コンピュータ理工学特別研究報告書

題目

iRobot Create2 と 360° カメラを使った
巡回監視ロボットの開発

学生証番号　444198

氏名　上村真矢

提出日　平成 30 年 1 月 24 日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

本研究では，iRobot Create2をロボットプラットフォームとし，RICOHのTHETA Sを監視・異常発見用のカメラとして用いる．これらをRaspberry Piで制御することで，巡回監視ロボットを開発する．開発するシステムは，ROS(Robot Operating System)というロボットソフトウェアプラットフォームを用いて巡回監視ロボットの動作を制御する．また，THETA Sにより撮影された画像から人物を検出した場合に，検出結果の画像を記録することで，異常発見の機能を備える．この巡回監視ロボットを自動走行させ，本学の14号館2階の廊下全体を巡回監視させることができた．

THETA Sで撮影した生画像は魚眼レンズで撮影した2枚の円形画像である．これらを正距円筒図法によって1枚の360°パノラマ画像に変換する．この画像に対して，OpenCVのHOGDescriptorクラスを用い，HOG特徴による人物検出を行う．人物を検出すると，検出対象を緑色の矩形で囲む．実験の結果，比較的高い精度で人物の検出を行うことができた．しかし，人物検出処理に10～20秒の時間を要する．人物検出をリアルタイムに行えるようにすることは課題である．

ROSにCreate2の走行を制御する関数が提供されている．これらを利用することで，直進・後退・回転動作の指示や速度設定が可能である．そして，Create2の右側4個の赤外線センサとバンパセンサの情報をもとに，次のようなルールで制御した．

① 右端の赤外線センサが障害物や壁を検知している間，直進する．

② バンパの右側もしくは左側が接触した場合，少し左に回転しながら後退する．

③ 右から2～4個目の赤外線センサが壁や障害物を検知している間，少し左に回転しながら前進する．

④ 赤外線センサおよびバンパセンサが障害物を検知しない場合，少し右に回転しながら前進する．

実験の結果，人体を検出させながら廊下を一周させることができた．しかし，下り階段の段差にさしかかる箇所で，段差を回避して走行を継続させることはできなかった．この箇所にだけ，仮の壁を設置して走行させた．下り段差を回避しながら自動走行させることが課題である．

目次

注）枠線は最後に非表示にする.

1章 序論

1973 年より日本の出生率は下がり始め，少子高齢化や人口減少が社会問題として取り上げられている．また就労しない若者が増えていることもあり，就労人口の減少は確実視されている．一方で治安の悪化により，セキュリティに対する要求が高まっている．警備業は労働集約型企業と呼ばれているが，優秀な若い労働力の確保が難しい現状において，警備の効率化は大きな課題である．その解決策の一つとして，警備ロボットの開発が始められた．

警備ロボットは，ビジネスとして発展することが期待されているロボットの一つである．ビル，工場，店舗などの施設を警備する際，警備員を置かずに警備用センサを設置して監視する機械警備を導入する場合も多いが，事故の予防性や即応性などの点から，警備員を必要とする施設も多い．警備員を必要とする施設にロボットを導入し，巡回監視等の作業をさせることにより、以下のような効果が期待できる．

（1） 警備員をロボットで代替することによる運用コストの削減
（2） 就労人口の減少による警備員不足への対応
（3） 危険な作業や長時間作業をロボット化することによる警備員の勤務環境の改善
（4） 受傷事故の回避

警備は巡回監視やボディガード，交通整理など様々な種類に分けられるが，その中で施設の警備業務は実現が比較的容易である．施設警備作業には，夜間に施設の内部や外周の巡回監視をする業務や，昼間に来訪者の受付案内や出入管理を行う業務などがある．これらの業務は長時間にわたって見回りや監視をするので，肉体的にも精神的にも負担が大きくなっている．このため巡回監視業務をロボットに行わせる要求が大きい．巡回監視を行うロボットは，自律走行，異常発見，状況判断，事故処理といった機能を必要とする．自律走行や異常発見などの機能を持ったロボットは存在する（図 1.1）が，全ての機能を兼ね備えたロボットはまだ開発されていない．足りない機能を，人間や他の手段が補完するシステムになっている[1]．

図 1.1　ALSOK の警備ロボット「Reborg-X」

本研究では，iRobot 社の Create2 をロボットプラットフォームとし，RICOH の THETA S を監視・異常発見用のカメラとして用いる．これらを Raspberry Pi で制御することで，巡回監視ロボットを開発する．開発するシステムは，ROS(Robot Operating System)というロボットソフトウェアプラットフォームを用いて制御する．また，THETA S が撮影した画像から人物を検出した場合に，検出結果の画像を記録することで，異常発見の機能を備える．この巡回監視ロボットを自動走行させ，本学の 14 号館 2 階の廊下を右回りに周回監視させることが本研究の目標である．

本論文は次のように構成される．2 章では，本研究で用いる ROS について述べる．3 章では巡回監視ロボットの開発方法や重要な処理について説明する．4 章では巡回監視ロボットの動作を検証し，その結果と考察を述べる．5 章で結論を述べる．

2章　ROS (Robot Operating System)

この章ではロボットソフトウェアプラットフォームとROSの概要,本研究において必要なROSの基本知識について説明する．これにより，ROSについての理解を深める.

2.1　ロボットソフトウェアプラットフォーム

従来のロボット開発において,ロボットのアプリ開発者は,リアルタイム性が高いオペレーティングシステムの上に,ハードウェア構成を強く意識した専用ソフトウェアを開発してきた.しかし,ハードウェアの違いをミドルウェアで吸収することができれば,アプリ開発者はアプリケーションソフトウェアの開発に集中できる.そこで,ロボットの種類によらずセンサやモータなどのロボットハードウェアの制御をつかさどるソフトウェアの開発が始まった.このようなソフトウェアをロボットソフトウェアプラットフォームと呼ぶ．現在では多くの種類が開発されている.

以下に，代表的なロボットソフトウェアプラットフォームの例と開発母体を示す.

・ERSP, Evolution Robotics
・MSRDS, Microsoft
・MARIE, LABORIOUS
・OpenRTM, 産業技術総合研究所（AIST, 日本）
・ROS, Open Source Robotics Foundation（OSRF, 米国）

ロボットソフトウェアプラットフォームを利用すると,ハードウェアに関する知識をそれほど持たなくても，ロボット用アプリケーションの開発が可能である．以前は，ロボット開発者，あるいはロボットメーカは，ハードウェアの設計からソフトウェア開発までの全てを手掛けていた．一方，ロボットソフトウェアプラットフォームに準拠するロボットを開発するのであれば,ソフトウェアに精通した多様な人材がロボットアプリケーション開発に参加できる．また，ハードウェア開発者は，ソフトウェアプラットフォームが提供するインターフェースに合わせてハードウェアを設計すればよい.ロボットメーカはアプリケーションの開発時間や管理コストを削減できる．このように，ロボットソフトウェアプラットフォームはロボット産業の水平分業化を促進し,ロボット分野を発展させることを可能にする.

