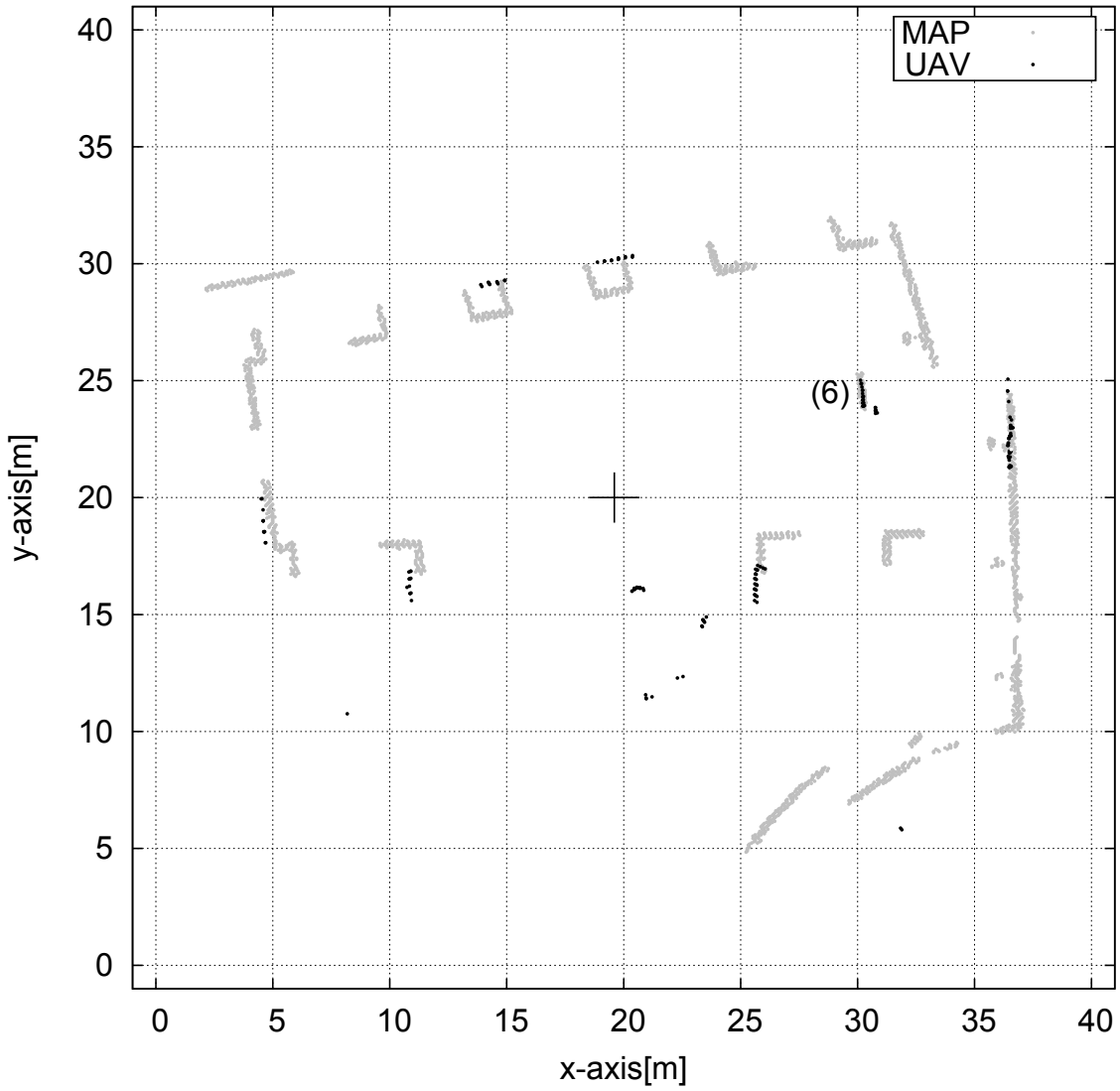


図 3.14　地図と計測データの重ね合わせ（6）

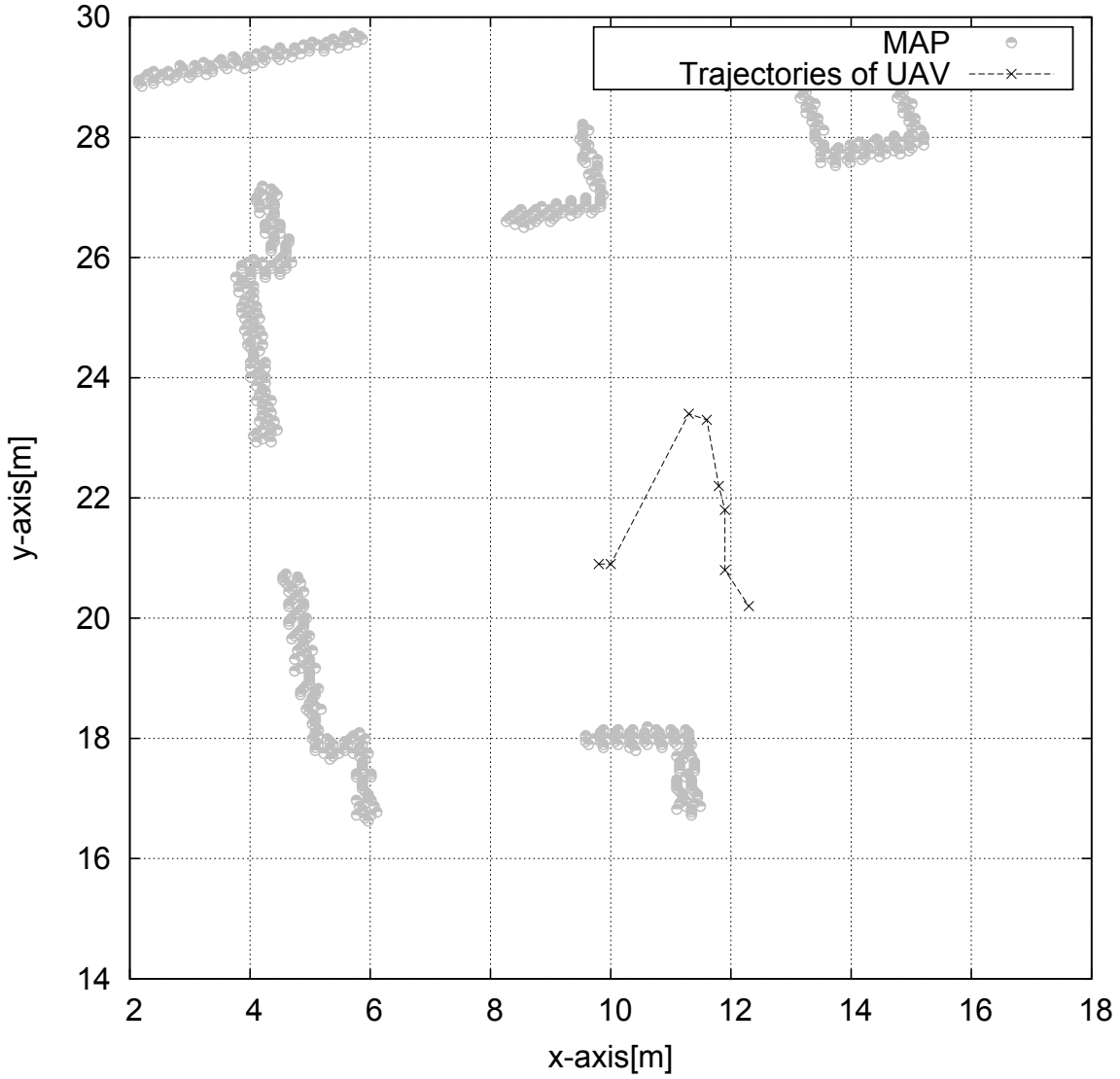


図 3.15　飛行ロボットの飛行軌跡

3.2.4　考察

実験結果における十字マークは飛行ロボットの位置の推定値とみなして良い．各図の十字マークの位置が変化しているのは，飛行ロボットが旋回中に移動していることを意味する．図 3.15 はそれらを連ねた，飛行ロボットの飛行軌跡の推定値である．本実験では，外部から飛行ロボットの位置を取得することができないため，飛行軌跡を厳密に評価する

ことはできないが，実験中に取得したビデオ映像 (サムネイル画像を付録 C に示す．) と比較すると，旋回中に飛行ロボットが移動していることが確認でき，移動軌跡とビデオ映像の動きが概ね一致しており，短い時間に取得された部位が，地図の特徴と一致する形をしていれば，特徴を活かして地図に重ね合わせを行せて，自己位置の推定が可能であるという目処が見えた．しかし，本実験で使用した一軸センサは 3ms 間隔で距離データを取得できる．このように高速でデータを計測できるセンサを必ず使用できる場合は，本実験で示したように，地図と局所データを重ねあわせすることができるが，データの取得周期がもっと大きかったり，飛行ロボットの移動速度が大きかった場合は，このような手段を用いることはできないと考えられる．次節では，逐次得られる距離データを用いて，飛行ロボットが移動したとしても可能な自己位置推定について考えていく．

3.3 自己位置推定

本研究では飛行ロボットによる自己位置推定を取り上げるが，自己位置推定とは無人機が無人機の位置と方向を推定することである．無人機は自分の位置と姿勢を知る手段を持っていないので，外界センサや内界センサの情報から位置と姿勢を推定しなければならない．自律的に行動できる無人機ならば，目的地に自分で行くことが求められることが予想されるため，自己位置推定は自律移動無人機に必ず必要な技術である．

図 3.16 に示すように，自己位置推定の方法としては環境中のランドマークと呼ばれる主要な物体の位置のデータを無人機に持たせ，無人機が主要な地物を探索，識別して，地物との位置や方向の関係から自分の位置と姿勢を推定する．この方法は船の航海において，星を観測することで自己位置を割り出す方法であるスターレコニングと同様なものである．スターレコニングは星やランドマークを見つけ出すことが必要であり，無人機にその作業をさせることも技術的な課題となる．一般的に主要な地物のデータを地図と言い，地図の構成の方法には多くの種類がある．ランドマークや周辺の状況を認識するための外界センサとしては二次元距離センサや超音波距離センサ，カメラや距離画像センサなどがある．またランドマーク自身が光や電波，音波などを発し，それらの信号を受動的に受けるセンサも外界センサに含めて考える．

また，自分の移動量や移動速度を取得する手段を持つ無人機はそれらのデータから自己位置を推定する方法もある．この手法はスターレコニングに対して，デッドレコニングとも呼ばれる．デッドレコニングはそれまでの移動量や速度の積分計算を行うため，それらの計測時の誤差も共に積分してしまうため，一般的には長い時間の移動では積分誤差が蓄積して自己位置推定の誤差が大きくなり，スターレコニングと同等の手法により誤差をリ

セットする必要がある．現在デッドレコニングに使われるセンサは，無人機の自身の情報を取得するためのセンサであるので内界センサと呼ばれている．主な内界センサは，車輪で移動するものに関しては，車輪の正逆転，回転数や回転量を計測するロータリーエンコーダや，空中を飛行する無人機や海中無人機では加速度や角速度を取得する加速度計やジャイロが使われる．

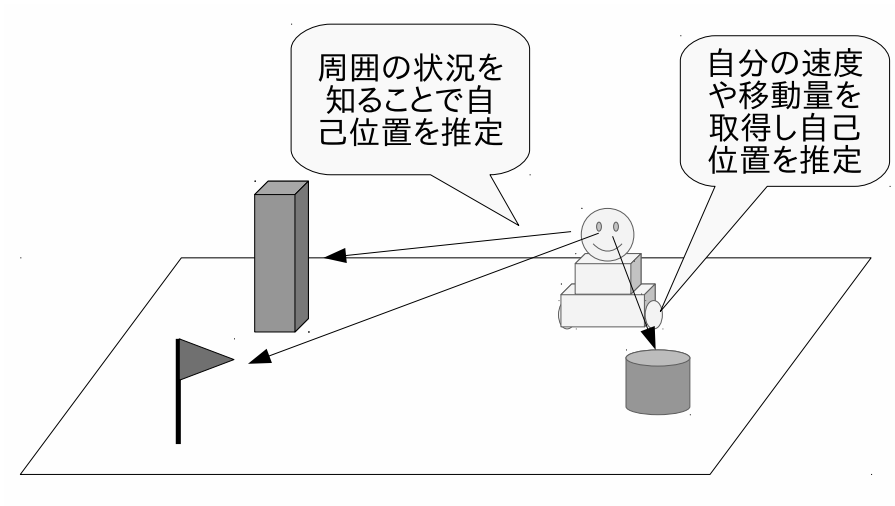


図 3.16　自己位置推定

近年，自己位置推定についても，確定的にロボットの場所を推定するのではなく，統計的手法を用いて，ある場所にロボットが存在する確率はどの程度であるのかを考えるようになってきた [18]．確率的な自己位置推定の基本をなしているのは式 (3.1) に示す，ベイズの定理である．

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{3.1}$$

ベイズの定理は事前分布 $p(x)$ と観測尤度 $p(y|x)$ によって事後分布 $p(x|y)$ を算出する．ロボットの自己位置推定問題においては，事前分布とは，事前にロボットがどのあたりにいそうなのか，場所ごとの確率と考えれば良い．もし，ロボットが何処にいるのかわからなければ，全ての場所にいる確率が等しい，一様分布を事前分布として与えればよい．

ベイズの定理に基づく自己位置推定の実装には様々なものがあるが，その一つに文献 [18] にも紹介されている拡張カルマンフィルタによるものがある．拡張カルマンフィルタを用いるためには，各分布はガウス分布を仮定しなければならない．飛行ロボットの自己位置推定を一軸距離センサで行う場合には，一回の計測では多数の場所に存在確率が分布

することが予想されるため，ガウス分布を仮定することは困難だと考える．

また，ベイズ定理に基づくものには，ランダムな粒子で分布を表現するものがある，一般的には粒子フィルタと呼ばれているが，ロボットの位置推定ではモンテカルロ位置推定（MCL）と呼ばれている．MCL は粒子を用いて分布を表現することから，様々な分布を仮定することができ，一軸距離センサ搭載の飛行ロボットの自己位置推定に有用では無いかと考えた．以降，MCL について述べる．

3.4 モンテカルロ自己位置推定

MCL では無人機の姿勢に関して正規分布のような単一の極値をもつ分布を仮定せず，無人機の姿勢の分布を多数の粒子 (Particle) で近似する．最初のステップにおいて各粒子は運動モデルに基づき遷移を行い遷移した各粒子ごとに，センサからの観測値のもっともらしさ，すなわち尤度を計算する．各粒子は尤度を重みとして，リサンプリングと呼ばれる取捨選択が行われ，尤度の高い粒子は重複して選択され，尤度の低い粒子は捨てられる．

以下，ベイズ推定を用いた移動無人機の自己位置推定とその実装である MCL について説明する．現在の時刻 t における無人機の位置及び姿勢を推定したいとする．無人機の初期状態と逐次更新されるセンサによる観測データ $\boldsymbol{Z}^t = \{\boldsymbol{z}_t, i = 1..t\}$ は与えられるものとする．飛行無人機の位置や姿勢，速度，加速度を表す状態量は，飛行無人機の自由度は 6 自由度であるため位置と姿勢の次元が 6 個，それに速度の次元を加えると 12 個となり，加速度の次元も加えると 18 次元の状態量を考えなければならないシステムである．高度制御が理想的に働き水平面内を同じ高度で飛行できると仮定し，位置と方向だけの三次元の状態 $\boldsymbol{x} = [x, y, \psi]$ として，ピッチやロール方向の揺れの計測への影響に関しては未知の外乱が加わった計測誤差とする．無人機の位置推定に観測結果を反映させた位置の事後分布を $p(\boldsymbol{x}_t|Z^t)$ とし，この分布を推定することが目的である．ベイズの手法では，状態 $\boldsymbol{x}_t$ を確定的に示す代わりに，この確率密度関数で表す．実際には，$p(\boldsymbol{x}_t|X^t)$ を用いて条件付き期待値を計算することで，位置及び姿勢の推定値を確定する．

