2018 年度（平成 30 年度）

創造工学セミナーⅡ

Final Report

ステレオカメラによる距離計測を用いた移動ロボットの製作

指導教員　：　金丸　隆志　准教授

チームメンバー

S5-15001　相澤　優介

S5-15019　柿沼　日向

S5-15021　金子　　悠

S5-15058　松田　竜騎

目次

第 1 章　緒言（松田担当）.....1
1.1　研究背景.....1
1.1.1　研究テーマ.....1
1.1.2　ステレオカメラ.....2
1.1.3　Ovrvision Pro.....4
1.2　研究の概要.....5
1.2.1　作成物の決定.....5
1.2.2　他社製品との比較.....6
1.2.3　研究の目標、目的.....8
1.2.4　研究の進行.....9
第 2 章　目標認識（相澤担当）.....10
2.1　概要.....10
2.1.1　目的.....10
2.1.2　仕様.....10
2.1.3　目標物の設定.....10
2.1.4　使用ソフト.....12
2.2　テンプレートマッチング.....13
2.2.1　物体検出方法の選択.....13
2.2.2　テンプレートマッチングの原理と利点.....13
2.2.3　テンプレートマッチングによる試行結果・考察.....14
2.3　色検出.....16
2.3.1　色の検出と HSV 色空間.....16
2.3.2　色検出による物体検出の試行結果と考察.....18
2.4　座標の出力.....20
2.4.1　座標出力.....20
2.4.2　検出ノイズに除去.....20
2.4.3　検出物体の輪郭の描写と輪郭の測定.....21
2.4.4　考察.....24
第 3 章　距離の測定（相澤担当）.....25
3.1　概要.....25
3.1.1　目標.....25
3.1.2　ステレオカメラによる距離計測の方法・原理.....25
3.2　距離の算出.....28
3.2.1　必要データの収集方法.....28

3.2.2　算出式の作成.....29
3.2.3　評価・考察.....30
第4章　移動ロボットの制御（柿沼担当）.....31
4.1　概要.....31
4.1.1　目的.....31
4.1.2　要求機能.....31
4.2 ルンバの接続と制御.....32
4.2.1　使用機器.....32
4.2.2　ルンバオープンインターフェイス(ROI).....33
4.2.3　接続方法.....34
4.2.4　制御方法.....35
4.2.5　ルンバの回転.....36
4.3　動作確認.....37
4.3.1　Arduino IDE を用いて動作確認.....37
4.3.2　Visual Studio 2017 を用いて動作確認.....38
4.3.3　カメラの測定可能距離を踏まえた動作確認.....40
4.4　最終評価.....41
第5章　周辺パーツの製作（金子担当）.....42
5.1　概要.....42
5.1.1　概要.....42
5.1.2　要求機能・制約条件.....42
5.2　周辺パーツの設計.....43
5.2.1　アイディア.....43
5.2.2　設計.....44
5.2.3　台座の制作.....45
5.2.4　Ovrvision Pro の固定部品の製作.....49
5.2.5　組立.....51
5.3　考察.....53
第6章　動作結果（金子担当）.....54
6.1　動作確認の目的.....54
6.1.1　概要.....54
6.2 動作確認の評価方法.....54
6.2.1　評価目標.....54
6.2.2　動作確認のロケーション.....55
6.2.3　確認手順.....56
6.3　動作確認の結果.....56

6.4　評価.....58
6.5　付加価値の提案.....59
6.5.1　追尾機能の提案.....59
6.6　実験結果.....60
6.6.1　実験方法.....60
6.6.2　実験結果.....61
第7章　結論（松田担当）.....62
7.1　目標.....62
7.2　目標への達成度.....62
7.3　総合評価.....63
第8章　参考文献・URL.....64
謝辞　.....66