上で説明したロボットソフトウェアプラットフォームの中で，ROSはユーザ数，提供ライブラリの種類と数，拡張性の高さ，開発の容易さなどの点で優れている．ROSコミュニティは世界各地で活発に活動しており，ROSコミュニティが中心となって蓄積したWeb上の数多くの資料から，ROSの利用中に生じた疑問点や不具合の情報を容易に得ることができる．さらに，ROSはOSRFが単独で開発しているのではなく，大学の研究者，

企業の開発者，趣味で活動する技術者や，さらにはロボットを専門としないコンピュータ科学者，コンピュータビジョンや通信ネットワークの専門家など，多様な人々が開発に携わっている点が特徴である．

2.2　ROS の概要

ROS は，米国のスタンフォード大学人工知能研究所（AILAB）が実施した STAIR（STanford AI Robot）プロジェクトにおいて，Morgan Quigley が開発した Switchyard が発端である．そして，2007 年 11 月，米国のロボット専門企業 Willow Garage が Switchyard を引き継ぎ，ROS の名前で開発を始めた（図 2.1）．現在では，ROS の開発はロボット向けのオープンソースソフトウェア財団である OSRF に移譲された．ROS は，フリーソフトウェアの代表的なライセンス体系である BSD ライセンス（BSD 3-Clause License）をベースにしており，だれでも修正，再利用，再配布が可能である．開発者とユーザを対象にしたカンファレンス（ROSDay，ROSCon）が定期的に開催され，アメリカとヨーロッパでは ROS Industrial（産業用途の ROS）を開発しているコンソーシアムがある．また、それぞれの国で，ROS Meetup と呼ばれるコミュニティが，セミナーなどのイベントを開催している．日本でも，東京オープンソースロボティクス協会（TORK）や ROS Japan Users Group が中心になってセミナーや勉強会を開催し，ROS の普及に努めている．

Personal Robot2 や TurtleBot など，ROS の利用を前提としたロボットの開発も行われている．これらのロボットを利用して多くのアプリケーション開発が行われ，ROS は最も標準的なロボットソフトウェアプラットフォームである．

図 2.1　ROS のロゴ

Robot Operating System という名前ではあるが，Windows や Linux，Android，iOS などの汎用コンピュータ向けのオペレーティングシステムと同様なソフトウェアではない．ROS は，メタオペレーティングシステムといった位置づけになる．メタオペレーティングシステムとは，オペレーティングシステム上で動作し，同一，あるいは異なるオ

ペレーティングシステムで動作しているプロセスに対し,プロセス間通信やスケジューリング,負荷の監視,エラー処理などを支援するシステムである.つまり ROS は,Windows や Linux などに取って代わるロボットのためのオペレーティングシステムではなく,上記のオペレーティングシステムで動作するプロセスを制御するシステムである(図2.2).

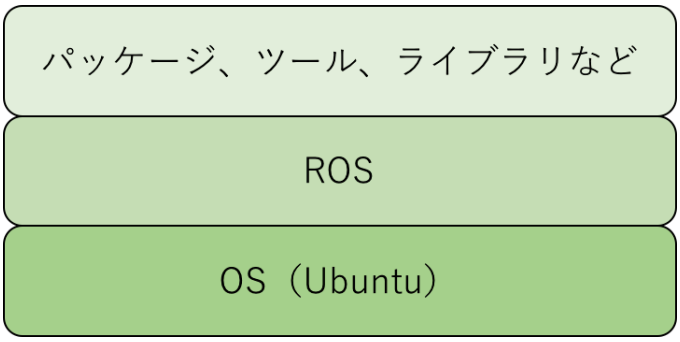

図 2.2　ROS の立ち位置

ROS が正式にサポートしているオペレーティングシステムは Ubuntu である．Ubuntu のプロセス管理システム，ファイルシステム，ユーザインタフェース，プログラムユーティリティなどを利用している．ROS は，これに加えて様々なセンサやロボットのハードウェア制御，プロセス間通信，スケジュール，エラー処理など，ロボットアプリケーションの開発に必要な機能をライブラリとして提供している．また，ROS を利用して開発された様々なアプリケーションパッケージを管理し,流通する仕組み(エコシステム)を備えている．このように既存の伝統的なオペレーティングシステムを利用しながら,ロボットアプリケーションの開発に必須となるロボットやセンサの制御システムを,ハードウェア抽象化の概念に基づいてパッケージ化したものである.さらに様々なアプリケーションもパッケージとして提供することで,ユーザによるロボットアプリケーション開発を強力に支援する．また，ROS でプロセス間通信に使用されるメッセージは，異なるオペレーティングシステムや異なるハードウェア上のプロセス間でも自由に情報をやり取りすることができる．したがって ROS は，様々なハードウェアから構成される複雑なロボットの開発にも適したソフトウェアプラットフォームである.

2.3　ROS の基本知識

ROS で用いられる専門用語の中で，本研究で必要な用語について説明を行う.

・ノード（Node）

Linux のプロセスに相当するもので，プログラムの最小単位である．ノード毎の独立したメモリ空間内でプログラムが動作しており,多くのシステムで繰り返し利用できる

ように設計される．例えば本研究で用いるCreate2の場合，センサドライバ，センサデータ処理，障害物検出，モータ駆動など，動作に必要な処理を細分化し、それぞれの処理を実装したノードを組み合わせてシステムを構築する．ノードを実行する際には，配信者，購読者，トピック，サービス，メッセージの形式，URIアドレスとポートをマスタに登録する．これらの情報を基に，ノード間のトピックメッセージ通信とサービスメッセージ通信を利用して情報を送受信する．ここで配信者はマスタにトピック名などの情報を登録して，対応した各トピックにメッセージを送信する．購読者はマスタからトピック名などの情報を得て，対応した各トピックのメッセージを受信する．トピック・サービスメッセージ通信については，以下で説明している．

・マスタ（Master）

マスタはノードやトピック等の名前を管理しており，ノード間接続を始める前に，ノード間がどのトピックでつながっているかを設定する．接続元・接続先のノードが見つかれば，その後は，1対1で直接通信する．このような機能を持つため，マスタはノードより先に起動する必要があり，かつ，システム中に複数のマスタは存在できない．