ベイズフィルタを自己位置推定問題に適用するには次に示す予測と観測更新の 2 つのステップを繰り返す必要がある．

予測ステップでは，運動モデルを用いて無人機が運動することで考えられる予測分布 $p(\boldsymbol{x}_t|Z^{t-1})$ を求める．ここで，状態 $\boldsymbol{x}_t$ はマルコフ過程であり，事前の状態 $\boldsymbol{x}_{t-1}$ と制御 $\boldsymbol{u}_{t-1}$ のみによって決まると仮定する．運動モデルについては状態遷移確率 $p(\boldsymbol{x}_t|x_{t-1}, u_{t-1})$ で表すとする．

$$p(\boldsymbol{x}_t|Z^{t-1}) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})p(\boldsymbol{x}_{t-1}|Z^{t-1})d\boldsymbol{x}_{t-1} \tag{3.2}$$

観測更新では，事後分布 $p(\boldsymbol{x}_t|Z^t)$ を得るために，センサからの観測データを観測モデルに適用する．ここで，観測 Z^t は一つ前の観測 Z^{t-1} には依存せず，観測する際の位置と姿勢 $\boldsymbol{x}_t$ によって決まるとする．また，観測モデル $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ は観測のもっともらしさを表す尤度である．そして，$\boldsymbol{x}_t$ の事後分布はベイズの定理を用いた次式で与えられる．

$$p(\boldsymbol{x}_t|Z^t) = \frac{p(\boldsymbol{z}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t|Z^{t-1})}{p(\boldsymbol{z}_t|Z^{t-1})} \tag{3.3}$$

以上の 2 つのステップを事後確率を事前確率として，反復することで，無人機の位置や姿勢の分布を推定していく．時刻 t_0 の時を初期状態 $\boldsymbol{x}_0$ とする．初期分布については，最初無人機がどこにいるか判らない場合は，一様分布とするのが適当である．もし，初期位置が既知であるならば，既知の場所における確率が 1 になるような分布を与えても良い．

では，実際に式 (3.2) や式 (3.3) の繰り返し計算をどのように実装するかが問題である．実装方法としては運動モデルや観測モデルを正規分布で与える例としてカルマンフィルタが挙げられる．カルマンフィルタは素晴らしい成果を挙げてきた実装であり，様々な応用に用いられ種々の実装例があるが，基本的には線形なシステムにのみ適用でき非線形システムには拡張カルマンフィルタのように線形化を行って対応するなどの工夫が必要である．また，事前分布，事後分布，尤度なども正規分布でなければならないので，無人機の位置が未知で，複数の位置候補が考えられるような場合に対応するのが難しい．その他の実装方法として，本研究で取り上げるサンプリングベースの MCL がある．運動モデルに沿った多数のサンプルをランダムに生成し，観測と観測モデルにそった尤度を用いたリサンプルを行って，事後分布を決定する．

MCL では $p(\boldsymbol{x}_t|Z^t)$ を表すために関数の代わりに，多数の粒子を用いる．多数の粒子の分布が $p(\boldsymbol{x}_t|Z^t)$ に従うと考える．粒子の数が無限大になった場合に粒子の分布と $p(\boldsymbol{x}_t|Z^t)$ は一致する．粒子の数を無限大にすることは実装上は不可能であるが，十分な数の粒子によって実用上問題のない近似として成立する．MCL は粒子の分布で無人機の位置や姿勢の確率分布を近似する手法なため，どのような分布でも表すことができる．そのため，無人機の位置が未知の状態から観測を用いて自己位置を推定する場合のような大域的な自己位置推定問題のような推定開始直後は，推定位置の候補が複数の領域に存在する場合においても対応することができる．

以下に MCL アルゴリズムを以下に示す．

Algorithm 1 MCL(χ_{t-1}, u_t, z_t, m)

1: $\bar{\chi}_t = \chi_t = \emptyset$
2: **for** $m = 1$ to M **do**
3: $\quad x_t^{[m]} =$ sample_motion_model $(u_t, x_{t-1}^{[m]})$
4: $\quad w_t^{[m]} =$ measurement_model $(z_t, x_t^{[m]})$
5: $\quad \bar{\chi}_t = \bar{\chi}_t + < x_t^{[m]}, w_t^{[m]} >$
6: **end for**
7: **for** $i = 1$ to M **do**
8: $\quad$ draw i with probablity $w_t^{[i]}$
9: $\quad$ add $x_t^{[i]}$ to χ_t
10: **end for**
11: **return** χ_t

ここで，χ_t は M 個の姿勢 $x_t^{[m]}$ の集合であり，信念をパーティクルの集合で近似表現したものである．sample_motion_model は無人機の運動をサンプリングする関数であり，制御 u_t と一つ前の姿勢 x_{t-1} により計算される．measurement_model は無人機の観測 z_t と m 番目の粒子の位置及び姿勢 $x_t^{[m]}$ から観測の尤度を計算する関数である．1 行目で観測を反映させない予測ステップでの粒子の集合を表す $\bar{\chi}_t$ と観測を反映させた後の事後分布を表す粒子の集合 χ_t をリセットする．2 行目から 6 行目は粒子の数 M だけ繰り返される部分で，3 行目で m 番目の粒子の移動を無人機の制御 u_t と運動モデルに基づいて計算し粒子の位置と姿勢に反映させる部分である．4 行目はセンサの観測 z_t と無人機の位置姿勢の候補である $x_t^{[m]}$ を使って観測の尤もらしさを計算する．実際は，$x_{t-1}^{[m]}$ を用いて無人機の存在している環境ならばどのような観測結果が得られるかをシミレーションして z_t と近い結果になるものはもっともらしいとして，粒子の重み $w_t^{[m]}$ を大きくし，近い結果にならない場合は重み $w_t^{[m]}$ は小さくする部分である．重みは，後で，粒子のリサンプルをする際に使用する．5 行目で，重み付けされた位置と姿勢の情報を粒子として集合 $\bar{\chi}_t$ に加える部分となっている．7 行目から 10 行目についてはリサンプルの部分で，各パーティクルを重み $w_t^{[m]}$ が大きいものは増やし，小さいものは削除する処理を行う．

3.5 移動モデル

MCL を用いるためには無人機の移動モデルを作らなければならない．本研究ではクアッドロータヘリコプタの運動を高度制御により同一平面内に限定できると仮定する．クアッドロータヘリコプタは自身の運動を慣性計測装置によって取得することができる．慣性計測装置は，加速度と角速度を取得できるセンサであるので，それらを用いた運動モデルを構築する．加速度や角速度は制御周期の間は変化せず等加速度運動又は等角速度運動に近似できると仮定すると，運動モデルは以下のように表すことを提案する．

$$x_{k+1} = 0.5a_x h^2 + v_{x_k} h + x_k \tag{3.4}$$

$$y_{k+1} = 0.5a_y h^2 + v_{y_k} h + y_k \tag{3.5}$$

$$v_{k+1} = a_x h + v_{x_k} \tag{3.6}$$

$$v_{k+1} = a_y h + v_{y_k} \tag{3.7}$$

$$\psi_{k+1} = ph + \psi_k \tag{3.8}$$

ここで，x_k,y_k は位置,v_{x_k},v_{y_k} は速度,a_x,a_y は加速度,ψ_k は方向,p は角速度,h は制御間隔を表す．

本研究では，機体は平面内を運動するとしたが，実際は慣性座標系と機体座標系の座標変換をして慣性演算と呼ばれる計算をしなければならない．慣性演算の詳細については付録 B に示した．

3.6 観測モデル

一軸レーザ距離センサの観測モデルについては，文献 [18] に従い，小さな計測雑音，想定外の物体による誤差，物体検出失敗，原因不明のランダム雑音を考慮した．

小さな測定雑音については計測値を z_t，真の距離を z_t^* とすると，計測確率は

$$p_{hit}(z_t^k|x_t, m) = \begin{cases} \eta\mathcal{N}(z_t; z_t^*, \sigma_{hit}^2) & \text{if } 0 \leq z_t \leq z_{max} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

ここで η は確率の積分を 1 にするための正規化パラメータである．

想定外の物体については，z_t^* よりも必ず小さな値として検知される．これは指数分布でモデル化でき次式で与えられる．

$$p_{short}(z_t^k|x_t, m) = \begin{cases} \eta\lambda e^{-\lambda_{short} z_t} & \text{if } 0 \leq z_t \leq z_t^* \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

ここで η は確率の積分を 1 にするための正規化パラメータである．

計測の失敗については，レーザ距離計においては光を吸収する物体においてそのような事態が発生する．センサーは距離が検知できなかた場合は最大距離を返すので，最大距離の確率について

$$p_{max}(z_t|x_t, m) = \begin{cases} 1 & \text{if } z = z_{max} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

でたらめな計測については，ときおり理解できない出力を返すことがあり，計測区間全域に渡る，一様分布で表すこととする

$$p_{rand}(z_t|x_t, m) = \begin{cases} \frac{1}{z_{max}} & \text{if } 0 \leq z_t \leq z_t^* \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

Algorithm 2 に以上をまとめた計測の尤度を求めるアルゴリズム示す

Algorithm 2 beam_range_finder_model(z_t, x_t, m)

1: compute z_t^* for the measurement z_t using ray casting
2: $p = z_{hit}p_{hit}(z_t|x_t, m) + z_{short}p_{short}(z_t|x_t, m)$
3: $+z_{max}p_{max}(z_t|x_t, m) + z_{rand}p_{rand}(z_t|x_t, m)$
4: **return** p

3.7 シミュレーション

飛行実験に先立ち，一軸距離センサを使用したモンテカルロ位置推定が有効に機能するかシミュレーションを行い確認した．シミュレーションプログラムは実機の運動については平面運動に限り，飛行ロボットの位置 x，y，とヨー角 ψ の状態量を加速度 a_x，a_y と角速度 r を与えて制御する．前提として制御間隔 h の間は制御は変化しないとした．実機においても制御間隔の間は制御は変化しない．センサの計測距離は 30m とした．飛行ロボットには一軸レーザ距離計が正面を計測するように取り付けられている．粒子の数は 10000 個で行った．粒子の初期分布は実験環境に一様に分布するように初期分布をとったが，方向に関しては初期情報が既知で使用できるものとして，すべての粒子の初期値が右向きすなわち 0 度と設定した．回転の角速度は 4π[rad/s] とした．シミュレーションプログラムについては付録 D に示した．

3.7.1　シミュレーションの目的

シミュレーションの目的とした確認事項を以下に示す．

- 一軸距離センサの場合，初期位置が未知で初期の事前分布を一様分布とした場合に，機体が回転せずに一定方向を見続けた場合は，得られた計測データを返す候補地は複数あるはずであるので，複数の領域にパーティクルが集中する結果となるか
- 回転を始めた場合に，パーティクルの分布が飛行ロボットが存在する場所周辺に集まるか
- 回転をやめて移動した場合に，パーティクルの集合は計測していない方向に広がっていくと考えられるが，実際にはどうなるのか

以上の 3 点をシミュレーションで確認した．

3.7.2　シミュレーションの結果

図 3.17 に計算開始時点の状態を示した．図において黒い点群は環境に存在する壁や柱，バツ印の点群は粒子，矢印が無人機の位置と方向を表している．初期状態では 10000 個の粒子が一様に分布している．

図 3.18 は 1 回めのリサンプルの後の状態を表し, 飛行ロボットは右方向 20m ほど離れている柱の方向を向いており緑色の粒子は右にある柱や壁から約 20m 離れた位置に分布していることが確認できる．

図 3.19 は回転を開始して 0.04 秒後のリサンプルの後である．回転にって今まで見ていなかった壁や柱を認識し，飛行ロボットが自己位置推定を結果が絞られてきたことがわかる．

図 3.20 は回転を開始して 0.12 秒後のリサンプルの後である．約 1/4 周飛行ロボットが回転した後であるが，ほとんどの粒子が飛行ロボットの位置に集まっているのが確認できる．

図 3.21 は回転を開始して 2.5 秒後のリサンプルの後である．2 秒の時点で回転が止まっており，そこからグラフで右方向に向けて移動を開始している．粒子が計測できない方向へ拡散していくのが確認できる．

シミュレーションの結果は一軸レーザ距離センサの予想された特性を表しており，アルゴリズムが有効であると考える．

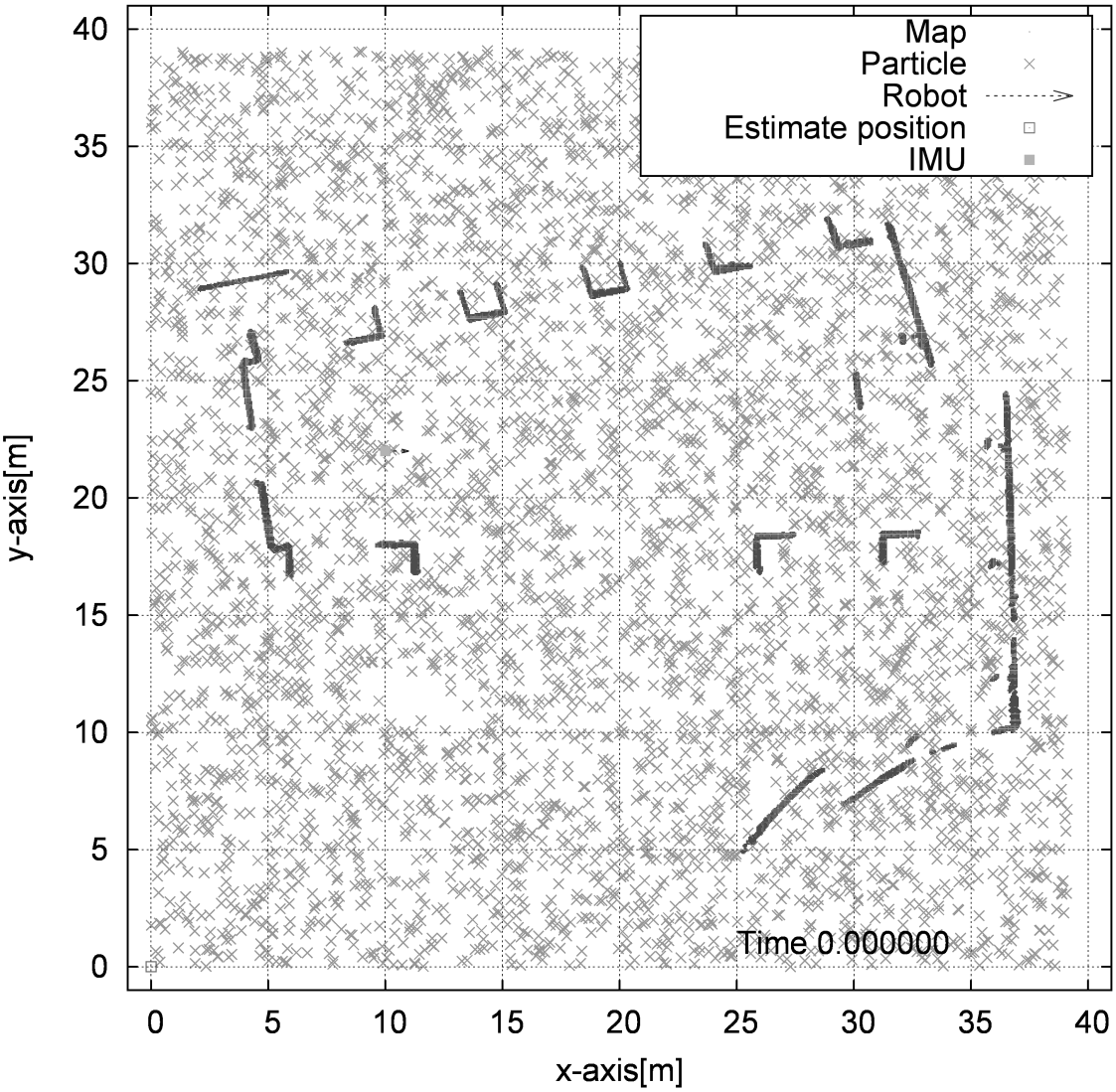

Map
Particle
Robot
Estimate position
IMU
Time 0.000000
y-axis[m]
x-axis[m]
0
5
10
15
20
25
30
35
40