第 1 章 緒言（松田担当）

1.1 研究背景

1.1.1 研究テーマ

我々はステレオカメラを用いて「利便性のある移動ロボットを作る」というテーマに取り組む。このテーマは、2016 年度工学院大学卒業論文の「画像処理に基づく自動運搬装置の開発」に続き、ステレオカメラを用いた移動ロボットの考案、製作を行うものである。テーマ遂行にあたり、私たちは図 1-1 のステレオカメラ「Ovrvision Pro」を用いる。

先行研究に用いられたステレオカメラが自作のものであったのに対し、既製品のステレオカメラを用いることでどのような変化が見られるかを調べるのが目的である。

図 1-1 ステレオカメラ「Ovrvision Pro」[1-1]

1.1.2 ステレオカメラ

本項ではステレオカメラについて説明する。

ステレオカメラとは、被写体を同じ方向を向いた 2 つのカメラを持つカメラである。奥行き方向の情報も取得できるという特徴がある。そのため、自動車に衝突防止システムとして搭載されることがある。

図 1-2　ステレオカメラを用いたスバルのアイサイト［1-2］

また、部品メーカーのボッシュの車載ステレオカメラでは、車両の走行可能な範囲(フリースペース)の検知精度が向上している。2 つのカメラを利用した距離の検出や移動体の検知、車線や光源の認識、物体の識別といったアルゴリズムが搭載されている。フリースペースの検知は、運転支援システムや市街地での自動運転技術に必要になり、白線や路肩だけではなく、車道と歩道の境界があいまいな環境にも対応している。また、工事中で車線がパイロンなどで規制され、本来の車線を無視して走行しなければならない場面でも、走行すべきエリアを認識できる。

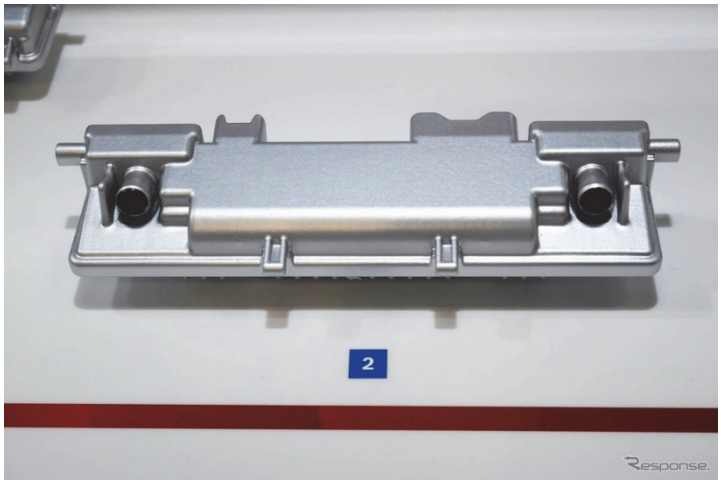

図 1-3　車載ステレオカメラ（ボッシュ）[1-3]

1.1.3 Ovrvision Pro

本項では本研究で使用する Ovrvision Pro について説明する。

Ovrvision Pro は株式会社しのびや.com から出されているステレオカメラである。HTC Vive や Oculus Rift（バーチャルリアリティ向けヘッドマウントディスプレイ）に装着することで没入型 AR（現実の情報をコンピュータ処理によって追加や変化などをさせる技術）やハンドトラッキング（人の行動やシステムの挙動、データの推移などの情報を継続的に収集、監視すること）が可能になるステレオカメラである。高 FPS、高解像度、高視野角、両眼同期、低遅延を実現し、ロボセンサーなどにも活用可能である。ハードウェア面においては 7 段階の解像度、FPS 設定に加えてノイズ軽減対応で産業用カメラと同等な設定項目数により高度な制御が可能である。その他にもソフトウェアの導入により、ゲームエンジン対応も可能である。

主な仕様を箇条書きにすると以下のようになる。

・VR モード推奨カメラ画素数は 1920×950px（片目 960×950）
・USB3.0 搭載で高速通信 5Gbps、高速 60FPS、低遅延 50msec 以下
・総重量 65kg 以下の計量設計、GPIO、EEPROM など組み込み系にも強い設計
・Windows、MAC、組み込み系に強い Linux にも対応
・簡易 AR エンジンなどの SDK を付属、オープンソースで商用利用の連絡なしに利用可能
・OpenCV を活用したハンドトラッキングを実現
・大手ゲームエンジンに対応した汎用性の高い開発環境
・VR モード以外では、解像度、FPS 共に 7 段階の可変モード搭載