・パッケージ（package）

ROSにおけるパッケージは一単位のソフトウェアのことである．

・launchファイル

複数のノードをまとめて実行するには,「.launch」という拡張子のファイルを用いる．roslaunchコマンドで対象のlaunchファイルを指定することで，複数のノードをまとめて実行することができる．

・メッセージ

トピックやサービスによる通信において，ノード間でやり取りされる情報である．整数型，浮動小数点型，論理型などの変数で構成されている．

・トピック

一方向で非同期方式のメッセージ送受信方式である．配信者が配信したいメッセージをトピックに登録する．その後，トピックに登録されているメッセージの受信を希望する購読者が配信者の情報を購読する．これによりメッセージのやり取りを行う．トピックは継続してメッセージを送信しなければならないセンサデータの通信処理に適している（図2.3）．

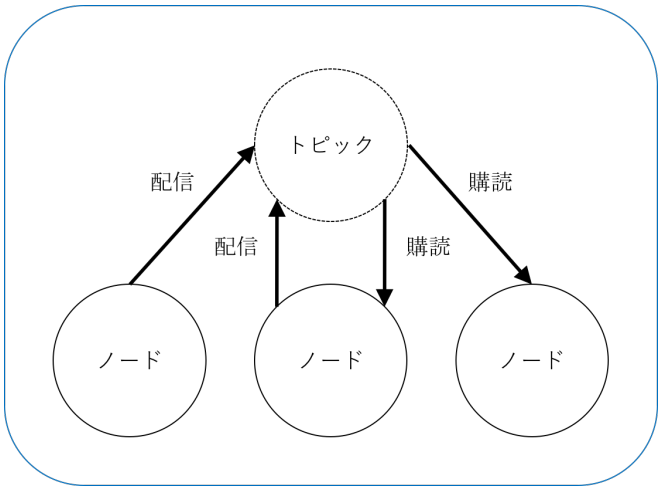

図 2.3　トピックによる通信モデル

・サービス

同期式のメッセージ通信方式である．リクエストを受け取るとレスポンスを送信するサーバと，リクエストを送信しレスポンスを受け取るクライアントから構成される．サービスは受け取ったメッセージ（リクエスト）に対して何らかの結果（レスポンス）を返すことができる（図 2.4）[2]．

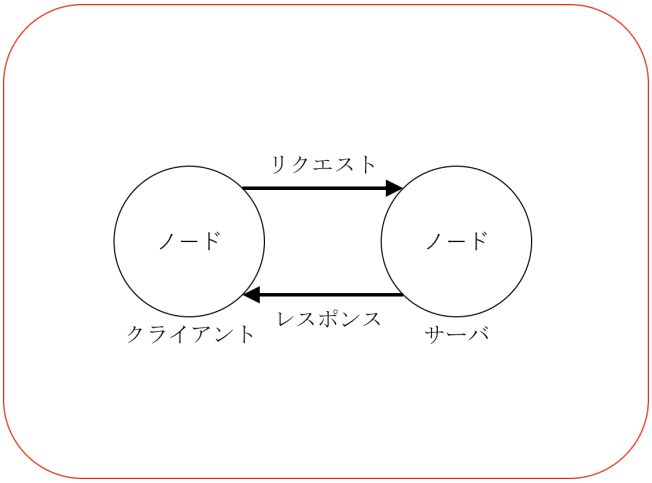

図 2.4　サービスによる通信モデル

これらの用語を具体的なロボットシステムと対応させるならば，ノードはアームやレッグなどのロボットパーツを動作させるそれぞれのプログラムに対応する．マスタは個別のプログラムが，ロボットシステム全体の中でどのように関係するかを示した接続図に対応すると言える．

3 章　巡回監視ロボットの開発

本研究では，Raspberry Pi を使った組込みシステムで巡回監視ロボットを開発した．巡回監視ロボットのプラットフォームとして iRobot 社の Create2 を利用した．これは掃除ロボットであるルンバから掃除機能を取り除いたもので，外部 PC をシリアル通信で接続することで，動き・音・LED の光り方などを制御可能な研究向けロボットである．また，360°パノラマ画像を取得可能な RICOH の THETA S を監視・異常発見用のカメラとして用いる．これらを Raspberry Pi に接続し，巡回監視ロボットを開発した（図 3.1）．

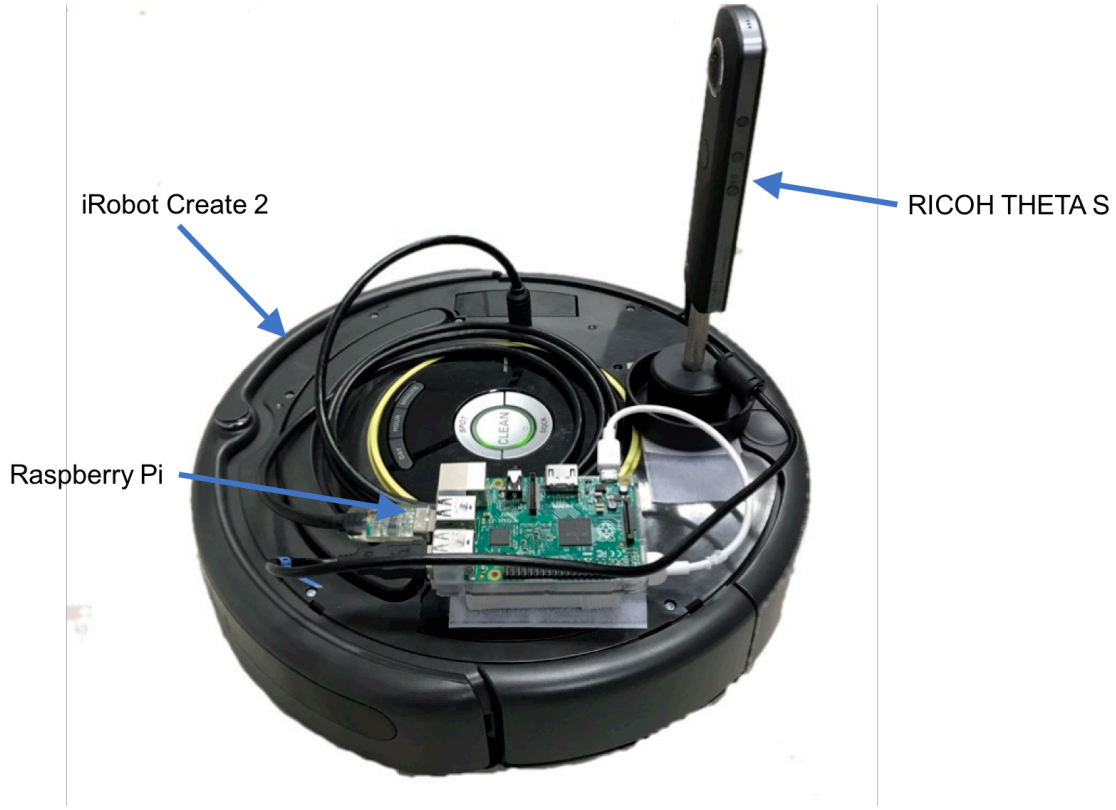

図 3.1　巡回監視ロボットの外観

Create2 を制御するコマンドは OI（Open Interface）とよぶものである．ROS のパッケージに OI をもとに作った Create2 用の ROS ドライバと C++関数があり，それらを利用した．C++の関数を呼び出すことで，Create2 に備わっているバンパセンサやバンパセンサ内に埋め込まれた 6 個の赤外線センサ，4 個の段差センサ（図 3.2）などのデータを取得することや走行制御が可能になる．