図 3.17　初期分布

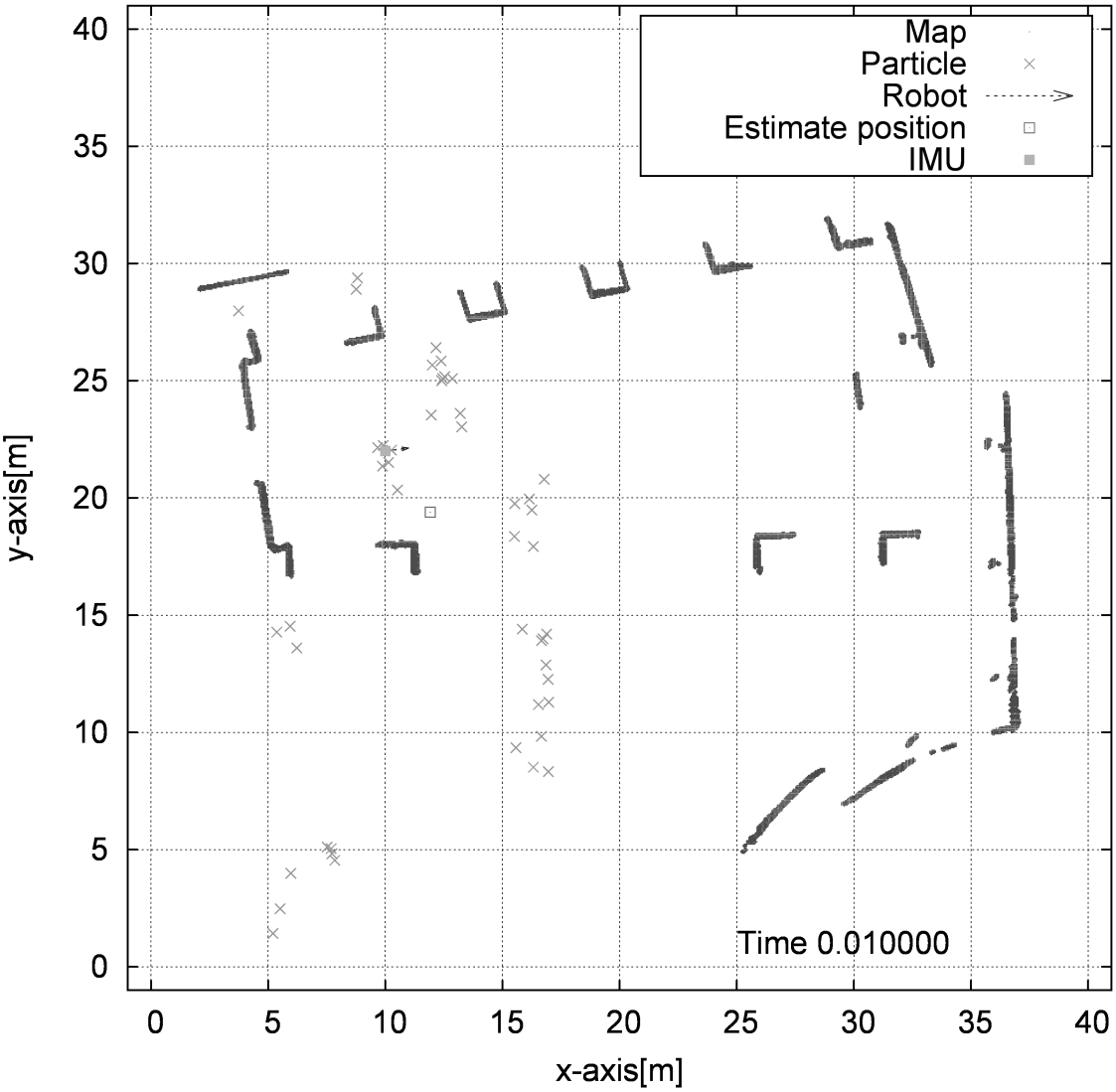


図 3.18　1 回目のリサンプル後

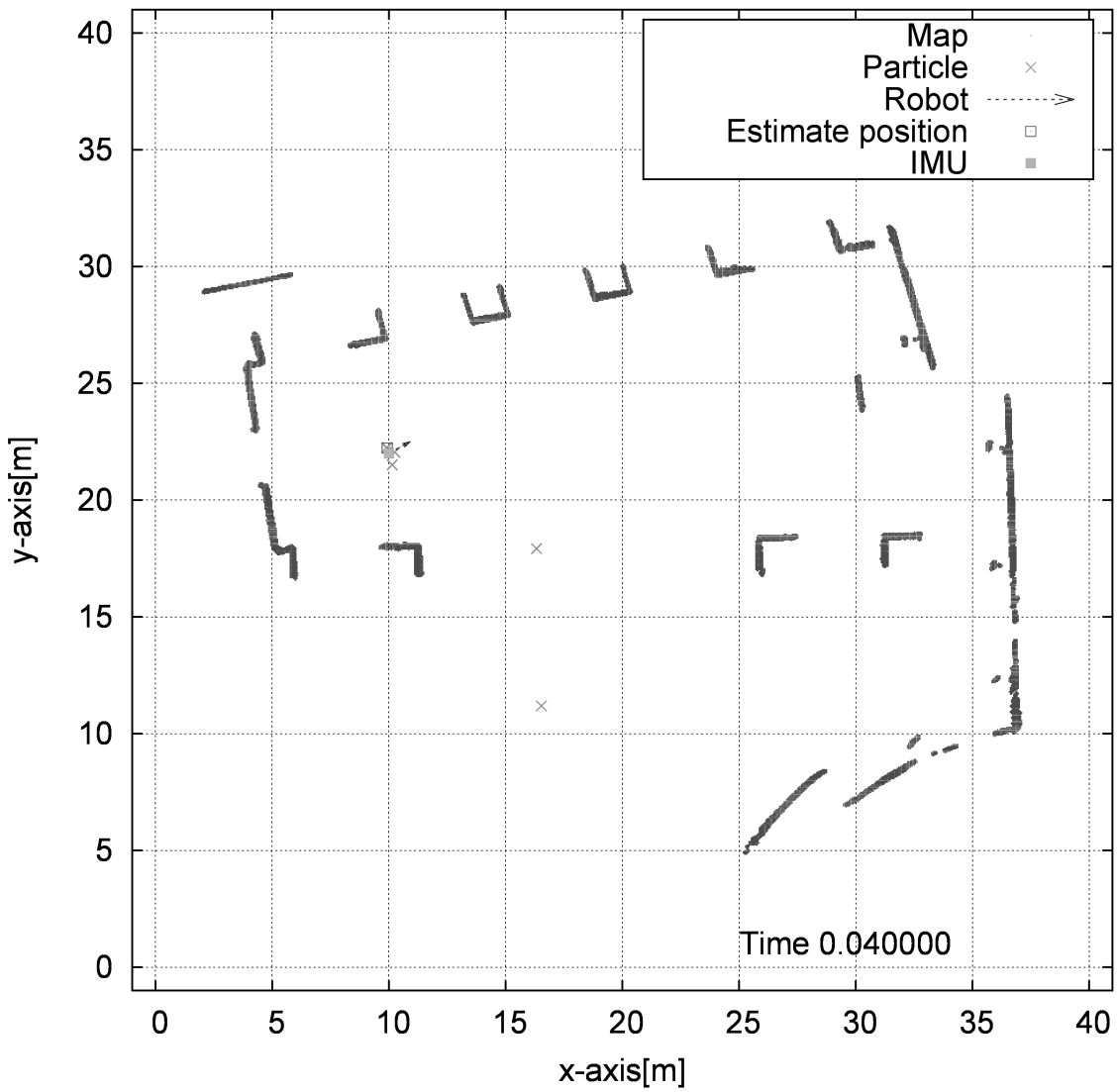


図 3.19　0.04 秒後のリサンプル後

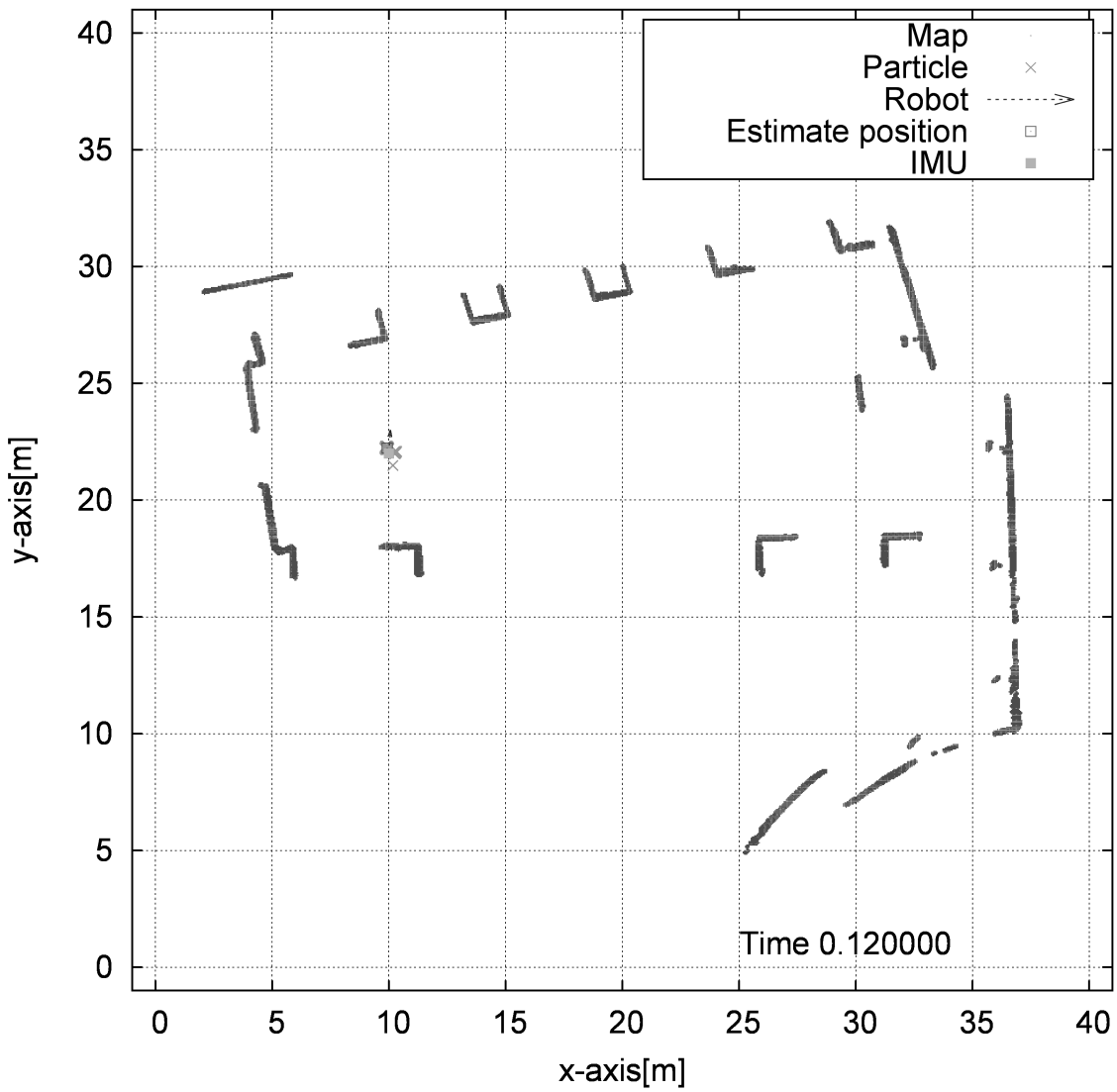


図 3.20　0.12 秒後のリサンプル後

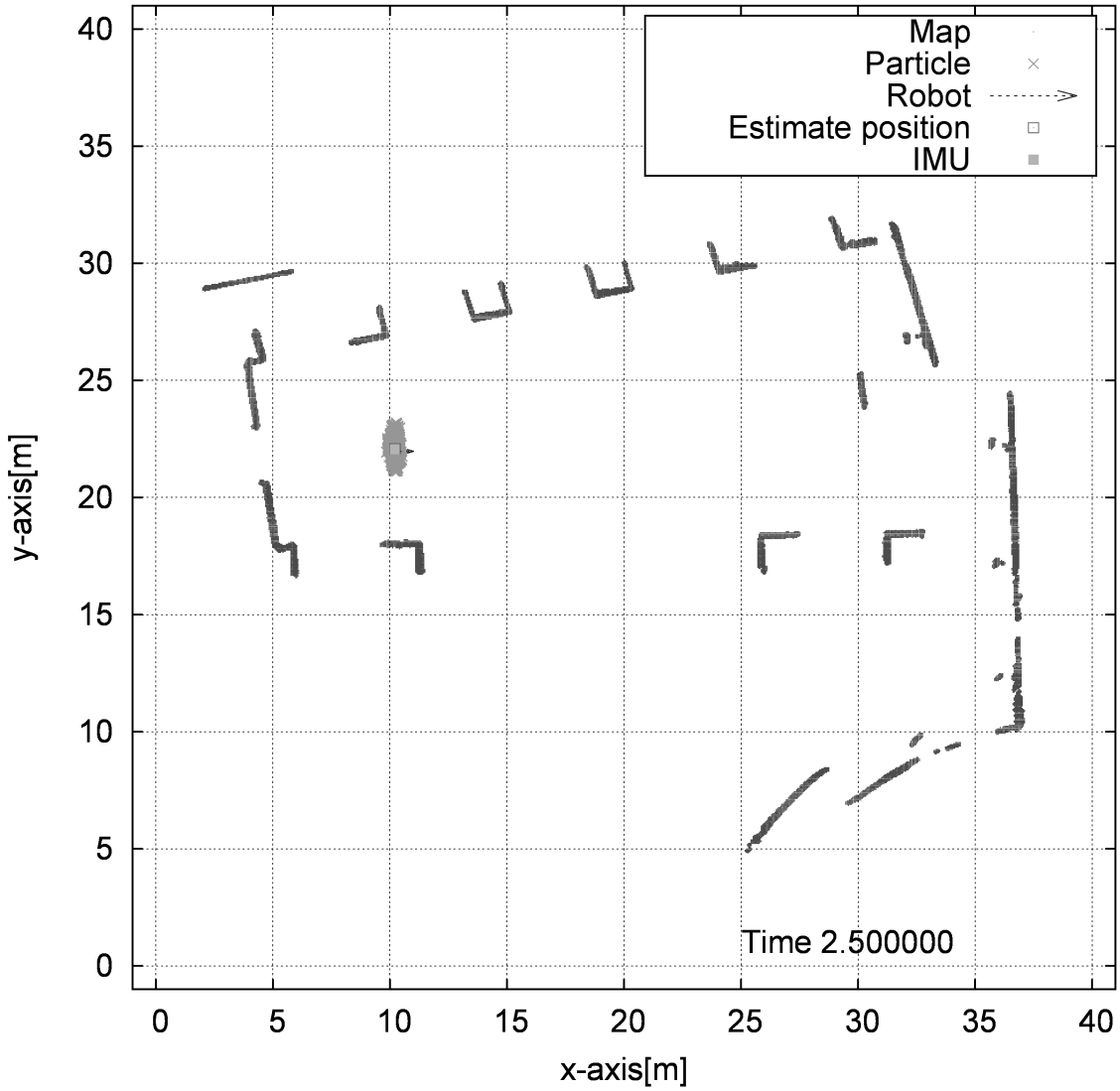


図 3.21　2.5 秒後のリサンプル後

3.8　飛行ロボットによる自己位置推定実験

自己位置推定は飛行ロボットに搭載しているコンピュータで実時間で実現できなければならないが，MCL のパーティクルフィルタの演算は計算量が大きいため，オンボードでは実現できなかった．そこで，飛行実験で取得した，距離データと慣性データをデスクトップコンピュータで処理をして，位置推定を行った．