図 1-4　Oculus Rift に装着した Ovrvision Pro［1-4］

1.2 研究の概要

1.2.1 作製物の決定

先行研究では、自作のステレオカメラが原因で物体検出や距離の算出が思うように取得できなかったという課題があった。市販のステレオカメラを用いることでこの課題を克服し、より確実に情報を取得できる移動ロボットの製作を行う。

利便性のある移動ロボットの製作にあたって、「正確な物体の検出」、「ステレオカメラによる目標物までの距離計測」、「ロボットの目標地点までの正確な移動と停止」などの機能を持ったシステムの開発を目指す。

このシステムにより、反復作業や単純な手動タスクを自動化することにより、作業負荷の軽減をする。

本研究の使用状況として、使用する場所は室内とする。理由は、天候によって生じる環境の変化を防ぎたいからである。

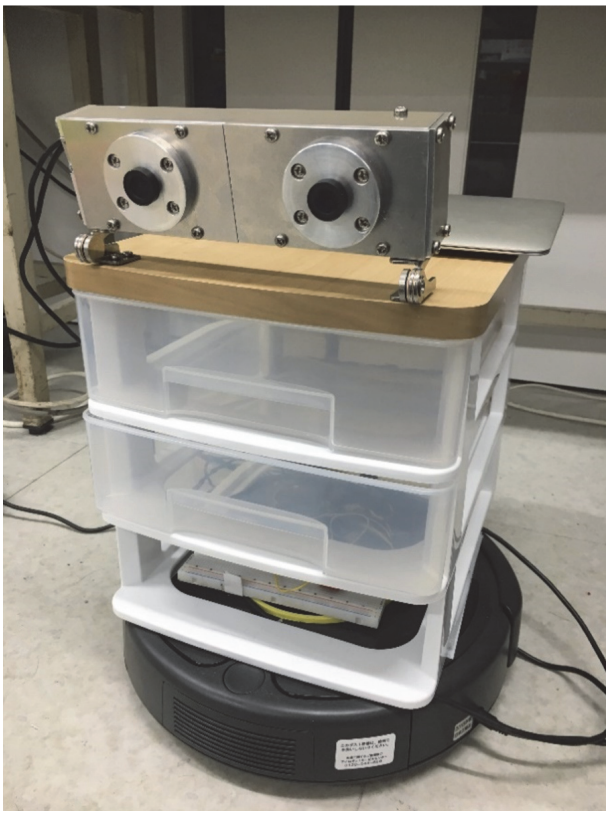

図 1-5　先行研究の自動書類運搬装置［1-5］

1.2.2 他社製品との比較

私たちが考える「利便性のある移動ロボット」の例として、パナソニック社の「HOSPI®」や、ムラテック社の「MKR-003」などがある。

パナソニック社の「HOSPI®」は医療サービス向上をサポートする自律搬送ロボットである。休日や夜間における患者のベットサイドサポートに集中したいとき、スタッフに代わり薬剤や検体の搬送を自動で行ってくれる。あらかじめ記憶させた地図情報に基づき、高性能センサと高度な障害物回避アルゴリズムにより、移動環境を認識しながら走行するため、人や車椅子に接触することなく、安全かつ効率的に様々なものを搬送することが可能である。具体的には、細い点滴台や背の低い歩行器などを検知し、減速や進路変更をすることで、安全に走行する。

さらに、搬送時における上下動の衝撃力は小さく、液体の泡立ちを抑えることができ、紙コップに入った液体の搬送も可能である。

従来の搬送設備とは異なり、壁や天井への機器埋め込みや通路へのガイドテープの施工が不要なため、搬送の目的地や施設のレイアウト変更も容易かつ低コストで行える。

図 1-6　HOSPI®（パナソニック）[1-6]