ROS と上で述べたパッケージを利用するために，ROS が動作する OS を用いる必要がある．今回は，ROS が公式サポートしている Ubuntu を Raspberry Pi にインストールした．

図 3.2　Create2 の各種センサ位置

3.1　巡回監視ロボットの走行制御

Create2 の走行制御に関する関数は，直進・後退・回転動作の指示や速度の設定が可能である．そして，赤外線センサやバンパセンサの状態に基づいて適切に走行状態を制御する．今回は図 3.2 の右から 4 個の赤外線センサとバンパセンサをもとに走行制御を行った．その制御ルールの概要を次に示す．

① 右端の赤外線センサが壁や障害物を検知している間，直進する．

② バンパの右側もしくは左側が接触した場合，少し左に回転しながら後退する．

③ 右から 2～4 個目の赤外線センサが壁や障害物を検知している間，少し左に回転しながら前進する．

④ 赤外線センサおよびバンパセンサが壁や障害物を検知しない場合，少し右に回転しながら前進する．

上記の制御ルールを用いて，14 号館 2 階の廊下を右側の壁沿いに自動走行させる（図 3.3）．

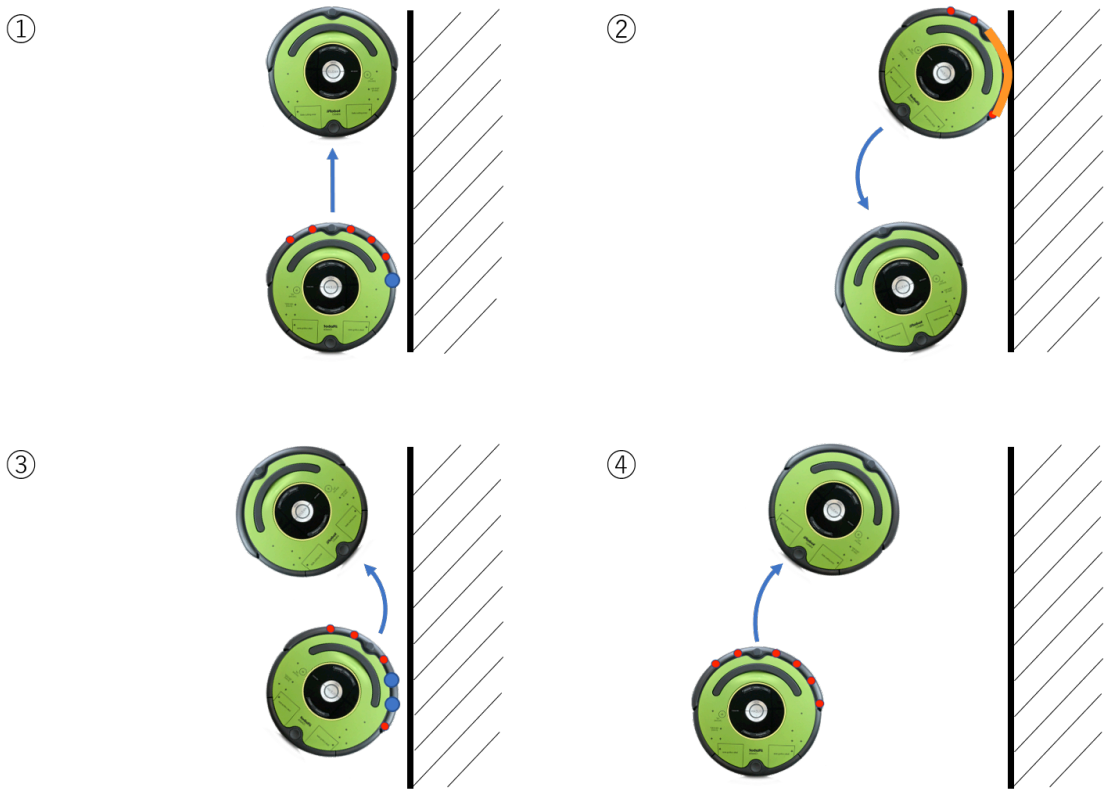

図 3.3　制御ルールにしたがう走行の例

3.2　THETA S の生画像から 360°パノラマ画像への変換

USB カメラとして用いた THETA S で撮影すると，図 3.4 のように，2 枚の魚眼レンズで撮影した円形画像を 1280×720 画素に収めた画像が出力される．これを正距円筒図法による 360°パノラマ画像に変換する方法を説明する．正距円筒図法は球面を平面に展開する図法の一つで，図 3.5 のように，緯度・経度の線が直交し均等間隔になる．画像の縦横比は 1:2 である．