3.8.1 実験装置

実験装置については，3.2.1 項と同じものを使用する．

3.8.2 実験方法

実験の手順を以下に示す．

1. 予め決めておいた場所に飛行ロボットを設置し，電源を入れる．
2. Beagleboard-xM の無線 LAN が無線ルータと接続したことを確認する．
3. 操作用端末としてノート PC を立ち上げ，端末プログラムを用いて，Beageboard-xM に SSH を用いて接続する．
4. roscore を実行する．
5. IMU のデータを取得する ROS のノードを実行する．
6. 一軸距離センサーのデータを取得する ROS のノードを実行する．
7. ROS のコマンド rosbag により，IMU の出力と一軸距離センサの出力をロギングする．
8. 離陸しホバリング状態にする．
9. ホバリングが安定したことを確認できたら，ヨー回転指令を与えて，機体を時計回りに回転させる．
10. 最低 1 周したら，回転を停止する．
11. データ取得が終了したならば, rosbag を停止しする．rosbag はデータをバッファーから，SD カードに記憶するのに時間がかかるため，記録終了を確実に確認して次の手順に進む．
12. 他の ROS のノードもすべて停止する．
13. roscore を停止する．
14. 電源を切る

3.8.3 実験結果

シミュレーションによって，アルゴリズムの有効性が確認できたので，続いて実データを用いた実験を行った．図 3.22 に示す，シミュレーションと同じ地図を持つ環境で飛行試験を行った．飛行で得られたデータにモンテカルロ位置推定処理をかけたところ，初期

位置付近に推定結果を得た．推定プログラムについては付録 E に示した．

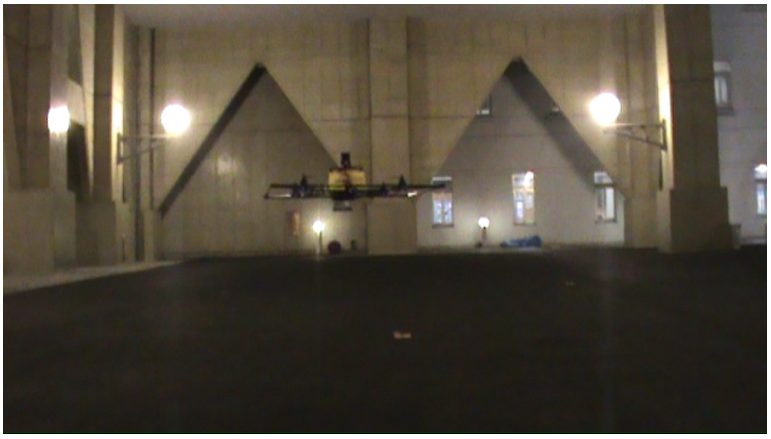


図 3.22　飛行試験の様子

図 3.23 に示すように，飛行実験のデータでは初期分布を実際の機体の付近に限定して計算を行った．これは，実際の機体にモデルのパラメータを合わせ込むことをしていないため，自己位置推定が誤った推定結果を出してしまい，粒子が誤った一定の領域に集まってしまうことが起こるため，そのような領域を除外するために，絞り込みを行った．

図 3.24 に示すように，0.02 秒後には，粒子の分布が 2 箇所に絞られた．図 3.25 に示すように，2 秒後から 2.5 秒にかけて旋回を行う．2.5 秒時点では，粒子の分布が 1 箇所に絞られた．

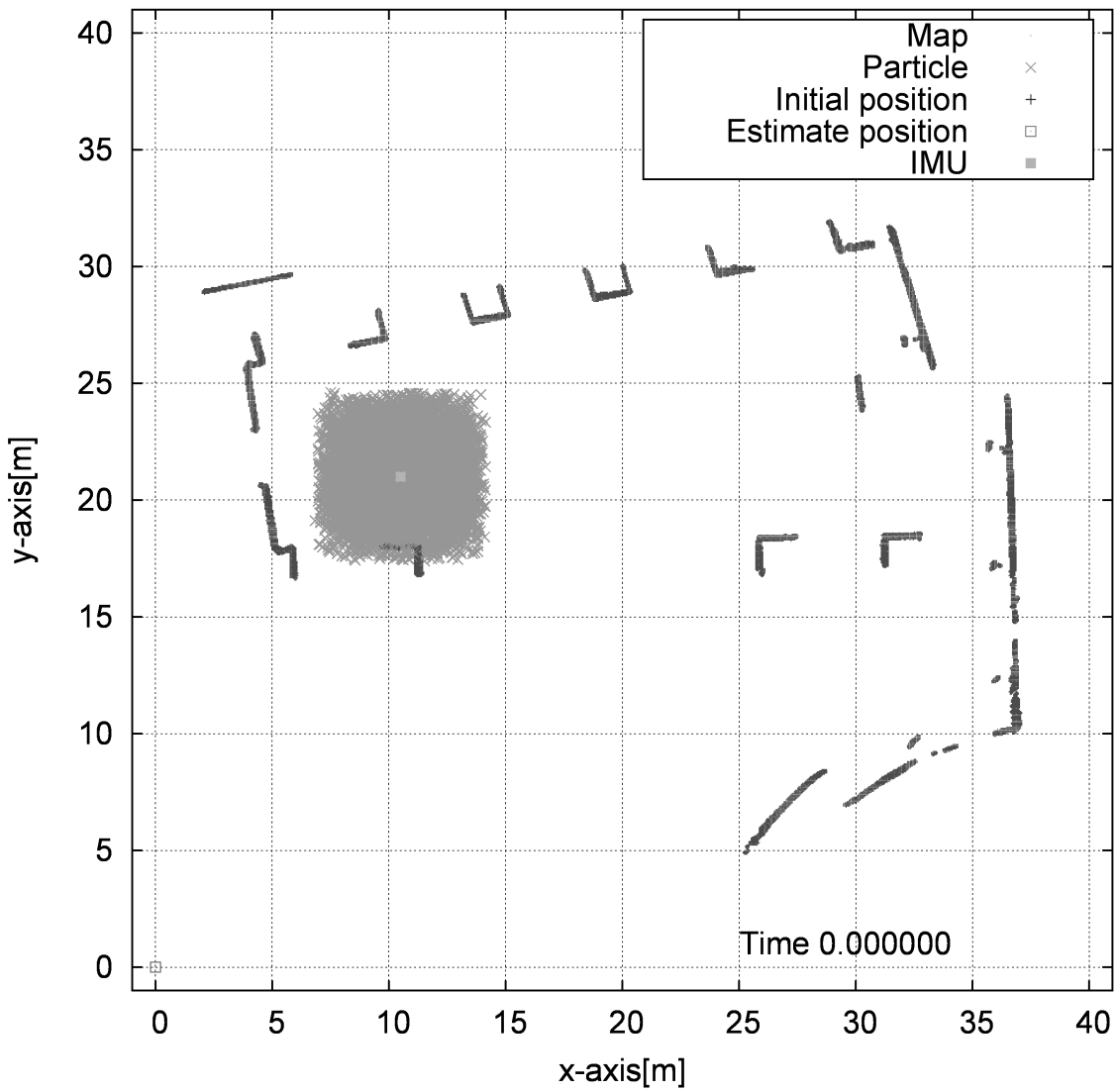

Map
Particle
Initial position
Estimate position
IMU
Time 0.000000
y-axis[m]
x-axis[m]
0
5
10
15
20
25
30
35
40

図 3.23　初期状態

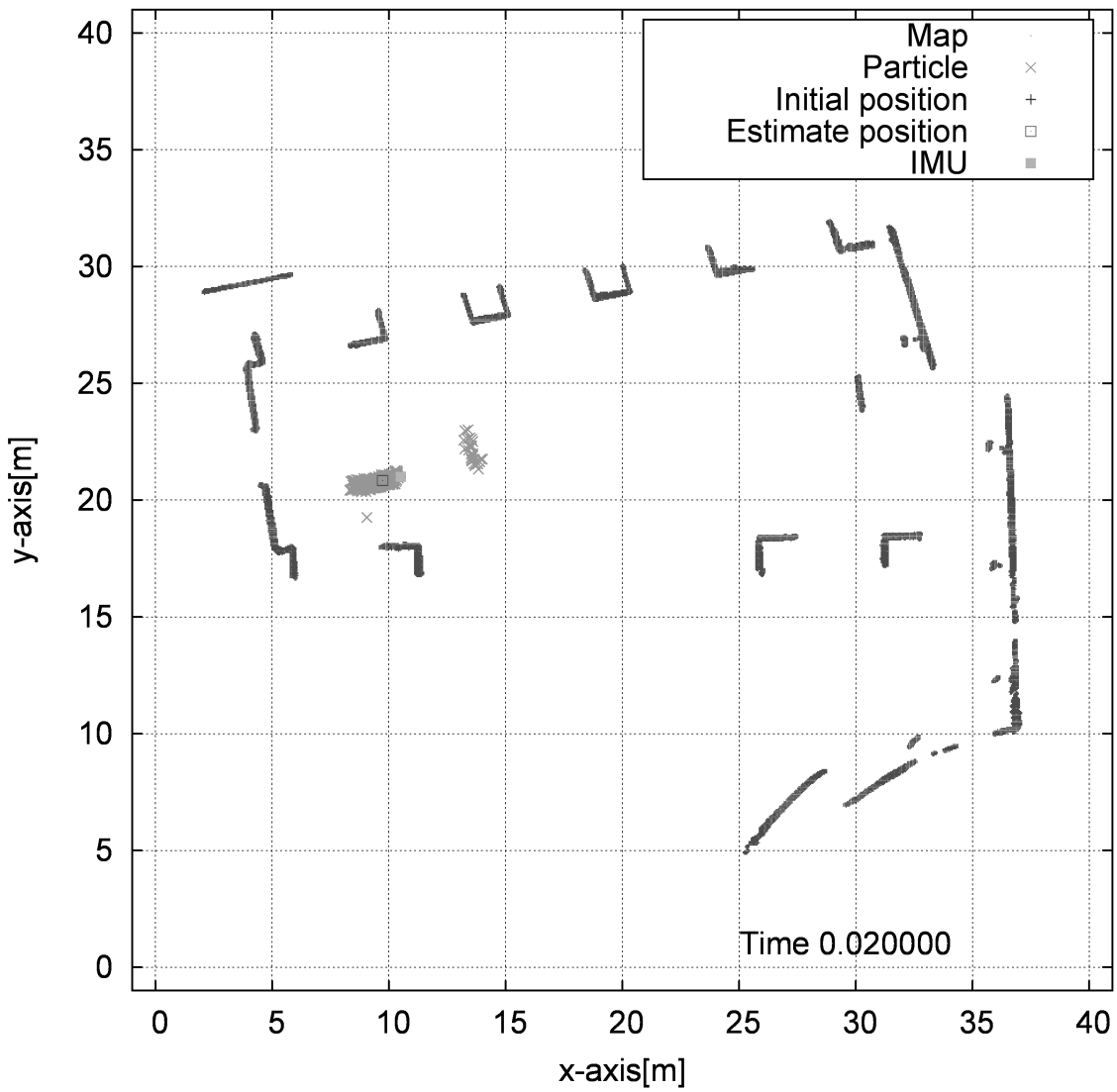


図 3.24　0.02 秒後のリサンプル後

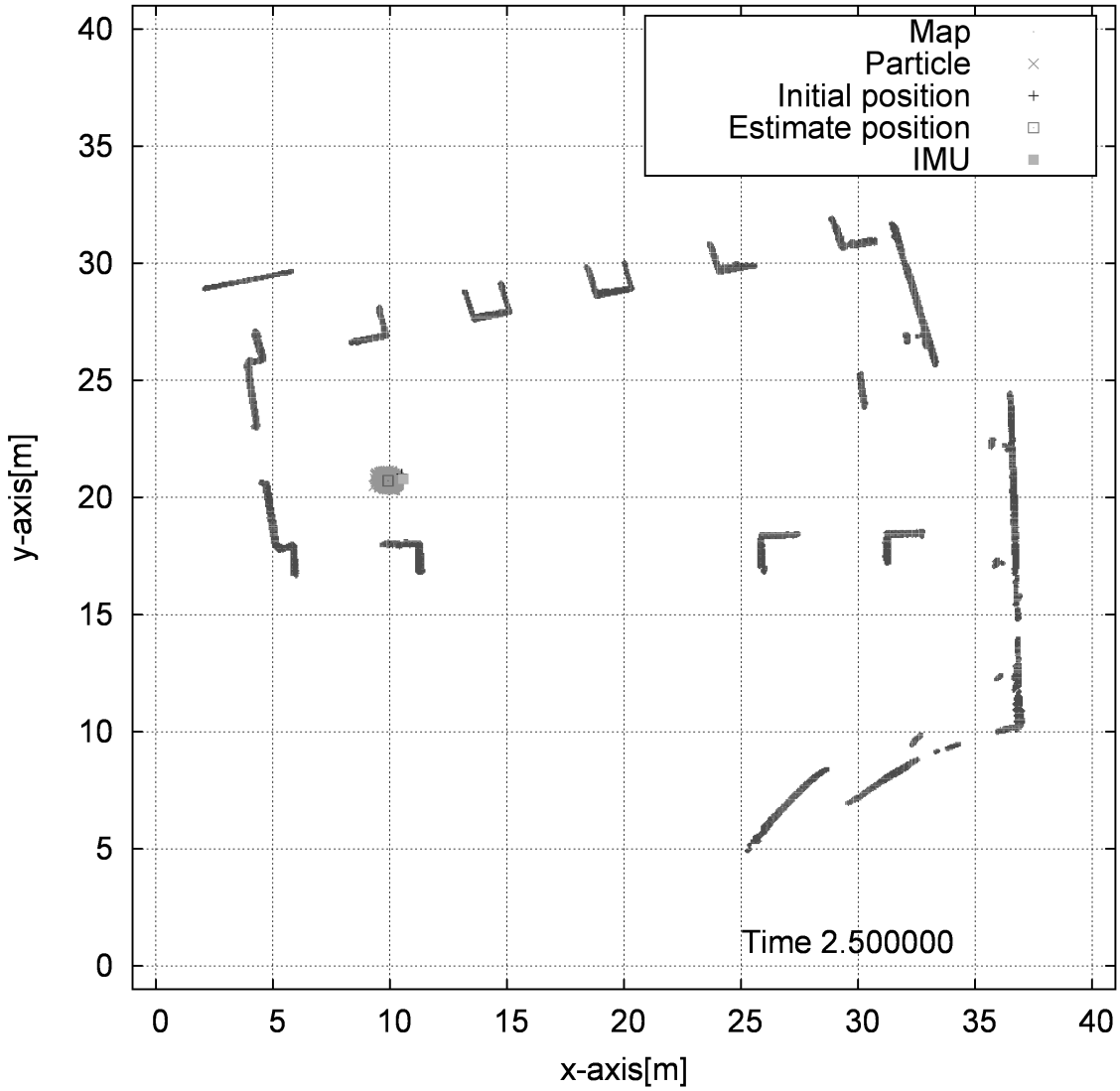


図 3.25　2.5 秒後のリサンプル後

3.8.4　考察

一軸レーザ距離センサデータを用いて，モンテカルロ位置推定を適用することに成功した．一軸距離センサを用いた自己位置推定は，同じ場所において回転することによて二次元距離センサの計測と同等のデータが得られれば容易に達成可能と考えた．しかし，実際の飛行では，飛行ロボットの回転や移動，外乱の影響での動揺があり，それらは一軸レー

ザ距離センサの出力データが二次元距離センサのような理想的な周辺状況の取得にはならないことを意味しているので，できるだけ低減したいと考えていた．しかし，一軸距離センサは機体が動かなければ，自己位置を絞り込むことはできないため，そのような微小変動が，一軸距離センサにおいては，多様な観測結果をもたらして，自己位置推定に寄与していることが判った．ときおり，間違った推定結果に陥ることも見られたが，尤度などを監視し，ある一定値以下に尤度が達した場合に，推定を失敗したとして，粒子を再び分布し直すことで，推定を再度開始することもできる．また，モデルの最適化を本研究では試みていない，推定失敗の確率が高くなっていることの原因として考えられるため，モデル最適化の処置も今後は試みたい．

3.9　あとがき

2 章で述べたように，一軸距離センサの軽量化の実現の目処が立ったため，飛行ロボットに搭載して，自己位置推定を達成するための検討を進めてきた．一軸距離センサでは，一度の計測のみでは一方向の距離が取得出来るだけであるので，情報が足りず，自己位置を限定することができない．しかし，二次元距離センサが，行っていることを，飛行ロボットがセンサと共に回転することによって，達成できるのではないかと考え，検証作業を行った．理想的には，同じ場所を回転できれば，二次元距離センサと同様に自己位置推定可能だが，実際の飛行ロボットは飛行中，アンバランスなどの内的要因や，風や地面効果，壁面効果などの外乱によって，移動したり，揺れている．