また、ムラテック社の「MKR-003」は自動地図作成機能と自己位置認識機能を持ち、エリア内の目的地まで最適なルートを自分で判断して障害物を回避しながら移動できるロボットである。

自己位置推定技術として「SLAM」(Simultaneous Localization and Mapping) の技術を用い、設備側に埋設工事等を必要としない導入容易な搬送システムとなっている。後部に連結した台車には 30kg までの荷物を搭載でき、人とすれ違う際は、音声とジェスチャーで自身の進行方向を伝えながら、全方向移動可能な車輪を用いてスムーズにすれ違うことが可能である。また、エレベーター制御部との通信により階間移動も実現したことにより、人材確保や看護師の付帯業務軽減の課題も解決している。

MKR-003 が担うのはおもに搬送業務になるが、看護師が扱う搬送物には、薬剤をはじめ生命に関わるものから、補充品などの緊急性の低いものまで様々である。

このように、看護師のバックステージの業務をロボットがサポートすることで、看護師の業務にゆとりを作り、それにより、院内サービスの品質を向上させることができる。

図 1-7　MKR-003（ムラテック）[1-7]

図 1-8　利用例 [1-8]

1.2.3 研究の目標、目的

「HOSPI®」は、あらかじめ記憶させた地図情報に基づき、高性能センサであるレーザーレンジファインダで周辺の距離や角度を検知し、カメラはセキュリティ用途で使われている。

「MKR-003」は、SLAM を用いて地図情報を読み取り、ステレオカメラだけでなく超音波センサやレーザーレンジファインダを使い、壁や障害物とロボットとの位置関係を認識して周囲環境を把握している。

一方、スバルの「アイサイト (Ver.4)」は以前のバージョンで搭載していたミリ波レーダーやレーダーレーザーを用いず、ステレオカメラのみでの走行が可能になっている。

我々は、ステレオカメラのみで物体との距離が測定でき、目的地までたどり着ける「利便性のある移動ロボット」を目指す。

1.2.4 研究の進行

我々は研究を進めていく上で、項目ごとに以下のように目標を設定した。

設計

・ステレオカメラをルンバの上に設置する簡易的な台を作る。

・ルンバの上にステレオカメラ、マイコンボード、Surface を載せる台を検討する。

画像処理

・テンプレートマッチングにより、座標の取得を行う。

・ステレオカメラにより取得した画像からデプス画像を作成し、距離を算出する。

・ガウシアンフィルタを用いて精度の向上を試みる。

・ステレオカメラとルンバを用いて利便性のあるロボットを考案、実験をする。

ロボット制御

・Arduino IDE を用いてルンバの動作確認を行う。

・Visual Studio 2017 で前進後退や回転の動作を行う。

以上の目標をそれぞれ達成することで、移動ロボットの開発を進める。

第 2 章　目標検知（相澤担当）

2.1 概要

2.1.1 目的

移動ロボットが実際に移動を行うためには、目標地点の設定が必要である。

本研究では、目標地点を「目標物の存在する位置」とし、その目標地点がどこにあるかを認識することが本節の目標である。

2.1.2 仕様

本節の目的を達成するために要求される要素は以下の二点である。

1. 目標物の存在を検出できること
2. 目標物の存在する画像内での座標を出力できること

2.1.3 目標物の設定

まず目標物は、本研究の最終的な目標とその想定環境（1.2.1 参照）から考えて実生活上に存在するものが好ましいと考えた。

そのため、第一の目標物として市販の菓子の箱を用いることとした(図 2-1)。その結果については、2.2 で述べる。

図 2-1　目標物 1

しかし、実際に研究を進めていくにあたり問題が発生したため、より検出が容易な単色の立方体(図 2-2)を用いることとした。その結果については 2.3 で述べる。

図 2-2　目標物 2

2.1.4 使用ソフト

本節の目標を達成するためには、カメラで取得した画像に対し、一定の処理を行う必要がある。その為に使用したツールについて解説する。

また、本章に登場するソフトウェア及びライブラリは、以後の章でも使用される。

・OpenCV

OpenCV(Open Source Computer Vision Library)は、画像処理や画像解析を行う際にC/C++やPythonなどで利用できるライブラリである。