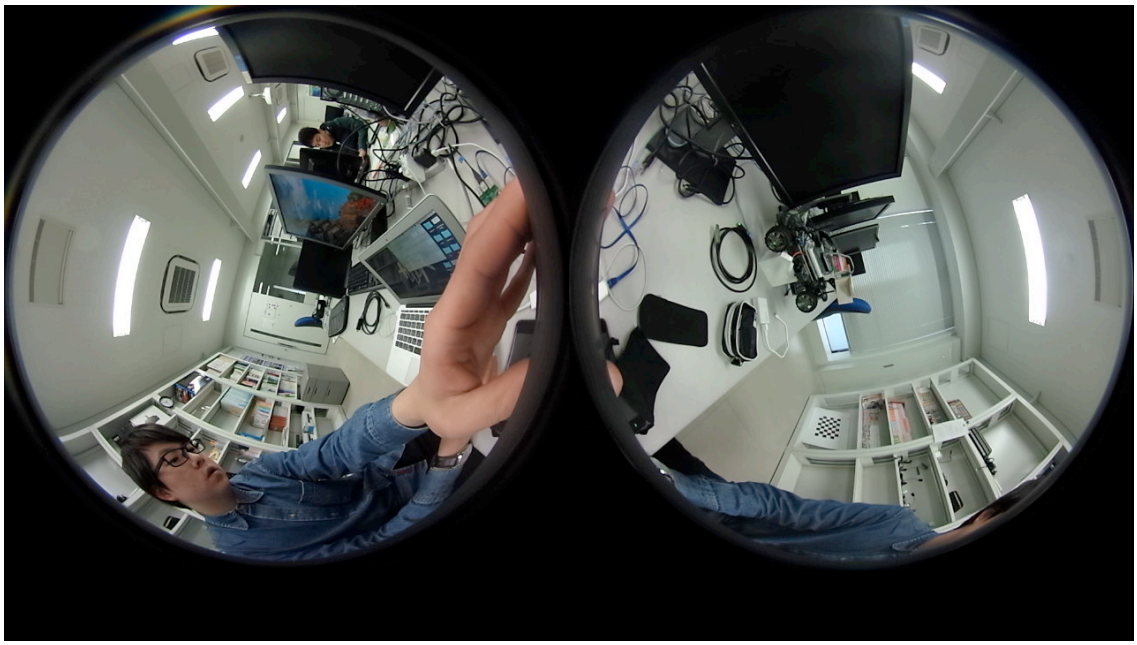

図 3.4　THETA S の生画像

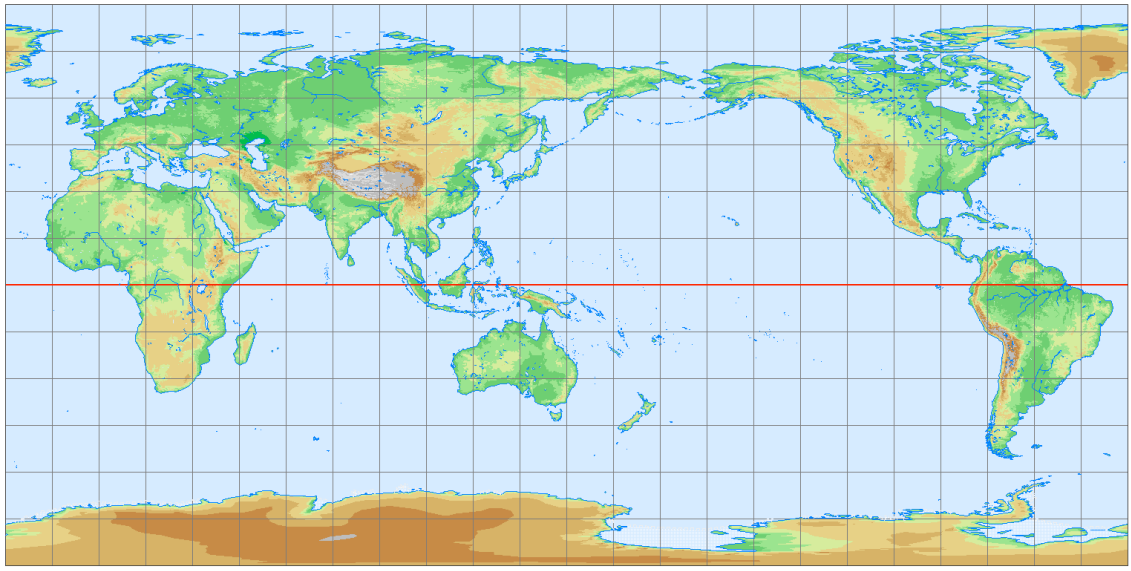

図 3.5　正距円筒図法で作成された世界地図の例

まず，前処理として図 3.4 の入力画像から魚眼レンズで撮影した領域を 2 つの正方形画像として抽出する．抽出した画像を図 3.6 に示す．図 3.6 の左の画像は元画像から，座標（315, 322）を中心として一辺 573 画素の正方形領域を抽出し，時計方向に 90° 回転したものである．図 3.6 の右の画像は，座標（960, 320）を中心として一辺 573 画素の正方形領域を抽出し，反時計方向に 90° 回転したものである．

図 3.6　生画像から魚眼レンズで撮影した領域を抽出し，方向を揃えた 2 枚の画像

次に幾何変換について説明する．球面上の位置を経度（θ）・緯度（φ）で表現した時，正距円筒画像の座標は横軸を経度，縦軸を緯度に対応させた直交座標になる（図3.7）．

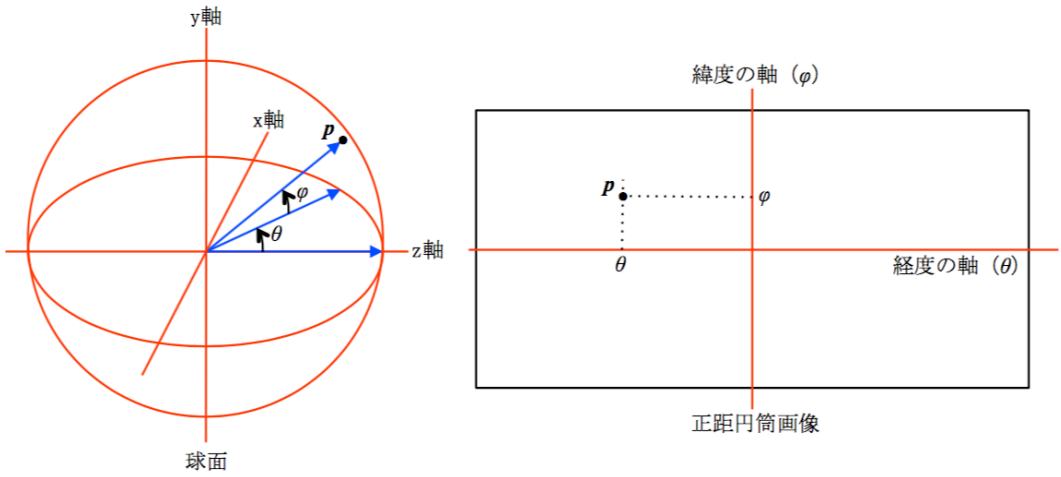

図 3.7　球面上の点と正距円筒画像上の点の関係

魚眼レンズは，180° 程度の視野を1枚の画像に投影するレンズである．図3.7の左の球を半径1の単位球と考え，球面上の点***p***が魚眼レンズによって，どのような投影を受けるかについて考える．

図3.8は図3.7の左の図に説明を追加したものである．点***p***はz軸方向の単位ベクトル（0, 0, 1）を水平方向に θ 回転させ，次いで，垂直方向に φ 回転させたものである．（0, 0, 1）を θ 回転させると（sin θ , 0, cos θ ）に移動する．（sin θ , 0, cos θ ）を垂直方