移動や，揺れの影響が自己位置推定を不可能にするのか，実際に飛行試験を実施して確認した．結果的には，位置が大幅に変化しない短時間に地形の特徴が顕著に現れる部分の測定ができれば，地図との重ね合わせで，かなり良い結果を得られる．

また，顕著な特徴を見出すことができなくても，毎回の計測データによってベイズ更新を行う MCL によって，自己位置推定が可能であることも確認した．

現状においては，最も基本的な MCL の実装に留めているため，自己位置を見失う場合に対する対処ができていない．これに対しては，尤度を監視し粒子をリセットするなどの手法を加えて，自己位置推定の確度を上げたい．また，計算コストが大きいため，実時間で計算が終わらない問題もある．これに対しては，粒子の数を状況に適合させて動的に調整するなど最低限の粒子数で計算が行われるように工夫することで改善が見込まれる．

第 4 章

結言

4.1　本研究のまとめと成果

小型の飛行ロボットを開発するためには，搭載する機器の小型・軽量化を考えていかなければならない．

飛行ロボットに必要不可欠なセンサとしてレーザ距離計がある．現在，多くの飛行ロボットに搭載されている二次元距離センサは約 130gf〜300gf の重量を持つ．これは，小型の飛行ロボット開発の足枷となりかねない重量である．そこで，小型軽量な距離センサを短時間に開発するために，マルチエコー測定機能を持つ二次元距離センサを改造し，一軸距離センサにする手法について提案した．

本研究において改造に用いた二次元距離センサは，スキャンの際に内部の基準板を測定し，帰ってきた測定値から回路の発熱による変動を補正する．しかし，回転機構を持たない一軸距離センサの場合，測定時に基準板の計測ができないため，補正のための代替手段を必要とした．本研究では，センサのレーザビームを完全に遮蔽しないように計測方向に基準板を設置し，マルチエコー測定機能を用い，第 1 エコーが距離既知の位置にある基準板からの反射光とし，第 2 エコーが被測定物からの反射光となるようにすることで，基準板からの計測結果と被計測物からの計測結果が得られるように工夫した．第 1 エコー，第 2 エコーの両方に同じ熱の影響による変動が含まれていることとなるため，互いの差を求めることにより，変動を除去できる．

しかし，測定結果には受光強度の影響と考えられる誤差が残っていたため，実験結果から受光強度と誤差に関する回帰直線を求めて実験式を導き出すことにより，その除去を行った．また，実際に運用する場面を考慮して，回転するターンテーブルに載せ動力を用いた連続的な回転による計測を行い，周辺状況の測定装置としての有効性を確認した．

また，小型の飛行ロボットに一軸距離センサを搭載しホバリング状態で旋回して周辺の状況を確認する試験を実施し，周辺状況を取得可能であることを確認すると共に，事前に用意した地図と，飛行時に取得された地図の特徴のある領域を重ねあわせることで，飛行ロボットの飛行軌跡を見出すことに成功した．しかし，距離センサのデータ取得周期や飛行ロボットの移動速度によっては，必ずしも同様な方法で飛行ロボットの位置を見出すことは難しいと考えられるため，他の方法が必要であった．

そこで，移動や旋回やホバリング中に逐次得られる計測データから，ロボットの存在確率を求めるためにモンテカルロ位置推定を用いたところ，ホバリング時の旋回では自己位置を見出すことに成功した．移動時については，シミュレーション結果において，センサの存在しない方向への不確かさが大きくなることが確認できた．

以上，本研究の成果をまとめると

1. 一軸距離センサの実現を困難にしていた熱による変動を低減する手法を提案し，実験により有効性を確認した．
2. 一軸距離センサと慣性演算の計算結果を逐次使用しながら，飛行ロボットの自己位置を推定する方法について提案し，ホバリング時の旋回時の自己位置推定に成功した．

本研究によって，二次元距離センサの技術を用いることで軽量な一軸距離センサを実現できると考える．そして，これまでセンサの重量によって制限されていた飛行ロボットの小型化への可能性を高めるとともに，二次元距離センサを搭載した飛行ロボットと同等の自己位置推定能力を持ち，自律飛行可能な従来より更に小型の飛行ロボットの実現が可能となる．

飛行ロボットは今後ますます人々の暮らしの中に活用されていくと思われるが，そのためには人間が活動するどんな場所にも入って行けることと，人間が立ち入ることが困難な場所にも容易に到達できることが求められる．小型・軽量化はそのために必要な条件であり，本研究がそのための一助となると考えている．

4.2　今後の課題

本研究では，従来の二次元距離センサの回転機構を固定して一軸距離センサとしたため，実際には軽量化を行っていない．実験に用いたクアッドロータも全長・全幅が 0.98m とやや大きい機体である．

本研究においてマルチエコーを用いることで熱による誤差が軽減できることが示された

ので，軽量な一軸距離センサをこの方式で実際に制作し，全長・全幅 30cm のクアッドコプターに搭載し実験を行っていきたい．

また，一軸レーザ距離計を用いた自己位置推定についても，飛行ロボット自体の機上の計算機で計算が終了させる必要がある．また，観測モデルの最適化などの課題が残っている．

更に，今回はその場で回転することに実験はとどめたが，簡単な壁に沿っての移動から自律移動プログラムと本方式を組み合わせて，自律移動の課題にも取り組んでいきたい．

謝辞

本論文は，筑波大学システム情報工学研究科コンピュータサイエンス専攻において博士 (工学)　大矢晃久　筑波大学システム情報系教授のご指導のもと，著者が同専攻博士課程学生として在籍した間に行った研究成果をまとめたものです．同教授には，本研究を進めるにあたり，大変ご多忙であるにも関わらず，始終変わらぬご指導，ご鞭撻をたまわりました．ここに謹んで厚く感謝を申し上げます．

また，工学博士　油田信一芝浦工業大学工学部電気工学科特任教授には本研究を始める機会を与えていただくと共に，有益なご指導とご助言を賜りましたことに深謝いたします．

また，研究を進める上で工学博士　坪内孝司　筑波大学システム情報系教授にはご助言とご指摘を頂きお礼申し上げます．

また，本研究を学位論文としてまとめるに際しましては，博士 (工学)　河辺 徹　筑波大学システム情報系教授，博士 (工学)　亀山啓輔システム情報系教授には本論文をご検討いただき，貴重なご指導，ご助言を賜りました，厚くお礼申し上げます．

また，工学博士　山田弘文　金沢工業高等専門学校前校長には，筆者に学位取得を勧めていただくと共に，その後も研究に対して常にご助言，ご指導，激励を頂きましたこと御礼申し上げます．

また，在学間，防衛大学校本科及び研究科の恩師である工学博士　金井喜美夫　防衛大学校名誉教授，工学博士　越智徳昌　防衛大学校航空宇宙工学科教授，工学博士　滝田好宏　防衛大学校情報工学科教授には始終変わらぬ激励を頂きました，感謝しております．

また，研究を遂行する上で，必要不可欠であった一軸測域センサを提供いただくと共に，測域センサについての技術的な詳細を北陽電機株式会社　青木健氏，嶋地直広氏からいただきました．感謝いたします．

また，田代泰典　公益財団法人ニューテクノロジー振興財団事務局長におかれまして

は，本研究を始めるための道筋を開いて頂きました，ありがとうございます.

また，筑波大学知能ロボット研究室のメンバーの皆様には，山彦シンポジウム等を通してのご助言，ご指摘を賜り感謝申し上げます.

また，博士 (工学)　林道大　金沢工業高等専門学校機械工学科教授の他，同僚の教職員並びに金沢工大学園の職員の皆様方には，筆者の研究活動に対しましてご理解とご協力を賜りましたこと深く感謝申し上げます.

また，ここにお名前を挙げることが叶わない多数の方々のご支援のもと，本論文をまとめることが出来ました，深く深く感謝申し上げます.

最後に，在学間，論文のチェックや事務処理上のアドバイスや精神面で献身的に支えてくれた妻に感謝します.

参考文献

[1] 田所諭, 大須賀公一, and 天野久徳. レスキューロボット. **日本ロボット学会誌**, 19(6):685–688, 2001.

[2] 久武経夫 and 中里邦子. 大災害に立ち向かうロボットの開発. **建設の施工企画**, (740):42–47, 2011.

[3] 加藤一郎 and 今永勇生. **ヘリコプタ入門**. 1985.

[4] Kenzo Nonami, Farid Kendoul, Satoshi Suzuki, Wei Wang, and Daisuke Nakazawa. *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles*. Springer, 2010.

[5] Markus Achtelik, Abraham Bachrach, Ruijie He, Samuel Prentice, and Nicholas Roy. Autonomous navigation and exploration of a quadrotor helicopter in gps-denied indoor environments. In *First Symposium on Indoor Flight*, number 2009, 2009.

[6] William Morris, Ivan Dryanovski, Jizhong Xiao, et al. 3d indoor mapping for micro-uavs using hybrid range finders and multi-volume occupancy grids. In *RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras, Zaragoza, Spain*, 2010.

[7] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, 23(1):34–46, 2007.

[8] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[9] Chiara Troiani, Stefano Al Zanati, and Agostino Martinelli. A 3 Points Vision Based Approach for MAV Localization in GPS Denied Environments. In *6th*

European Conference on Mobile Robots, Barcelona, Espagne, 2013.

[10] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6d slam—3d mapping outdoor environments. *Journal of Field Robotics*, 24(8-9):699–722, 2007.

[11] 今村 彰隆, 三輪 昌史, and 日野 順市. クアッドコプタのマニューバ制御による三次元計測システム. In *2012* 年度 *SI2012* 予稿集, volume 1, page 2195. SICE, 2012.

[12] 野波健蔵, 岩倉大輔, and 宋澤. 産業応用型マルチロータ電動ヘリコプタのテレオペレーション技術. 日本ロボット学会誌, 30(6):574–577, 2012.

[13] 一般社団法人 日本建築学会. A 0005-1966 建築用開口部構成材の標準モデュール呼び寸法.

[14] 北陽電機株式会社. 測域センサ smart-urg mini ust-10lx(uust003) 仕様書, 2014.

[15] Toshiro Mori, Masanori Hino, Toshihiro Kamitani, and Shinichi Yuta. Development of am and pulsed sokuiki sensor using tof principle. In *Annual conference of the Robotics Society of Japan, 2007.(RSJ 2007). Proceedings. 2007 RSJ Conference on*, volume 1, page 2A13. RSJ, 2007.

[16] S. Manhart, P. Dyrna, and B. Kunkel. Ranging method and apparatus, September 13 1988. US Patent 4,770,526.

[17] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1322–1328. IEEE, 1999.

[18] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

付録 A

プロペラデータ

表 A.1: APC 5.5× 4.5 プロペラデータ

RPM [rpm]	V [mph]	J -	Pe -	Ct -	Cp -	PWR [Hp]	Torque [In-Lbf]	Thrust [Lbf]
1000	0	0	0	0.1196	0.0631	0	0.002	0.003
2000	0	0	0	0.1196	0.063	0	0.006	0.014
3000	0	0	0	0.1195	0.0631	0.001	0.014	0.031
4000	0	0	0	0.1196	0.0631	0.002	0.026	0.056
5000	0	0	0	0.1196	0.0631	0.003	0.04	0.087
6000	0	0	0	0.1196	0.0632	0.006	0.058	0.125
7000	0	0	0	0.1196	0.0632	0.009	0.079	0.171
8000	0	0	0	0.1196	0.0633	0.013	0.103	0.223
9000	0	0	0	0.1196	0.0634	0.019	0.131	0.282
10000	0	0	0	0.1196	0.0635	0.026	0.162	0.349
11000	0	0	0	0.1196	0.0636	0.034	0.196	0.422
12000	0	0	0	0.1196	0.0636	0.045	0.234	0.502
13000	0	0	0	0.1196	0.0637	0.057	0.275	0.589
14000	0	0	0	0.1196	0.0639	0.071	0.319	0.684
15000	0	0	0	0.1196	0.064	0.087	0.367	0.785
16000	0	0	0	0.1197	0.0641	0.106	0.419	0.893
17000	0	0	0	0.1197	0.0643	0.128	0.474	1.008

18000	0	0	0	0.1196	0.0636	0.15	0.526	1.129
19000	0	0	0	0.1195	0.0629	0.175	0.579	1.258
20000	0	0	0	0.1194	0.0622	0.201	0.635	1.393
21000	0	0	0	0.1194	0.0615	0.231	0.692	1.534
22000	0	0	0	0.1193	0.0609	0.263	0.752	1.683
23000	0	0	0	0.1192	0.0604	0.297	0.815	1.839
24000	0	0	0	0.1192	0.0598	0.335	0.879	2.001
25000	0	0	0	0.1191	0.0593	0.375	0.946	2.17
26000	0	0	0	0.1191	0.0588	0.418	1.014	2.346
27000	0	0	0	0.119	0.0583	0.465	1.084	2.529
28000	0	0	0	0.119	0.0578	0.514	1.157	2.719
29000	0	0	0	0.1193	0.0613	0.606	1.316	2.925
30000	0	0	0	0.1202	0.0706	0.772	1.621	3.152
31000	0	0	0	0.121	0.0805	0.971	1.974	3.391
32000	0	0	0	0.122	0.0922	1.223	2.408	3.642
33000	0	0	0	0.1235	0.1121	1.632	3.116	3.919
34000	0	0	0	0.1239	0.131	2.084	3.863	4.175
35000	0	0	0	0.126	0.1459	2.533	4.561	4.5
36000	0	0	0	0.1273	0.1631	3.081	5.395	4.81
37000	0	0	0	0.1275	0.1593	3.267	5.566	5.089
39000	0	0	0	0.1284	0.1633	3.921	6.337	5.693
40000	0	0	0	0.1298	0.1781	4.616	7.273	6.052

付録 B

飛行ロボットの慣性演算

機体に固定された慣性計測装置を用いている無人機の自己位置は，ストラップダウン方式の慣性演算により求められる．以下に，慣性演算について説明する．

クアッドロータの座標系を図 B.1 に示す．

クアッドロータに固定された座標系の x 軸回りの角速度を p，y 軸回りの角速度を q,z 軸回りの角速度を r とするとオイラー角との関係は次式で与えられる

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{B.1}$$

equation(B.1) を積分することでオイラー角を得られる．この場合は，$\theta = 90deg$ のとき不連続点が発生するためにオイラー角を用いることができない．そこで，姿勢角の演算にはクオータニオンを用いる．クオータニオンの 4 つのパラメータは機体軸の角速度 p, q, r と次のような関係がある