ライブラリとは、汎用性の高いプログラムをまとめたものであり、ユーザーは内包された任意の機能を呼び出すことができる。

本研究に利用した画像処理は、このライブラリを利用して行われている。

・Visual Studio

Visual Studioは、マイクロソフトの総合開発環境である。

本研究では、この開発環境を用いVisual C++を使用するものとする。

・Ovrvision Pro SDK

Ovrvision Proを使用するために必要なツールセットのことである。

Ovrvision Proの公式サイトから配布されており、このステレオカメラを利用して開発を行うために必要となる。

主にカメラの設定の調整と、その入出力に関わる機能などを有している。

2.2 テンプレートマッチング

2.2.1 物体検出方法の選択

カメラから出力される画像を用いて物体を検出しようとする場合、その方法は複数存在している為、対象に適切と思われる手法を選択する必要がある。

本章では、図 2-1 の目標物を検出することが目的であり、その方法としてテンプレートマッチングを選択した。

2.2.2 テンプレートマッチングの原理と利点

テンプレートマッチングとは、検出したい目標物のもととなる画像(テンプレート)を用意し、その画像をカメラの映像中から探索する方法のことである。

図 2-1 の画像を用いてテンプレートマッチングを行う場合、図 2-3 のように画像の左上から右下に向かってテンプレートを探すこととなる。

この手法の利点は、複雑な形状や配色の物体に対しても物体検出が可能なことである。

図 2-3　テンプレートマッチングの模式図

2.2.3 テンプレートマッチングによる試行結果・考察

Ovrvision Pro はステレオカメラであるが、本研究における物体検出には、その一方である右側のカメラの映像を用いることとした。

実際にテンプレートマッチングを行った結果、図 2-4 のように、テンプレートとほぼ同じサイズの目標物が画像内に存在するときは探索に成功した。しかし、図 2-5、図 2-6 と距離が変化していくことで、図 2-5 では物体を中心にとらえておらず、図 2-6 では位置をとらえることもできないという問題が起こった。