向にφ回転させると（sinθcosφ, sinφ, cosθcosφ）になる．したがって，点***p***の3次元座標は（sinθcosφ, sinφ, cosθcosφ）である．

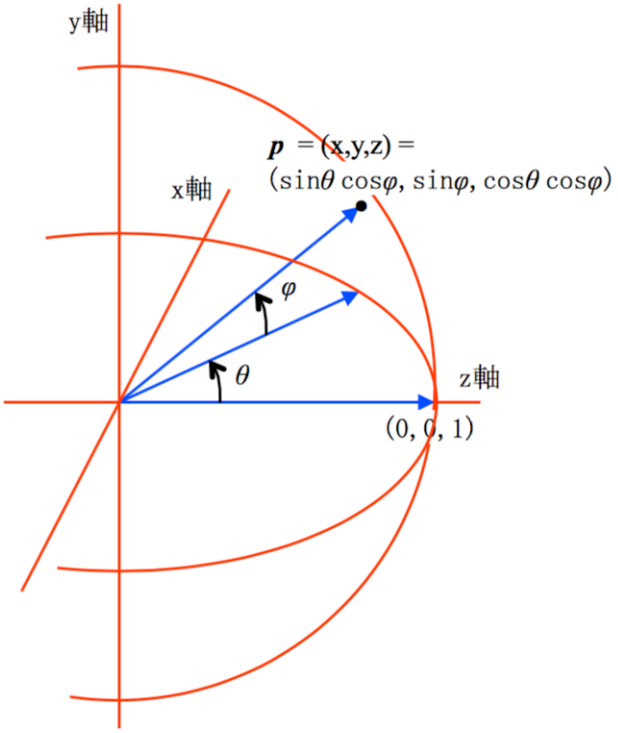

図3.8　単位球面上の点の3次元座標

次に単位球面上の点***p***（x, y, z）が，魚眼レンズによって画像上のどの位置に投影されるかを考える．ここで説明を単純化するために，魚眼レンズの撮影範囲は180°（-90°～90°），魚眼レンズで撮影した画像の座標（u, v）は-1～1の範囲に正規化されているとする．図3.9の左は正規化された魚眼レンズの投影面，図3.9の右はTHETA Sにおける投影のモデルである．魚眼レンズの投影にはいくつかの方式があり，THETA Sは立体投影方式を採用している．立体投影方式では，魚眼レンズに入射する光線の角度をγとしたとき，図3.9の右に示すように，tan（γ/2）の位置に像が形成される．これらのことを考慮すると，単位球面上の点***p***（x, y, z）に入射した光線の像***q***の位置を，次のように計算することができる．

① 入射光線の角度γを求める．これはベクトル（x, y, z）とベクトル（0, 0, 1）の成す角度である．

② ***q***の座標は（x, y）の定数倍であり，原点からtan（γ/2）離れた位置である．

以上で，魚眼レンズで撮影した画像の座標と正距円筒画像の座標を関係付けることができる．これを用いて，図3.6の2枚の画像を幾何変換し，貼り合わせることにより360°パノラマ画像が生成される（図3.10）．

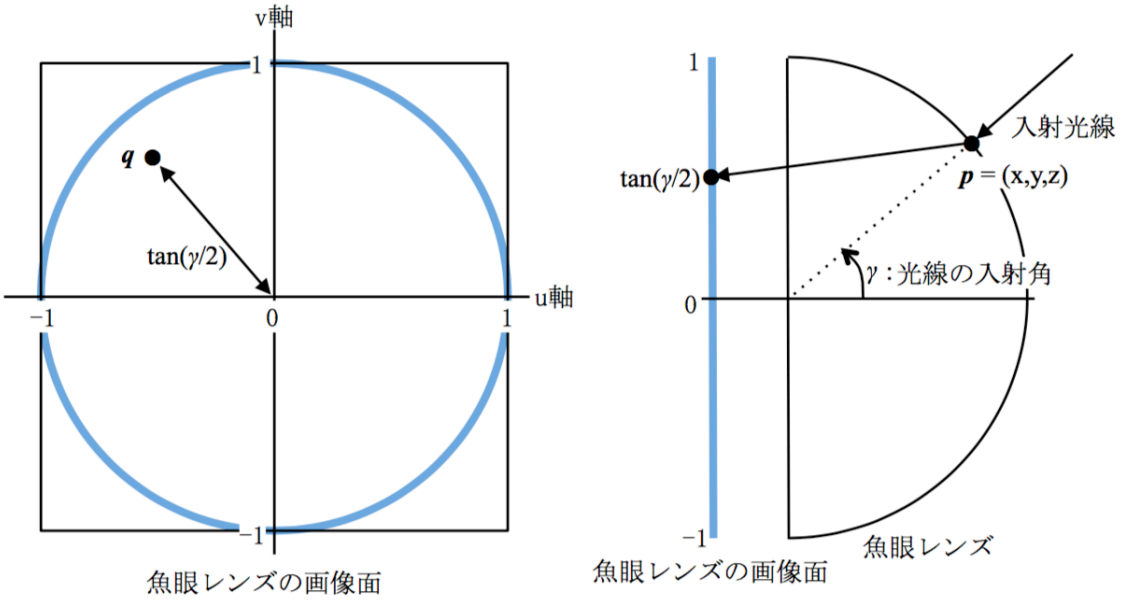

図3.9　魚眼レンズの正規化された画像座標と投影のモデル

図3.10　THETA Sの生画像から生成した360°パノラマ画像

3.3　HOG 特徴による人物検出

一般物体認識のための勾配ベースの特徴量として HOG（Histograms of Oriented Gradients）特徴が提案されている．HOG は局所領域における輝度の勾配方向をヒストグラム化した特徴量である．ある大きさを持つ領域に対して特徴量を記述するため，おおまかな物体形状を表現することが可能である．人物検出や車検出などの一般物体認識に用いられる．

本研究では，OpenCV にある HOGDescriptor クラスを用いることで，HOG 特徴による人物検出を行う．HOGDescriptor は，被探索画像に対して 64×128 の探索窓を 8 ピクセル単位で移動させてスキャンし，HOG 特徴の集まりを生成する．また，setSVMDetector 関数で，あらかじめ作成した人検出用の分類データを読み込んで，SVM 分類器を用意しておく．

今回は，HOGDescriptor クラスにある人検出用の分類データを利用した．人検出用の分類データには，getDefaultPeopleDetector を用いた．検出の実行には，被探索画像と探索窓との相対的なサイズを変えながら検出を繰り返す detectMultiScale 関数を用いる．ここで HOG 特徴の集まりと検出用データを比較し，人物検出をする．人物を検出すると，検出対象を緑色の矩形で囲む（図 3.11）．この処理を，THETA S の画像を 360°パノラマ変換した後の画像に対して行い，人物を検出する．