$$\dot{e_0} = -\frac{1}{2}(e_1 p + e_2 q + e_3 r) \tag{B.2}$$

$$\dot{e_1} = \frac{1}{2}(e_0 p + e_2 q - e_3 r) \tag{B.3}$$

$$\dot{e_2} = \frac{1}{2}(e_0 p + e_3 q - e_1 r) \tag{B.4}$$

$$\dot{e_3} = \frac{1}{2}(e_0 p + e_1 q - e_2 r) \tag{B.5}$$

これらのパラメータは常に以下の関係を満たす．

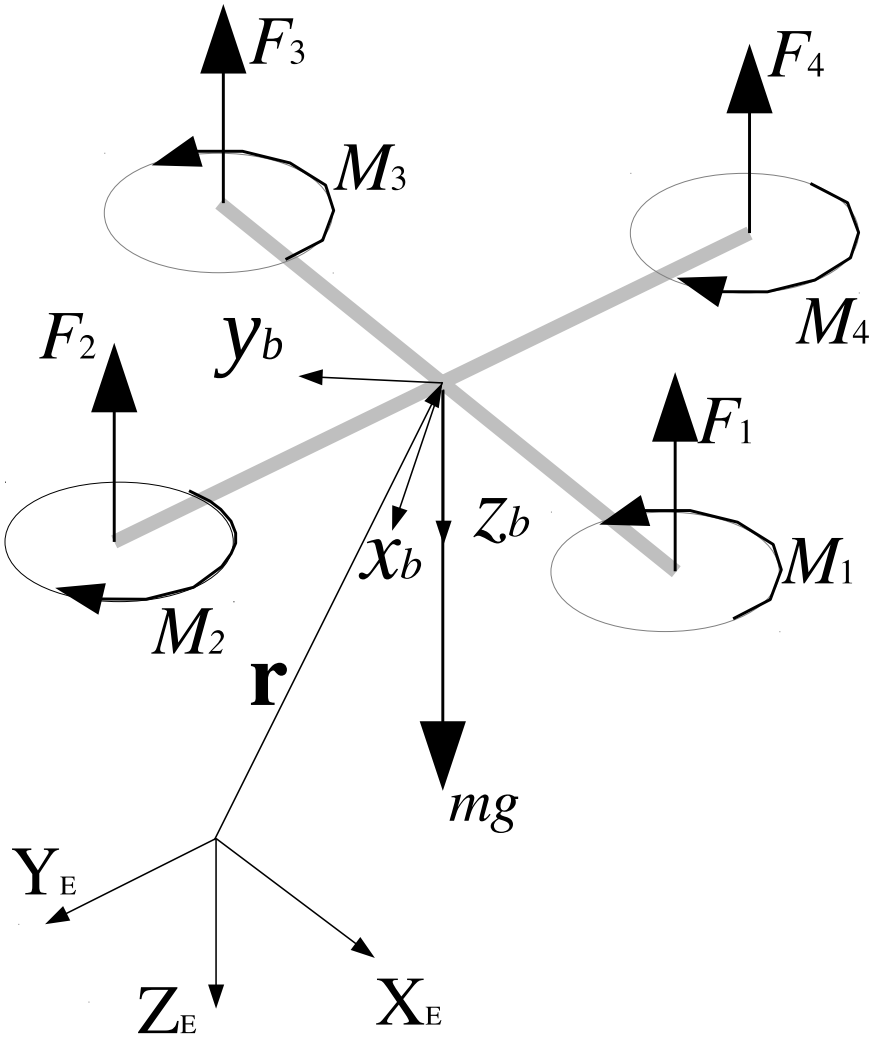


図 B.1　Coordinate system

$$e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1 \tag{B.6}$$

以上の方程式によりパラメータ e_0, e_1, e_2, e_3 を常に計算し続けることができる．その際それぞれのパラメータの初期値はオイラー角を用いて

$$e_0 = \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \tag{B.7}$$

$$e_1 = \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \tag{B.8}$$

$$e_2 = \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \tag{B.9}$$

$$e_3 = -\cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} \tag{B.10}$$

一度，4 つのパラメータが計算できると，オイラー角を次式から計算できる．

$$\theta = \cos^{-1}[-2(e_1e_3 - e_0e_2)] \tag{B.11}$$

$$\phi = \cos^{-1}\left[\frac{e_0^2 - e_1^2 - e_2^2 + e_3^2}{\sqrt{1-4(e_1e_3-e_0e_2)^2}}\right] sign[2(e_2e_3 + e_0e_1)] \tag{B.12}$$

$$\psi = \cos^{-1}\left[\frac{e_0^2 + e_1^2 - e_2^2 - e_3^2}{\sqrt{1-4(e_1e_3-e_0e_2)^2}}\right] sign[2(e_1e_2 + e_0e_3)] \tag{B.13}$$

以上から，クアッドロータの姿勢角が得られる．

IMU から得られる機体軸の加速度 a_x, a_y, a_z, 及び機体軸の角速度 p, q, r, 速度 u, v, w, 重力加速度 g より次式から $\dot{u}, \dot{v}, \dot{w}$ が計算できる．

$$\dot{u} = a_x + vr - wq + g\sin\theta \tag{B.14}$$

$$\dot{v} = a_y - ur + wp - g\cos\theta\sin\phi \tag{B.15}$$

$$\dot{w} = a_z - uq - vp - g\cos\theta\cos\phi \tag{B.16}$$

地球は地軸を回転軸として $\Omega(15deg/hour)$ で回転している

$$\boldsymbol{\Omega} = \begin{bmatrix} \Omega\cos\lambda \\ 0 \\ -\Omega sin\lambda \end{bmatrix} \tag{B.17}$$

ここで，$lambda$ は latitude である．

クアッドロータが一定高度を飛行すると地球に対して新たな回転運動が発生し次式で表される

$$\boldsymbol{\omega'} = \begin{bmatrix} \dot{\mu}\cos\lambda \\ -\dot{\lambda} \\ -\dot{\mu} sin\lambda \end{bmatrix} \tag{B.18}$$

ここで，μ は longitude である．IMU によって計測される角速度は Ω と ω' を含んでいるため，実際の角速度は次式で与えられる

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}_m - \boldsymbol{DCM}[\boldsymbol{\Omega} + \boldsymbol{\omega'}] \tag{B.19}$$

ここで $\boldsymbol{DCM}$ は the direction cosine matrix または the transformation matrix とよばれ，慣性座標から機体座標への変換を与える，以下の示される．

$$\boldsymbol{DCM} = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\theta s\phi c\psi - s\psi c\phi & s\psi s\theta s\phi + c\psi c\phi & s\phi c\theta \\ s\theta c\phi c\psi + s\psi s\phi & s\phi s\theta c\phi - c\psi s\theta & c\phi c\theta \end{bmatrix} \tag{B.20}$$

$\dot{u}, \dot{v}, \dot{w}$ は $\boldsymbol{DCM}^T$ により変換し，積分することで the navigation frame または the local earth frame における北向きの速度 V_N，東向きの速度 V_E 及び下向きの速度 V_D を求められる

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix} = \boldsymbol{DCM}^T \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{B.21}$$

V_N, V_E,V_D を積分することで, 地表での位置を計算することができる．latitude$lambda$ と longitudeμ と height$H$ との関係を見ると

$$\dot{\lambda} = \frac{V_N}{R_e} \tag{B.22}$$

$$\dot{\mu} = \frac{V_E}{R_e \cos\lambda} \tag{B.23}$$

$$\dot{H} = -V_D \tag{B.24}$$

これらから地球上の座標を計算することができる．

付録 C

飛行実験動画サムネイル画像

飛行実験時に撮影した動画の 0.0 秒から 5.8 秒までの部分を，0.2 秒毎に静止画として取り出した画像を以下に示す．鮮明な画像ではないため，補足として 0 秒の図にクアッドロータの位置を白枠で囲み示した．

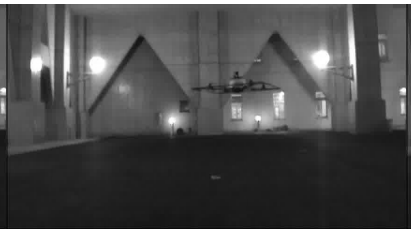
クワッドロータは当初画面左方向に移動し，続いてカメラから遠ざかりながら画面右方向に移動していることが確認できる．

表 C.1: 飛行実験動画のサムネイル画像

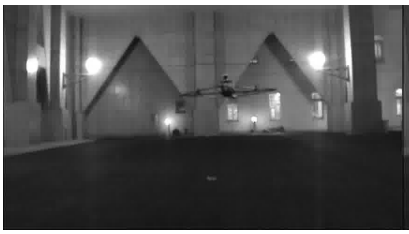
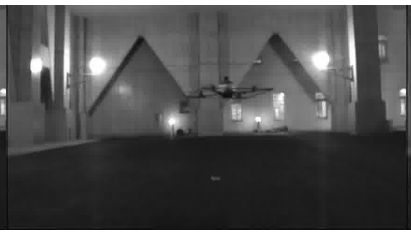
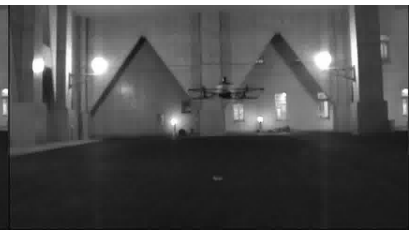
0 秒　　0.2 秒後　　0.4 秒後

0.6 秒後　　0.8 秒後　　1.0 秒後

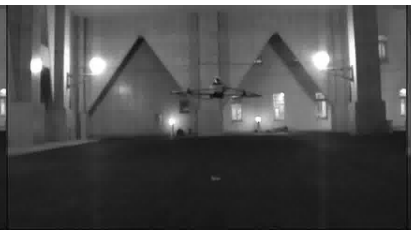
1.2 秒　　1.4 秒後　　1.6 秒後



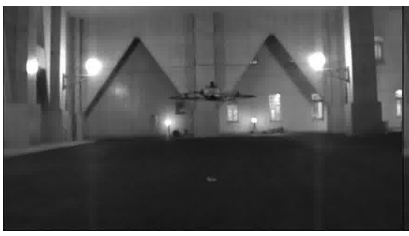
1.8 秒後　　2.0 秒後　　2.2 秒後

2.4 秒　　2.6 秒後　　2.8 秒後

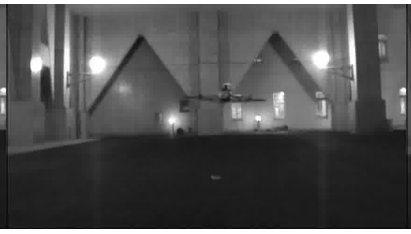


3.0 秒後　　3.2 秒後　　3.4 秒後

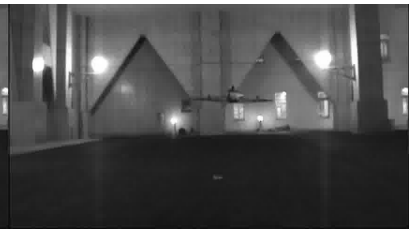


3.6 秒　　3.8 秒後　　4.0 秒後

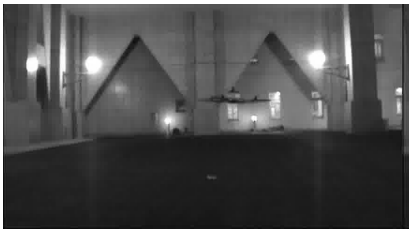
4.2 秒後



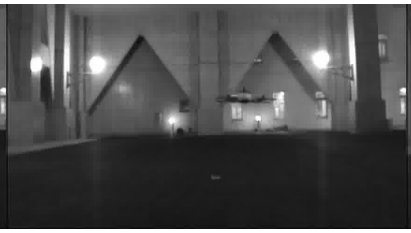
4.4 秒後



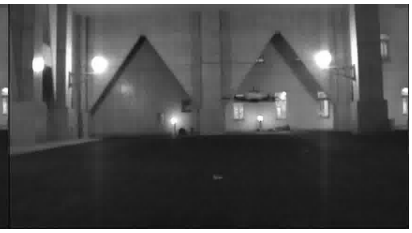
4.6 秒後



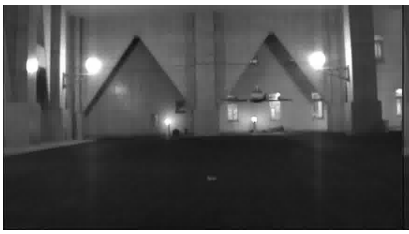
4.8 秒



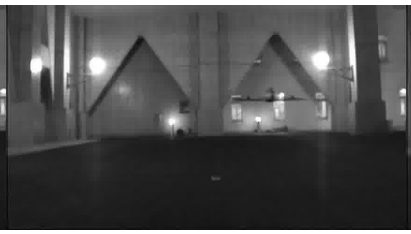
5.0 秒後



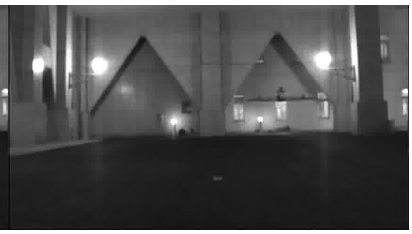
5.2 秒後



5.4 秒後



5.6 秒後



5.8 秒後

付録 D

モンテカルロ位置推定シミュレーションソースコード

ソースコード D.1　モンテカルロ位置推定シミュレーションソースコード

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "MT.h"

#define N 5000 //Number of particle
#define SN 1 //Number of sensor node
#define S_STD 0.1//Sensor STD
#define XMAX 39.12
#define YMAX 39.12
#define ROBOT_INIT_X 10.0
#define ROBOT_INIT_Y 22.0
#define RANGEMAX 30.0
#define MAPRESO 0.0489
#define MAPSIZEX 800
#define MAPSIZEY 800
#define SKIP 1

void sample_motion( double h, double ax, double ay, double r,
                    double *vx, double *vy,
                    double *x, double *y, double *psi);
double measurement_model(int n, double *particle, double *robot, double sigma
    );
double normpdf(double x, double mean, double sigma);
void measure(unsigned char *map,double x, double y, double psi, double *
    true_r);
```

```
double normal(double mu, double dist);
void resample(double *x, double *y, double *psi, double *vx, double *vy,
    double *w);
double raycast(unsigned char *map,double sx,double sy,double co,double si,
    double reso,int xmax,int ymax);

int main(void){
        //File
        FILE *fpr,*fpp,*fps,*fpi,*fpm;

        char *smap="map48_9mm.bmp";
        char *sr="robot.dat";
        char *sp="particle.dat";
        char *ss="sensor.dat";
        char *si="imu.dat";

        //General
        int i,j,k,l,m,size;
        double t =0.0;
        double h=0.01;
        unsigned seed;

        //Particle
        double px[N],py[N],ppsi[N],pvx[N],pvy[N],pw[N];//Particle state
        double sum;

        //Robot
        double rx,ry,rpsi;//robot pose
        double rvx,rvy;//robot velocity
        double rax,ray,rr;//robot control
        double oldrx,oldry;

        //IMU
        double iax,iay,iaxold,iayold;
        double ivx,ivy,ir,ivxold,ivyold,irold;
        double ix,iy,ipsi;

        //Estimate
        double xhat,yhat;

        //Sensor
        double range_virtual[SN],range_measure[SN];
        double gpsi,spsi,sk,sx,sy,range;

        //Map
        unsigned char buf[2800000],map[640000];
        double xmax=XMAX;//map x size max
```

```
71          double ymax=YMAX;//map y size max
72
73
74          seed = (unsigned) time(NULL);
75          init_genrand(1418228112);
76          printf("Seed=%d\n",seed);
77
78          fpm = fopen( smap, "rb" );
79          if( fpm == NULL ){
80                  printf( "%s ファイルが開けません\n", smap );
81                  return −1;
82          }
83          size = fread( buf, sizeof( unsigned char ), 2800000, fpm );
84
85          for(i=0;i<640000;i++){
86                  map[i]=buf[139+i*4];
87          }
88          fclose(fpm);
89
90
91
92          fpr = fopen( sr, "w" );
93          if( fpr == NULL ){
94          printf( "%s ファイルが開けません¥n", sr );
95          return −1;
96          }
97          fpp = fopen( sp, "w" );
98          if( fpp == NULL ){
99          printf( "%s ファイルが開けません¥n", sp );
100         return −1;
101         }
102         fps = fopen( ss, "w" );
103         if( fps == NULL ){
104         printf( "%s ファイルが開けません¥n", ss );
105         return −1;
106         }
107
108         fpi = fopen( si, "w" );
109         if( fpi == NULL ){
110         printf( "%s ファイルが開けません¥n", si );
111         return −1;
112         }
113
114         //Initialize robot pose
115         rx = ROBOT_INIT_X;
116         ry = ROBOT_INIT_Y;
117         rpsi = 0.0;
```

```
118         //Initialize robot state
119         rvx=0.0;
120         rvy=0.0;
121         //Initialize robot control
122         rax=0.0;
123         ray=0.0;
124         rr=0.0;
125
126         //Initialize IMU
127         iax=iay=ir=iaxold=iayold=0.0;
128         ivx=ivy=ir=ivxold=ivyold=irold=0.0;
129         ix=rx;
130         iy=ry;
131
132         //Initialize Estimate
133         xhat = 0.0;
134         yhat = 0.0;
135
136         //Initialize particle
137         for(i=0;i<N;i++){
138                 //px[i]=ROBOT_INIT_X+normal(0.0,1.0);//XMAX*genrand_real1
                        ();
139                 //py[i]=ROBOT_INIT_Y+normal(0.0,1.0);//YMAX*genrand_real1
                        ();
140                 do {
141                         px[i]=XMAX*genrand_real1();
142                         py[i]=YMAX*genrand_real1();
143                 }while(map[((int)(px[i]/MAPRESO))%MAPSIZEX+((int)(py[i]/
                        MAPRESO))*MAPSIZEY]==0);
144                 ppsi[i]=2.0*M_PI*genrand_real2()/100.0;
145                 pvx[i]=0.0;
146                 pvy[i]=0.0;
147                 pw[i]=1.0;
148                 //printf("%f %f %f\n",px[i],py[i],pw[i]);
149         }
150
151         //Main loop
152         for(i=0;i<1500;i++){
153                 printf("[%d]",i);
154                 fflush(stdout);
155
156
157                 //Save
158                 if(i%SKIP==0){
159                         //Robot
160                         fprintf(fpr,"#Time␣%f\n",h*i);
```