図 2-4 テンプレートマッチングの実験例 1

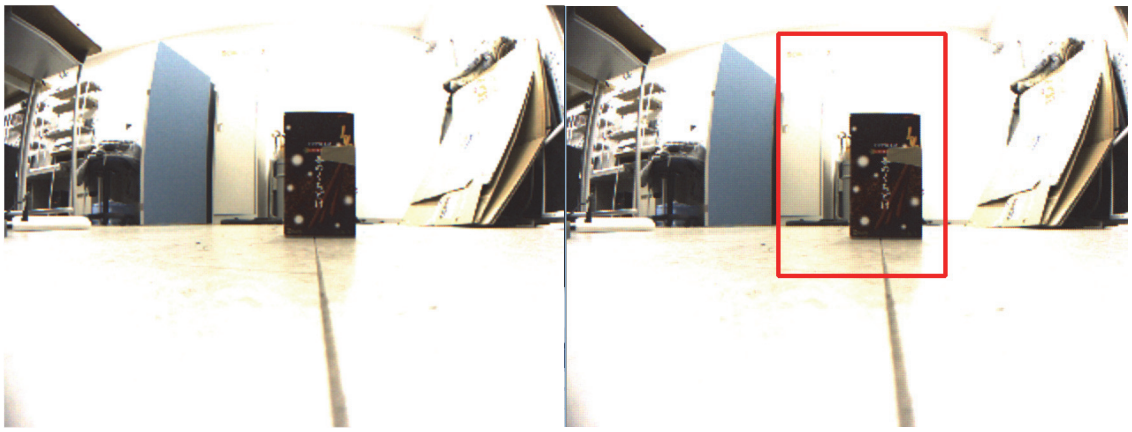

図 2-5 テンプレートマッチングの実験例 2

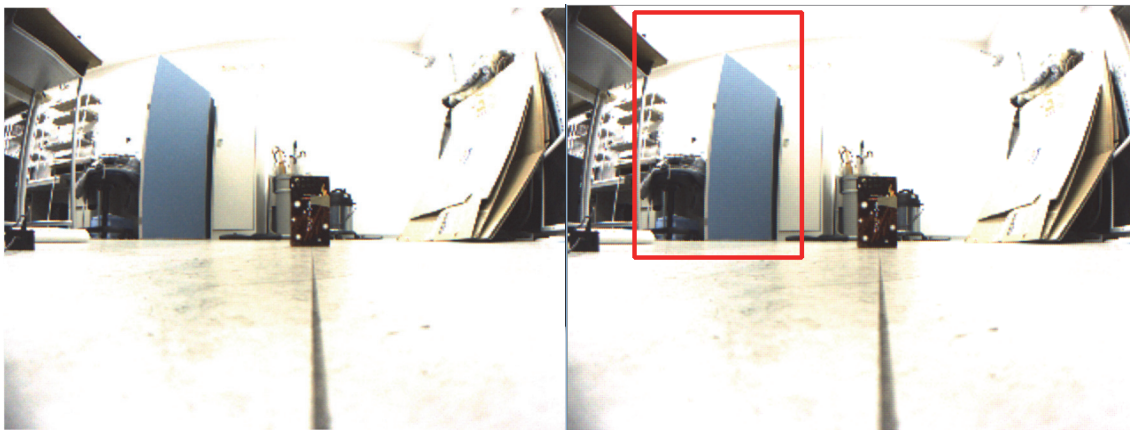

図 2-6 テンプレートマッチングの実験例 3

このように、テンプレートマッチングではテンプレートとほぼ同じサイズの目標物しか見つけることができない。

この問題点を解決するためには、複数のテンプレートを用意し、様々な距離に対応させる方法がある。しかしこれは探索時間が大幅に伸びるという欠点がある。

その為、我々はテンプレートマッチングを、移動ロボットのための物体検出の手法として適切ではないと結論づけることとした。

2.3 色検出

2.3.1 色の検出と HSV 色空間

テンプレートマッチングの結果を踏まえ、本研究に用いる物体検出の方法を特定の色を検出する手法に変更した。

色の検出は、カメラから取得した画像内から、指定した色の部分を探すというものである。

色の指定は RGB 色空間による指定が基本である(図 2-7)。

図 2-7　RGB 色空間[2-1]

しかし、実際にこの指定方法を用いると明るさの変化が考慮に入っていない為、目標物周辺の明るさが変化すると正常に動作しない。

これを解決するために HSV 色空間を用いることとした。

HSV とは、色相(Hue)、彩度(Saturation)、明度(Value)の三成分を指している。

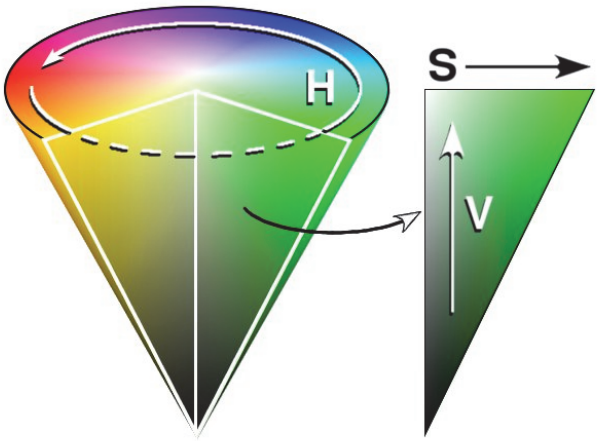

図 2-8　HSV 色空間[2-2]