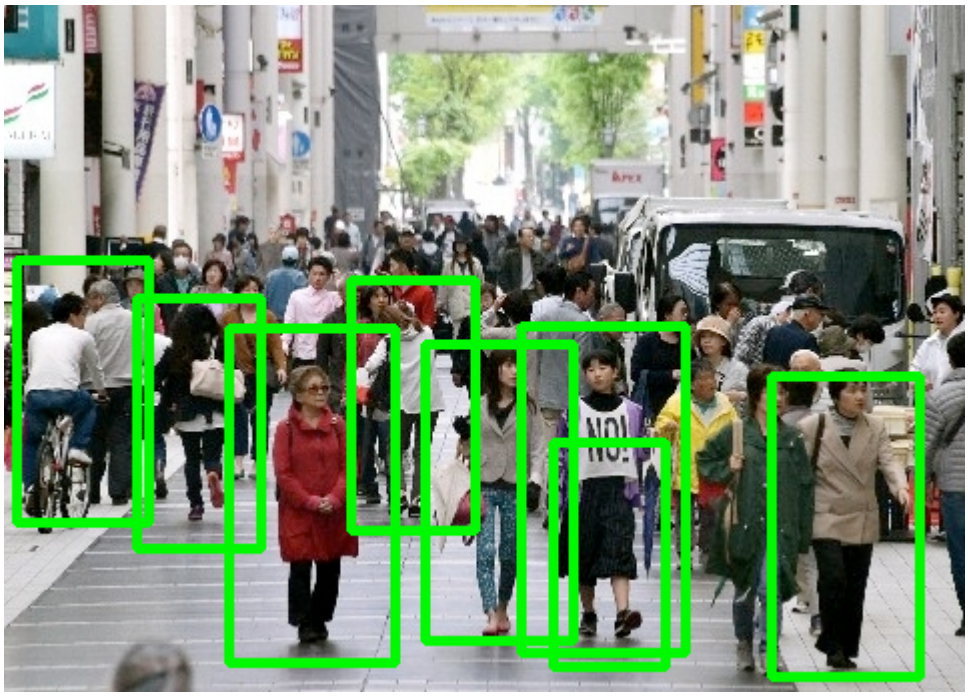

図 3.11　HOGDescriptor による人物検出結果の例

4 章　実験結果と考察

開発した巡回監視ロボットの動作を 14 号館 2 階の廊下を走行させることで検証した．

4.1　人物検出の実験結果と考察

14 号館 2 階の廊下を走行させ，15 秒間隔で人物検出処理を行った．人物を検出した際に記録した画像の例を図 4.1，図 4.2 に示す．

図 4.1　14 号館 2 階廊下での人物検出結果（1）

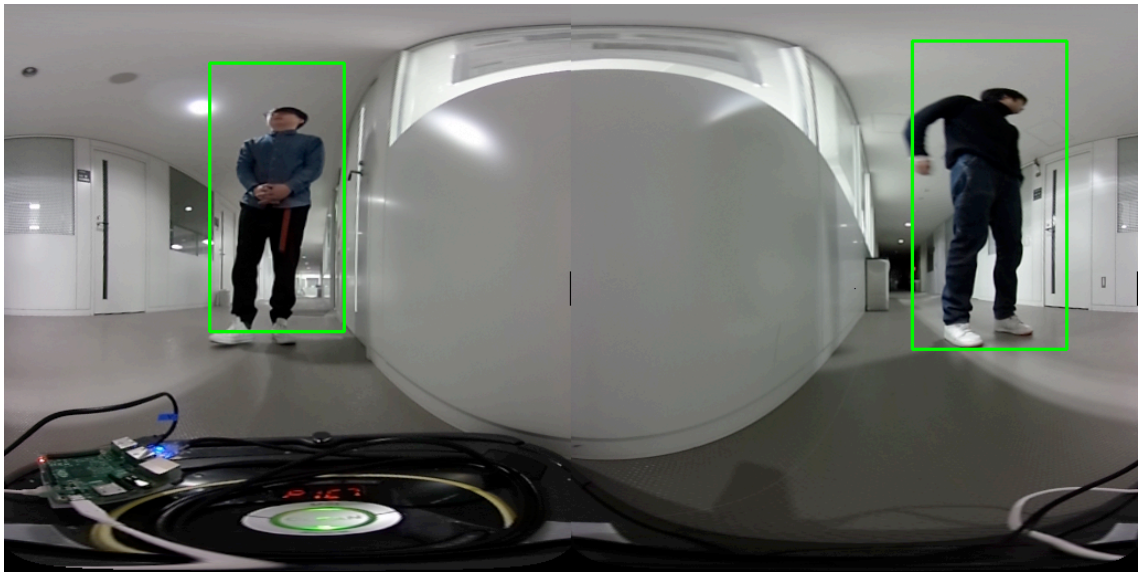

図 4.2　14 号館 2 階の廊下での人物検出結果（2）

これらの結果から人物を正しく検出できていることがわかる．このように比較的高精度な検出ができたのは，人物と紛らわしい物体が少ないためだと考えられる．また，走

行を停止させてから撮影，人物検出処理を行い，走行を再開するまでに 10 秒～20 秒の時間を要した．

本研究で開発した巡回監視ロボットの人物検出において問題であると考えた点は，15 秒間隔での人物検出処理とその処理に要する 10～20 秒の時間である．巡回監視という業務において，これらの時間の間，異常を発見できないことは致命的である．人物検出処理をリアルタイムに行えるようにすることが課題である．

4.2　自動走行の実験結果と考察

本研究では 14 号館 2 階の廊下を内側の壁沿いに一周させることを目標とした．今回は右回りに走行することを想定して実験を行なった．図 4.3 に 14 号館 2 階廊下の全体と，その中で問題のあった箇所を赤い星で示す．