```
161                     fprintf(fpr,"%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %
                            lf %d\n",t,rx,ry,rpsi,rvx,rvy,rr,rax,ray,xhat,yhat,seed);
162                     fprintf(fpr,"\n");
163
164                     //Particle
165                     fprintf(fpp,"#Time %f\n",h*i);
166                     for(m=0;m<N;m++){
167                             fprintf(fpp,"%d %lf %lf %le\n",m,px[m],py[m],pw[
                                    m]);
168                     }
169                     fprintf(fpp,"\n");
170
171                     //Sensor
172                     fprintf(fps,"#Time %f\n",h*i);
173                     for(j=0;j<SN;j++){
174                             spsi = j*2.0*M_PI/SN;
175                             fprintf(fps,"%lf %lf %lf %lf %lf %lf\n",rx,ry,rpsi
                                    ,spsi,range_virtual[j],range_measure[j]);
176                     }
177                     fprintf(fps,"\n");
178
179                     //IMU
180                     fprintf(fpi,"#Time %f\n",h*i);
181                     fprintf(fpi,"%lf %lf %lf %lf %lf %lf %lf %lf %lf\n",t,ix
                            ,iy,ipsi,ivx,ivy,ir,iax,iay);
182                     fprintf(fpi,"\n");
183             }
184
185             //Move robot
186             //Control
187             if(i<200){
188                     rax = 0.0;
189                     ray = 0.0;
190                     rr = 4*M_PI;
191             }
192             else if(i<400){
193                     rax = 1.0;
194                     ray = 0.0;
195                     rr = M_PI*0;
196             }
197             else if(i<600){
198                     rax = -1.0;
199                     ray = 0.0;
200                     rr = 0*M_PI;
201             }
202             else if(i<800){
203                     rax = 0.0;
```

```
                ray = 0.0;
                rr = M_PI*4;
        }
        else if(i<1000){
                rax = 0.0;
                ray = 1.0;
                rr = M_PI*0;
        }
        else if(i<1200){
                rax = 0.0;
                ray = -1.0;
                rr = 0*M_PI;
        }
        else if(i<1400){
                rax = 0.0;
                ray = 0.0;
                rr = 4*M_PI;
        }


        else {
                rax = 0.0;
                ray = 0.0;
                rr = 4*M_PI;
        }

        oldrx=rx;
        oldry=ry;
        rx = rx + 0.5*rax*h*h + rvx*h;
        ry = ry + 0.5*ray*h*h + rvy*h;
        if(map[((int)(rx/MAPRESO))%MAPSIZEX+((int)(ry/MAPRESO
            ))*MAPSIZEY]==0||rx<0.0||ry<0.0||rx>XMAX||ry>YMAX){
                rx = oldrx;
                ry = oldry;
                rax=0.0;
                ray=0.0;
                rvx=0.0;
                rvy=0.0;
        }


        rvx = rvx + rax*h + 1.0*normal(0.0,0.5)*h;
        rvy = rvy + ray*h + 1.0*normal(0.0,0.5)*h;
        rpsi = rpsi + rr*h;

        //IMU
        iaxold = iax;
```

```
iayold = iay;
ivxold = ivx;
ivyold = ivy;
irold = ir;
iax = rax+normal(normal(0.0,1.0), 5.0*rax);
iay = ray+normal(normal(0.0,1.0), 5.0*ray);
ir = rr+normal(0.0,0.1*rr);

//Estimate position using IMU
ivx = ivx + (iax+iaxold)*h*0.5;
ivy = ivy + (iay+iayold)*h*0.5;
ix= ix+(ivx+ivxold)*h*0.5;
iy= iy+(ivy+ivyold)*h*0.5;
ipsi = (ir+irold)*h*0.5;



//Robot measure
if(i%3==0){//Measure range interval is 3ms.
        measure(map,rx, ry, rpsi, range_virtual);
        for(j=0;j<SN;j++){
                range_measure[j]=range_virtual[j]+normal(0.0,0.1);
        }
}

//Particle Motion & Measure
sum=0.0;
for(m=0;m<N;m++){
        //Particle Motion
        sample_motion(h,iax, iay, ir, &(pvx[m]), &(pvy[m]), &(px[m
            ]), &(py[m]), &(ppsi[m]));
        //Particle Measure
        measure(map,px[m],py[m],ppsi[m],range_virtual);
        pw[m]=measurement_model(SN,range_virtual,range_measure
            ,S_STD);
        sum=sum+pw[m];
}
//Particle weight normlize
for(m=0;m<N;m++){
        pw[m]=pw[m]/sum;
}

if(i%3==0){
        //Paricle Resample
        resample(px, py, ppsi, pvx, pvy, pw);

        //Estimate position
```

```
295                             sum=0.0;
296                             xhat=0.0;
297                             yhat=0.0;
298                             for(m=0;m<N;m++){
299                                     sum=pw[m]+sum;
300                                     xhat=xhat+pw[m]*px[m];
301                                     yhat=yhat+pw[m]*py[m];
302                             }
303                             xhat = xhat/sum;
304                             yhat = yhat/sum;
305                     }
306
307                     t = t+h;
308             }
309             //Main loop end
310             printf("\n");
311
312             fclose(fpr);
313             fclose(fpp);
314             fclose(fps);
315             fclose(fpi);
316             return 0;
317     }
318
319
320     void sample_motion( double h, double ax, double ay, double r,
321                                     double *vx, double *vy,
322                                     double *x, double *y, double *psi)
323     {
324             double wx,wy,axn,ayn,rn;
325             axn = ax + normal(normal(0.0,1.0), 10.0*ax);
326             ayn = ay + normal(normal(0.0,1.0), 10.0*ay);
327             rn = r + normal(0.0, 0.1*r);
328             wx = normal(0.0,0.5);
329             wy = normal(0.0,0.5);
330
331             *x = *x + 0.5*axn*h*h + *vx*h;
332             *y = *y + 0.5*ayn*h*h + *vy*h;
333
334             if(*x<0.0)*x=0.0;
335             if(*y<0.0)*y=0.0;
336             if(*x>XMAX)*x=XMAX;
337             if(*y>YMAX)*y=YMAX;
338
339             *vx = *vx + axn*h + wx*h;
340             *vy = *vy + ayn*h + wy*h;
341             *psi = *psi + rn*h;
```

```
}

double measurement_model(int n, double *particle, double *robot, double sigma){

        int i;
        double q,p;

        q =1.0;
        for(i=0;i<n;i++){
                p = normpdf(particle[i],robot[i],sigma);
                q = q*p;
        }
        return q;

}

double normpdf(double x, double mean, double sigma){

        return exp(-0.5*(x-mean)*(x-mean)/sigma/sigma)/sqrt(2*M_PI)/sigma;

}

void measure(unsigned char *map,double x, double y, double psi, double *measure){
        double spsi,gpsi,sx,sy,sk;
        int j;
        for(j=0;j<SN;j++){
                spsi = j*2.0*M_PI/SN;
                gpsi = spsi + psi;
                measure[j]=raycast(map,x,y,cos(gpsi),sin(gpsi),MAPRESO,MAPSIZEX,MAPSIZEY);

        }
}

double raycast(unsigned char *map,double sx,double sy,double co,double si,double reso,int xmax,int ymax){
        int i;
        int xi,yi;
        double x,y,k,oldk,x1,y1,x2,y2,range1,range2;
        double maxrange=40.0;
        //x-loop
        if(co>0.0){
                xi=(int)(sx/reso)+1;
                for(i=xi;i<xmax-1;i++){
```

```
385                     x=i*reso;
386                     if(si==0.0) y =sy;
387                     else y=(x-sx)*si/co+sy;
388                     xi = i;
389                     yi = (int)(y/reso);
390                     if(yi>=ymax||yi<0){
391                             x = maxrange*co+sx;y=maxrange*si+sy;
392                             break;
393                     }
394                     //printf("x1 %lf % lf %d %d\n",co,si,xi,yi);
395                     if(map[xi+yi*xmax]==0 || map[(xi-1)+yi*xmax]==0)
                            break;
396                     x = maxrange*co+sx;y=maxrange*si+sy;
397             }
398     }
399     else if(co<0.0){
400             xi=(int)(sx/reso);
401             for(i=xi;i>0;i--){
402                     x=i*reso;
403                     if(si==0.0) y =sy;
404                     else y=(x-sx)*si/co+sy;
405                     xi = i;
406                     yi = (int)(y/reso);
407                     if(yi>=ymax||yi<0){
408                             x = maxrange*co+sx;y=maxrange*si+sy;
409                             break;
410                     }
411                     //printf("x2 %d %d\n",xi,yi);
412                     if(map[xi+yi*xmax]==0 || map[(xi-1)+yi*xmax]==0)
                            break;
413                     x = maxrange*co+sx;y=maxrange*si+sy;
414             }
415     }
416     x1=x;y1=y;
417
418     //y-loop
419             if(si>0.0){
420             yi=(int)(sy/reso)+1;
421             for(i=yi;i<ymax-1;i++){
422                     y=i*reso;
423                     if(co==0.0) x =sx;
424                     else x=(y-sy)*co/si+sx;
425                     yi = i;
426                     xi = (int)(x/reso);
427                     if(xi>=xmax || xi<0){
428                             x = maxrange*co+sx;y=maxrange*si+sy;
429                             break;
```

```
430                         }
431                         //printf("y1 %lf %d %d\n",co,xi,yi);
432                         if(map[xi+yi*xmax]==0 || map[(xi-1)+(yi-1)*xmax]==0)
                                break;
433                         x = maxrange*co+sx;y=maxrange*si+sy;
434                 }
435         }
436         else{
437                 yi=(int)(sy/reso);
438                 for(i=yi;i>0;i--){
439                         y=i*reso;
440                         if(co==0.0) x = sx;
441                         else x=(y-sy)*co/si+sx;
442                         yi = i;
443                         xi = (int)(x/reso);
444                         if(xi>=xmax||xi<0){
445                                 x = maxrange*co+sx;y=maxrange*si+sy;
446                                 break;
447                         }
448                         //printf("y2 %d %d\n",xi,yi);
449                         if(map[xi+yi*xmax]==0 || map[(xi-1)+(yi-1)*xmax]==0)
                                break;
450                         x = maxrange*co+sx;y=maxrange*si+sy;
451                 }
452         }
453         x2=x;y2=y;
454
455
456         range1 = sqrt((x1-sx)*(x1-sx)+(y1-sy)*(y1-sy));
457         range2 = sqrt((x2-sx)*(x2-sx)+(y2-sy)*(y2-sy));
458         if(range1>=range2)return range2;
459         else return range1;
460
461 }
462
463
464 void resample(double *x, double *y, double *psi, double *vx, double *vy,
        double *w){
465         double xb[N],yb[N],psib[N],vxb[N],vyb[N],wb[N];
466         double r,c,U,h;
467         int i,m;
468
469         h=1.0/N;
470         r = 0.5*h;//genrand_real1()*h;
471         c= w[0];
472         i=0;
473         for(m=0;m<N;m++){
```

```
474                 U = r + m*h;
475
476                 while(U>c){
477                         i = i + 1;
478                         c = c + w[i];
479                         //printf("%lf %lf %d\n",U,c,i);
480                 }
481
482                 xb[m]=x[i];
483                 yb[m]=y[i];
484                 psib[m]=psi[i];
485                 vxb[m]=vx[i];
486                 vyb[m]=vy[i];
487                 wb[m]=w[i];
488         }
489
490         for(m=0;m<N;m++){
491                 x[m]=xb[m];
492                 y[m]=yb[m];
493                 psi[m]=psib[m];
494                 vx[m]=vxb[m];
495                 vy[m]=vyb[m];
496                 w[m]=wb[m];
497         }
498
499 }
500
501
502 double normal(double mu, double dist){
503         return mu+0.5*sqrt(6)*( (genrand_real1()/0.5-1)*dist+(genrand_real1
                ()/0.5-1)*dist);
504 }
```

付録 E

モンテカルロ位置推定飛行実験データ処理ソースコード

ソースコード E.1　モンテカルロ位置推定シミュレーションソースコード

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "MT.h"

#define N 10000 //Number of particle
#define SN 1 //Number of sensor node
#define S_STD 0.1//Sensor STD
#define XMAX 39.12
#define YMAX 39.12
#define ROBOT_INIT_X 10.5
#define ROBOT_INIT_Y 21.0
#define RANGEMAX 30.0
#define MAPRESO 0.0489
#define MAPSIZEX 800
#define MAPSIZEY 800
#define SKIP 1

void sample_motion( double h, double ax, double ay, double r,
                    double *vx, double *vy,
                    double *x, double *y, double *psi);
double measurement_model(int n, double *particle, double *robot, double sigma
    );
double normpdf(double x, double mean, double sigma);
void measure(unsigned char *map,double x, double y, double psi, double *
    true_r);
```

```
double normal(double mu, double dist);
void resample(double *x, double *y, double *psi, double *vx, double *vy,
    double *w);
double raycast(unsigned char *map,double sx,double sy,double co,double si,
    double reso,int xmax,int ymax);

int main(void){
        //File
        FILE *fpr,*fpp,*fps,*fpi,*fpm,*fprange,*fpimu;

        char *smap="map48_9mm.bmp";
        char *sr="robot.dat";
        char *sp="particle.dat";
        char *ss="sensor.dat";
        char *si="imu.dat";
        char *rangename="range1118.dat";
        char *imuname="imu1118.dat";


        //General
        int i,j,k,l,m,size;
        double t =0.0;
        double h=0.01;
        double tim2;
        unsigned seed;
        char dummy[20];

        //Particle
        double px[N],py[N],ppsi[N],pvx[N],pvy[N],pw[N];//Particle state
        double sum;

        //Robot
        double rx,ry,rpsi;//robot pose
        double rvx,rvy;//robot velocity
        double rax,ray,rr;//robot control
        double oldrx,oldry;
        double tim,tim0,raz,rp,rq,rqx,rqy,rqz,rqw;
        double rax2,ray2;

        //IMU
        double iax,iay,iaxold,iayold;
        double ivx,ivy,ir,ivxold,ivyold,irold;
        double ix,iy,ipsi;

        //Estimate
        double xhat,yhat,psihat,vxhat,vyhat;