この成分のうち V に当たる明度は、色の明るさに該当する成分であるため、この成分を除いて色の指定を行うことで、影などの影響を受けにくい色の検出が可能となる

実際に HSV 色空間を用いるために使用したプログラムは以下の通りである。

```
242             //HSV変換
243             Mat hsv_image, mask_image, output_image, dst_img;
244             cv::cvtColor(right_img, hsv_image, CV_BGR2HSV, 3);
```

図 2-9　HSV 変換のプログラム

244 行目の命令で、Ovrvision Pro の右カメラ右画像 right_img を HSV 色空間における画像 hsv_image として出力する。

```
245 #define RH_MAX 15
246 #define RH_MIN 0
247 #define RS_MAX 255
248 #define RS_MIN 160
249 #define RV_MAX 255
250 #define RV_MIN 0
```

図 2-10 指定する色の定義

検出対象となる色は図 2-10 のように指定している。これは赤色を検出する際のパラメータである。

RH は色相、RS は彩度、RV は明度を示しており、それぞれの最大値と最小値が指定されている。

彩度と明度は 0～255 の値、色相は 0～180 の値を指定できる。この中で V の指定領域が 0～255 になっているが、これは明るさに対して条件を課さないことを意味する。

```
251             Scalar s_min = Scalar(RH_MIN, RS_MIN, RV_MIN);
252             Scalar s_max = Scalar(RH_MAX, RS_MAX, RV_MAX);
253             inRange(hsv_image, s_min, s_max, mask_image);
```

図 2-11 色の識別プログラム

図 2-11 では図 2-10 で指定した範囲の色を白、それ以外を黒として二値化した画像を mask_image に出力する。これによって画像内から指定した色の部分を識別することができる。

2.3.2 色検出による物体検出の試行結果と考察

HSV 色空間を用いて実際に目標物の検出を行った結果、距離の変化や影の影響を大きく受けることなく、指定した色の物体の検出を行うことができた(図 2-12 から図 2-17)。

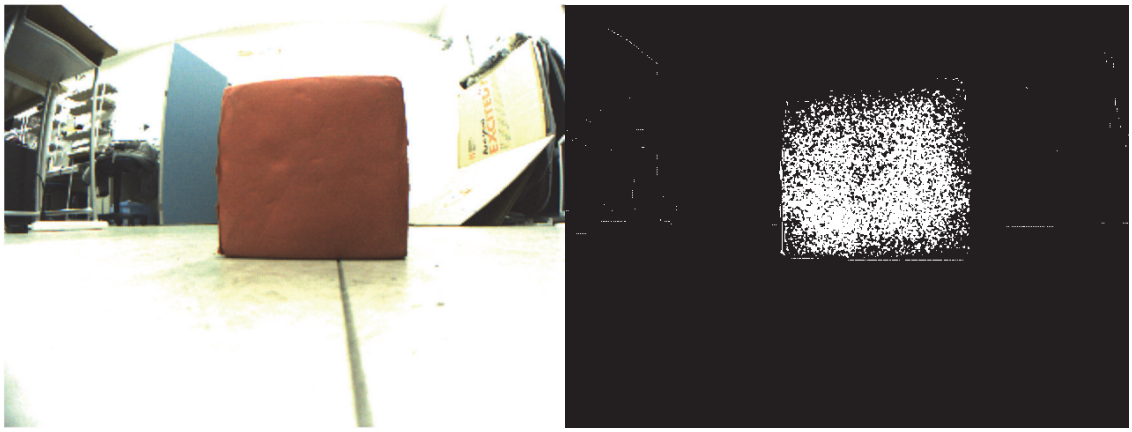

図 2-12　色検出の試行結果(近距離)

図 2-13　色検出の試行結果(中距離正面)

図 2-14　色検出の試行結果(中距離右寄り)