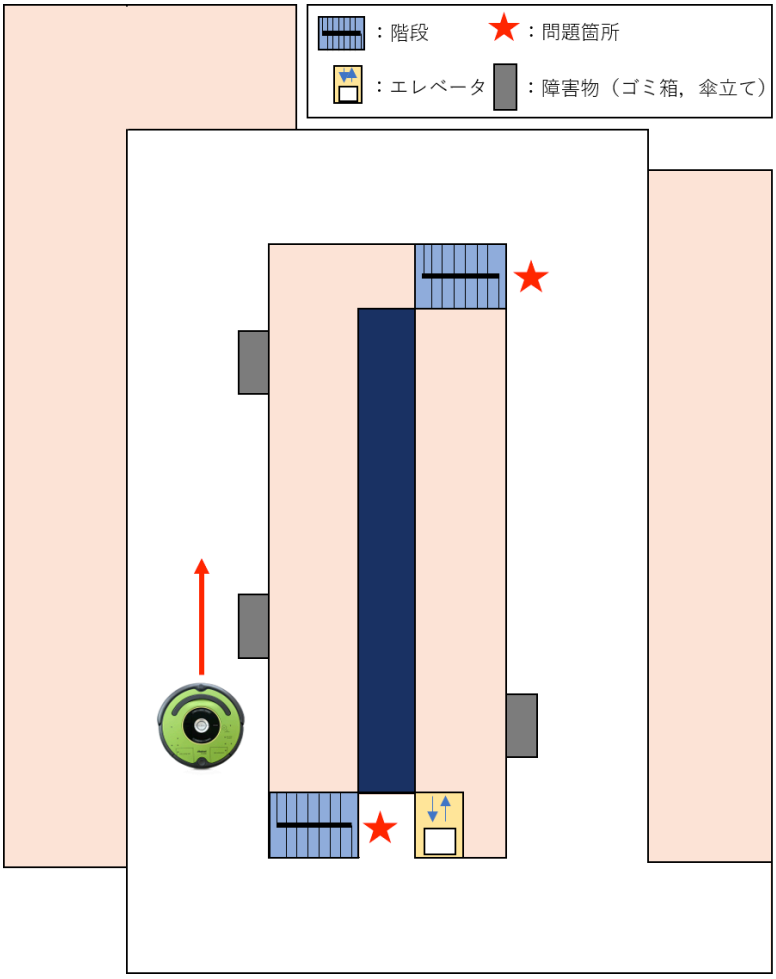

図 4.3　14 号館 2 階廊下の全体と問題のあった箇所を表した図

巡回監視ロボットを 14218 室の前から開始させた．経路上にある灰色の四角で描かれ

た障害物に対して，次のように動作した．

(1) ゴミ箱や傘立は赤外線センサが検知し，正しく回避して走行した．

(2) ガラスや黒い壁は赤外線が透過・吸収されてしまう材料であったため，赤外線センサによる検知はできない場合が多かった．しかし，バンパセンサによる検知で対処することができた．

(3) 下り階段の段差を回避して走行させることが必要であった．段差センサで段差を検出できたが，回避走行させることができず，走行を停止させた．

本研究によって，自動走行がいかに高度な課題であるかを実感した．開発した巡回監視ロボットの走行制御に関しては，段差部分に一時的な壁を設けることにより，廊下を一周させることができた．しかし，これでは自動走行とはいえない．走行制御を変えずに段差を回避する方法として，iRobot 社がルンバの動きをアシストするために開発した「バーチャルウォール」（図 4.4）を用いることが考えられる．これは赤外線の壁を作ることのできる装置であり，Create2 でもその赤外線を読み取ることができる．今回はバーチャルウォールを用意できなかったが，これを段差部分の手前に，並行に設置することで段差の回避を行うことが可能になる．

図 4.4　iRobot 社のバーチャルウォールの外観

5章　結論

360°カメラのTHETA Sを接続したRaspberry PiをCreate2に搭載し，巡回監視ロボットを開発した．Raspberry Piにインストールした ROSを利用して，Create2に備わっている赤外線センサ，バンパセンサの情報から障害物を検出し，14 号館 2 階の廊下を巡回監視させた．ゴミ箱や傘立てのように赤外線センサが検知できる障害物は，障害物に衝突することなく回避できた．ガラスや黒い壁の場合，赤外線センサによる検知ができなかった．こういった障害物の場合，バンパセンサによる検知で対処できた．14 号館2階の廊下に2箇所ある下りの階段の段差は，段差センサで検知できた．しかし，回避して走行を継続することができず，停止して落下を防ぐことにとどまった．また，15秒間隔で走行を停止させ，人物検出を行う処理においても，比較的高い精度で人物の検出を行うことができた．しかし，THETA S で撮影し 360°パノラマ変換した画像には，ずれや歪みが生じているので，人の検出に成功している場合であっても，正確に人の全身を矩形で囲めていない場合があった．

今後の課題としては，段差を検知し回避しながらの走行を可能にすること，走行しながらリアルタイムで常に人物の検出を可能にすることなどが挙げられる。こういった課題を解決することで，巡回監視への実用性をより高めることができる．

結論として，本研究でCreate2の制御に用いたROSというロボットソフトウェアプラットフォームを利用することで，容易に巡回監視ロボットの動作処理を実装することができた．この研究を通して，ROSは今後のロボット開発において，実用性の高さや開発の効率化といった点で有用であると考えられる．

参考文献

[1] 下笹洋一,「警備ロボットビジネス」, 日本ロボット学会誌, 20 巻（2002）7 号 p. 692-695

[2] 倉爪亮, 渡邊裕太,「詳説 ROS ロボットプログラミング」, Kurazume Laboratory, 2015 年 11 月 30 日（初版）

謝辞

本論文を作成にあたり,丁寧な御指導を賜りました蚊野浩教授に感謝いたします.

付録　本研究で開発したプログラム

[プログラム名]

observe_robot.cpp

[内容]

このプログラムは，Create2 と THETA S を Raspberry Pi に接続して，巡回監視ロボットとして制御するための組み込みシステムである．プログラムの動作を，「MOVE モード，WAIT モード，PICT モード」の 3 つのモードに分けて説明する．

・MOVE モード

Create2 の赤外線センサやバンパセンサにより壁や障害物を検知する．検知したデータをもとに Create2 のステアリング制御を行い，右側の壁沿いを走行する．約 15 秒走行すると，走行を停止させ自動的に PICT モードに移行する．Create2 の SPOT ボタンを押す，段差センサにより段差を検知する，脱輪すると WAIT モードに移行する．Create2 の CLEAN ボタンを押すことによってプログラムを終了する．

・WAIT モード

このモードの間は常に走行が停止している状態で，プログラム起動時は WAIT モードである．Create2 の SPOT ボタンを押すと MOVE モードに移行する．CLEAN ボタンを押すとプログラムを終了する．

・PICT モード

走行を停止させた状態で THETA S により撮影をする．そして，撮影された生画像を 360°パノラマ画像に変換する．変換後の 360°パノラマ画像をもとに，人物検出を行う．人物を検出した場合，検出結果を示した画像を Raspberry Pi に保存する．人物を検出しなかった場合，画像は保存されない．また人物検出の有無は，それぞれ効果音が鳴るようになっている．全ての処理を終えると，自動的に MOVE モードに移行する．