```

```
71          //Sensor
72          double range_virtual[SN],range_measure[SN];
73          double gpsi,spsi,sk,sx,sy,range;
74          double echo1,echo2,diff;
75
76          //Map
77          unsigned char buf[2800000],map[640000];
78          double xmax=XMAX;//map x size max
79          double ymax=YMAX;//map y size max
80
81          //Initialize rand
82          seed = (unsigned) time(NULL);
83          init_genrand(1418515869);
84          printf("Seed=%d\n",seed);
85
86          //File open
87          fpm = fopen( smap, "rb" );
88          if( fpm == NULL ){
89                  printf( "%s ファイルが開けません\n", smap );
90                  return −1;
91          }
92          size = fread( buf, sizeof( unsigned char ), 2800000, fpm );
93
94          for(i=0;i<640000;i++){
95                  map[i]=buf[139+i*4];
96          }
97          fclose(fpm);
98
99
100
101         fpr = fopen( sr, "w" );
102         if( fpr == NULL ){
103         printf( "%s ファイルが開けません¥n", sr );
104         return −1;
105         }
106         fpp = fopen( sp, "w" );
107         if( fpp == NULL ){
108         printf( "%s ファイルが開けません¥n", sp );
109         return −1;
110         }
111         fps = fopen( ss, "w" );
112         if( fps == NULL ){
113         printf( "%s ファイルが開けません¥n", ss );
114         return −1;
115         }
116
117         fpi = fopen( si, "w" );
```

```
118         if( fpi == NULL ){
119         printf( "%s ファイルが開けません¥n", si );
120         return −1;
121         }
122
123         fprange = fopen( rangename, "r" );
124                 if( fprange == NULL ){
125                 printf( "%s ファイルが開けません¥n", rangename );
126                 return −1;
127         }
128
129         fpimu = fopen( imuname, "r" );
130                 if( fpimu == NULL ){
131                 printf( "%s ファイルが開けません¥n", imuname );
132                 return −1;
133         }
134
135
136
137         //Initialize robot pose
138         rx = ROBOT_INIT_X;
139         ry = ROBOT_INIT_Y;
140         rpsi = M_PI*90.0/180.0;
141         //Initialize robot state
142         rvx=0.0;
143         rvy=0.0;
144         //Initialize robot control
145         rax=0.0;
146         ray=0.0;
147         rr=0.0;
148
149         //Initialize IMU
150         iax=iay=ir=iaxold=iayold=0.0;
151         ivx=ivy=ir=ivxold=ivyold=irold=0.0;
152         ix=rx;
153         iy=ry;
154
155         //Initialize Estimate
156         xhat = 0.0;
157         yhat = 0.0;
158         psihat =0.0;
159
160         //Initialize particle
161         for(i=0;i<N;i++){
162                 //px[i]=ROBOT_INIT_X+normal(0.0,1.0);//XMAX*genrand_real1
                        ();
```

```
163                 //py[i]=ROBOT_INIT_Y+normal(0.0,1.0);//YMAX*genrand_real1
                        ();
164                 do {
165                         px[i]=normal(ROBOT_INIT_X,1.5);//(10.0*genrand_real1
                                ()-5.0)+ROBOT_INIT_X;//;
166                         py[i]=normal(ROBOT_INIT_Y,1.5);//(10.0*genrand_real1
                                ()-5.0)+ROBOT_INIT_Y;//normal(ROBOT_INIT_Y
                                ,5.0);
167                 }while(map[((int)(px[i]/MAPRESO))%MAPSIZEX+((int)(py[i]/
                        MAPRESO))*MAPSIZEY]==0);
168                 ppsi[i]=90.0*M_PI/180.0+2.0*M_PI*genrand_real2()/100.0;
169                 pvx[i]=0.0;
170                 pvy[i]=0.0;
171                 pw[i]=1.0;
172                 //printf("%f %f %f\n",px[i],py[i],pw[i]);
173         }
174
175
176         fscanf( fprange, "%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%lf␣%lf␣%lf␣%lf",dummy,
                dummy,dummy,dummy,
177                 &tim,&echo1,&echo2,&diff );
178         tim0=tim;
179         i=0;
180         while(tim-tim0<210.0||(echo2-echo1)<0.0){
181                 fscanf( fprange, "%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%lf␣%lf␣%lf␣%lf",
                        dummy,dummy,dummy,dummy,
182                 &tim,&echo1,&echo2,&diff );
183                 i++;
184
185         }
186         do{
187                 fscanf( fpimu, "%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%[^
                        ␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%
                        lf␣%lf␣%lf",
188                         dummy,dummy,dummy,dummy,dummy,dummy,dummy,
                                dummy,dummy,dummy,dummy,
189                         &tim2,&rax,&ray,&raz,&rp,&rq,&rr,&rqx,&rqy,&rqz,&rqw
                                );
190
191         }while(tim2<tim);
192
193
194         //Main loop
195         for(i=0;i<500;i++){
196                 printf("[%d]",i);
197                 fflush(stdout);
198
```

```
199
200                 //Save
201                 if(i%SKIP==0){
202                         //Robot
203                         fprintf(fpr,"#Time␣%f\n",h*i);
204                         fprintf(fpr,"%lf␣%lf␣%lf␣%lf␣%d\n",t,xhat,yhat,psihat,seed
                                );
205                         fprintf(fpr,"\n");
206
207                         //Particle
208                         fprintf(fpp,"#Time␣%f\n",h*i);
209                         for(m=0;m<N;m++){
210                                 fprintf(fpp,"%d␣%lf␣%lf␣%le\n",m,px[m],py[m],pw[
                                        m]);
211                         }
212                         fprintf(fpp,"\n");
213
214                         //Sensor
215                         fprintf(fps,"#Time␣%f\n",h*i);
216                         for(j=0;j<SN;j++){
217                                 spsi = j*2.0*M_PI/SN;
218                                 fprintf(fps,"%lf␣%lf␣%lf␣%lf␣%lf␣%lf\n",rx,ry,rpsi
                                        ,spsi,range_virtual[j],range_measure[j]);
219                         }
220                         fprintf(fps,"\n");
221
222                         //IMU
223                         fprintf(fpi,"#Time␣%f\n",h*i);
224                         fprintf(fpi,"%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf\n",t,ix
                                ,iy,ipsi,ivx,ivy,ir,iax,iay);
225                         fprintf(fpi,"\n");
226                 }
227
228                 //Move robot
229                 //Control
230                 fscanf( fpimu, "%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%[^
                        ␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%lf␣%
                        lf␣%lf␣%lf",
231                         dummy,dummy,dummy,dummy,dummy,dummy,dummy,
                                dummy,dummy,dummy,dummy,
232                         &tim,&rax,&ray,&raz,&rp,&rq,&rr,&rqx,&rqy,&rqz,&rqw);
233                 rr=-rr;
234                 rax2 = rax*cos(psihat)+ray*sin(psihat);
235                 ray2 = rax*sin(psihat)-ray*cos(psihat);
236                 rax = rax2;
237                 ray = ray2;
238
```

```
//IMU
iaxold = iax;
iayold = iay;
ivxold = ivx;
ivyold = ivy;
irold = ir;
iax = rax;
iay = ray;
ir = rr;

//Estimate position using IMU
ivx = ivx + (iax+iaxold)*h*0.5;
ivy = ivy + (iay+iayold)*h*0.5;
ix= ix+(ivx+ivxold)*h*0.5;
iy= iy+(ivy+ivyold)*h*0.5;
ipsi = (ir+irold)*h*0.5;



//Robot measure
do{
        fscanf( fprange, "%[^␣]␣%[^␣]␣%[^␣]␣%[^␣]␣%lf␣%lf␣%lf␣%lf",dummy,dummy,dummy,dummy,
        &tim2,&echo1,&echo2,&diff );
}while(tim2<tim);
range_measure[0]=echo2-echo1;


printf("%lf␣%lf␣%lf",tim2,180.0*psihat/M_PI,range_measure[0]);

//Particle Motion & Measure
sum=0.0;
for(m=0;m<N;m++){

        //Particle Motion
                sample_motion(h,iax, iay, ir, &(pvx[m]), &(pvy[m]), &(px[m]), &(py[m]), &(ppsi[m]));
        //Particle Measure
        measure(map,px[m],py[m],ppsi[m],range_virtual);
        pw[m]=measurement_model(SN,range_virtual,range_measure,S_STD);
        sum=sum+pw[m];
}
//Particle weight normlize
for(m=0;m<N;m++){
        pw[m]=pw[m]/sum;
}
```

```
283
284
285                 //Paricle Resample
286                 resample(px, py, ppsi, pvx, pvy, pw);
287
288                 //Estimate position
289                 sum=0.0;
290                 xhat=0.0;
291                 yhat=0.0;
292                 psihat =0.0;
293                 for(m=0;m<N;m++){
294                         sum=pw[m]+sum;
295                         xhat=xhat+pw[m]*px[m];
296                         yhat=yhat+pw[m]*py[m];
297                         psihat = psihat + pw[m]*ppsi[m];
298                 }
299                 xhat = xhat/sum;
300                 yhat = yhat/sum;
301                 psihat = psihat/sum;
302
303
304                 t = t+h;
305         }
306         //Main loop end
307         printf("\n");
308
309         fclose(fpr);
310         fclose(fpp);
311         fclose(fps);
312         fclose(fpi);
313         fclose(fprange);
314         fclose(fpimu);
315         return 0;
316 }
317
318
319 void sample_motion( double h, double ax, double ay, double r,
320                                 double *vx, double *vy,
321                                 double *x, double *y, double *psi)
322 {
323         double wx,wy,axn,ayn,rn;
324         axn = ax + normal(normal(0.0,0.0), 1.0*ax);
325         ayn = ay + normal(normal(0.0,0.0), 1.0*ay);
326         rn = r + normal(0.0, 1.0*r);
327         wx = normal(0.0,0.5);
328         wy = normal(0.0,0.5);
329
```

```
        *x = *x + 0.5*axn*h*h + *vx*h;
        *y = *y + 0.5*ayn*h*h + *vy*h;

        if(*x<0.0)*x=0.0;
        if(*y<0.0)*y=0.0;
        if(*x>XMAX)*x=XMAX;
        if(*y>YMAX)*y=YMAX;

        *vx = *vx + axn*h + wx*h;
        *vy = *vy + ayn*h + wy*h;
        *psi = *psi + rn*h;
        if(r>0.5)printf("r=%lf\n",r);

}

double measurement_model(int n, double *particle, double *robot, double sigma
    ){

        int i;
        double q,p,p1,p2,p3,p4;

        q =1.0;
        for(i=0;i<n;i++){
                if(particle[i]>RANGEMAX)particle[i]=RANGEMAX;
                if(robot[i]>0.0&&robot[i]<RANGEMAX)
                        p1 = normpdf(particle[i],robot[i],sigma);
                else
                        p1=0.0;
                if(robot[i]>0.0&&robot[i]<particle[i])
                        p2=0.001*exp(-0.001*robot[i])/(1-exp(-0.001*particle[i]));
                else
                        p2=0.0;

                if(robot[i]<0.7||robot[i]>RANGEMAX){
                        p3=1.0;
                }
                else p3=0.0;
                if(robot[i]>0.0&&robot[i]<RANGEMAX)
                        p4 = 1/RANGEMAX;
                else p4 = 0.0;
                p= 0.8*p1+0.05*p2+0.1*p3+0.05*p4;
                q = q*p;
                if(q==0.0)printf("likelehood %lf particle %lf robot %lf\n",p,
                    particle[i],robot[i]);
        }

        return q;
```

```
}

double normpdf(double x, double mean, double sigma){

        return exp(-0.5*(x-mean)*(x-mean)/sigma/sigma)/sqrt(2*M_PI)/sigma;

}

void measure(unsigned char *map,double x, double y, double psi, double *
    measure){
        double spsi,gpsi,sx,sy,sk;
        int j;
        for(j=0;j<SN;j++){
                spsi = j*2.0*M_PI/SN;
                gpsi = spsi + psi;
                measure[j]=raycast(map,x,y,cos(gpsi),sin(gpsi),MAPRESO,
                        MAPSIZEX,MAPSIZEY);

        }
}

double raycast(unsigned char *map,double sx,double sy,double co,double si,
    double reso,int xmax,int ymax){
        int i;
        int xi,yi;
        double x,y,k,oldk,x1,y1,x2,y2,range1,range2;
        double maxrange=40.0;
        //x-loop
        if(co>0.0){
                xi=(int)(sx/reso)+1;
                for(i=xi;i<xmax-1;i++){
                        x=i*reso;
                        if(si==0.0) y =sy;
                        else y=(x-sx)*si/co+sy;
                        xi = i;
                        yi = (int)(y/reso);
                        if(yi>=ymax||yi<0){
                                x = maxrange*co+sx;y=maxrange*si+sy;
                                break;
                        }
                        //printf("x1 %lf % lf %d %d\n",co,si,xi,yi);
                        if(map[xi+yi*xmax]==0 || map[(xi-1)+yi*xmax]==0)
                            break;
                        x = maxrange*co+sx;y=maxrange*si+sy;
                }
        }
```

```
418         else if(co<0.0){
419                 xi=(int)(sx/reso);
420                 for(i=xi;i>0;i--){
421                         x=i*reso;
422                         if(si==0.0) y =sy;
423                         else y=(x-sx)*si/co+sy;
424                         xi = i;
425                         yi = (int)(y/reso);
426                         if(yi>=ymax||yi<0){
427                                 x = maxrange*co+sx;y=maxrange*si+sy;
428                                 break;
429                         }
430                         //printf("x2 %d %d\n",xi,yi);
431                         if(map[xi+yi*xmax]==0 || map[(xi-1)+yi*xmax]==0)
                                break;
432                         x = maxrange*co+sx;y=maxrange*si+sy;
433                 }
434         }
435         x1=x;y1=y;
436
437         //y-loop
438                 if(si>0.0){
439                 yi=(int)(sy/reso)+1;
440                 for(i=yi;i<ymax-1;i++){
441                         y=i*reso;
442                         if(co==0.0) x =sx;
443                         else x=(y-sy)*co/si+sx;
444                         yi = i;
445                         xi = (int)(x/reso);
446                         if(xi>=xmax || xi<0){
447                                 x = maxrange*co+sx;y=maxrange*si+sy;
448                                 break;
449                         }
450                         //printf("y1 %lf %d %d\n",co,xi,yi);
451                         if(map[xi+yi*xmax]==0 || map[(xi-1)+(yi-1)*xmax]==0)
                                break;
452                         x = maxrange*co+sx;y=maxrange*si+sy;
453                 }
454         }
455         else{
456                 yi=(int)(sy/reso);
457                 for(i=yi;i>0;i--){
458                         y=i*reso;
459                         if(co==0.0) x = sx;
460                         else x=(y-sy)*co/si+sx;
461                         yi = i;
462                         xi = (int)(x/reso);
```

```
                        if(xi>=xmax||xi<0){
                                x = maxrange*co+sx;y=maxrange*si+sy;
                                break;
                        }
                        //printf("y2 %d %d\n",xi,yi);
                        if(map[xi+yi*xmax]==0 || map[(xi-1)+(yi-1)*xmax]==0)
                            break;
                        x = maxrange*co+sx;y=maxrange*si+sy;
                }
        }
        x2=x;y2=y;


        range1 = sqrt((x1-sx)*(x1-sx)+(y1-sy)*(y1-sy));
        range2 = sqrt((x2-sx)*(x2-sx)+(y2-sy)*(y2-sy));
        if(range1>=range2)return range2;
        else return range1;

}


void resample(double *x, double *y, double *psi, double *vx, double *vy,
    double *w){
        double xb[N],yb[N],psib[N],vxb[N],vyb[N],wb[N];
        double r,c,U,h;
        int i,m;

        h=1.0/N;
        r = 0.5*h;//genrand_real1()*h;
        c= w[0];
        i=0;
        for(m=0;m<N;m++){
                U = r + m*h;

                while(U>c){
                        i = i + 1;
                        c = c + w[i];
                        //printf("%lf %lf %d\n",U,c,i);
                }

                xb[m]=x[i];
                yb[m]=y[i];
                psib[m]=psi[i];
                vxb[m]=vx[i];
                vyb[m]=vy[i];
                wb[m]=w[i];
        }
```

```
508
509         for(m=0;m<N;m++){
510                 x[m]=xb[m];
511                 y[m]=yb[m];
512                 psi[m]=psib[m];
513                 vx[m]=vxb[m];
514                 vy[m]=vyb[m];
515                 w[m]=wb[m];
516         }
517
518 }
519
520
521 double normal(double mu, double dist){
522         return mu+0.5*sqrt(6)*( (genrand_real1()/0.5-1)*dist+(genrand_real1
            ()/0.5-1)*dist);
523 }
```

研究業績

本論文の内容に関係の深い発表文献

学術論文

- Kouhei Ito, Akihisa Ohya, Naohiro Shimaji and Takeshi Aoki, “Reduction of distance drift with temperature in uniaxial laser rangefinder by using multi-echo sensing”, Journal of Robotics and Mechatronics, Vol.27, No.1, Feb. 2015 (採録決定)

査読付き国際会議論文

- Kohei Ito, Justin Han and Akihisa Ohya, “Localization using uniaxial laser rangefinder and IMU for MAV”, 2014 IEEE/SICE International Symposium on System Integration, Dec. 2014, SuP2B.2, pp.706

本論文と異なる内容の発表文献

査読付国際会議論文

- Kohei Ito, Keisuke Yokota and Akihisa Ohya, “Localization of mobile robot by matching three-dimensional data of SOKUIKI sensor and aerial imagery”, 2012 IEEE/SICE International Symposium on System Integration, Dec. 2012, SA-B.3, pp.49-54