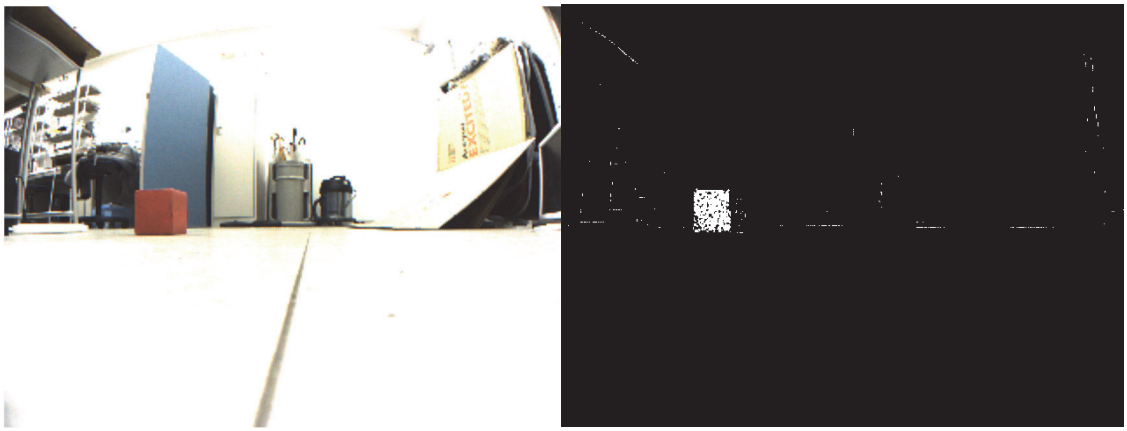

図 2-15　色検出の試行結果(中距離左寄り)

図 2-16　色検出の試行結果(遠距離正面)

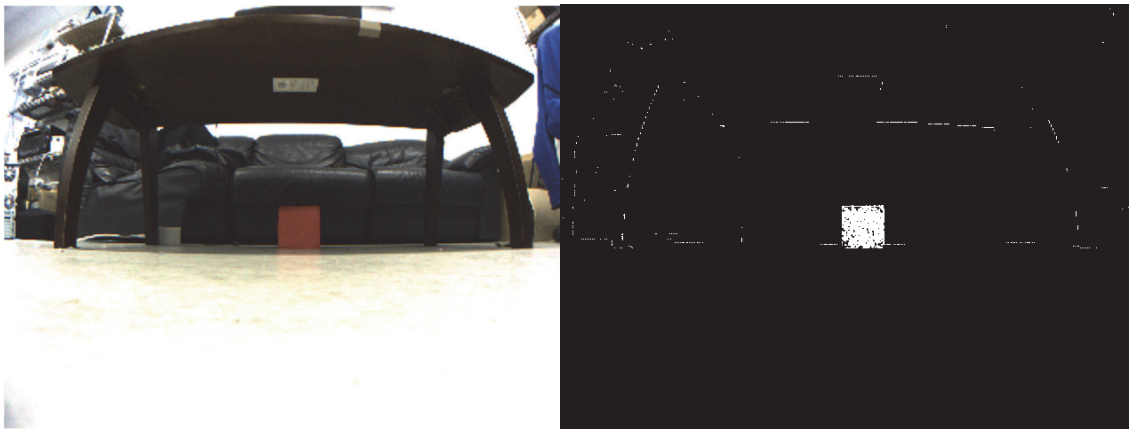

図 2-17　色検出の試行結果(正面、影のある環境下)

しかし物体の輪郭を正確にとらえられている一方で、別の物体の輪郭がところどころノイズとして映り込んでいるため、ノイズに対する対策は必要であると考えられる。

その対策方法を 2.4.2 で解説する。

2.4 座標の出力

2.4.1 座標出力

2.3.2 の結果から、画像内から指定した色の物体を検出することが可能であることが分かった。本節ではその位置である座標を出力する方法を解説する。

座標の出力は、今後我々の制作したロボットを動かす上で、その処理の対象地を決めるという重要な役割を担っている。

2.4.2 検出ノイズの除去

2.3.2 で指定した色の物体を検出することが可能になったが、目標物以外にも白い点がたくさんある。このようなノイズは、物体の座標を出力する際、誤った座標を出力する要因となる可能性がある。その為、結果画像にメディアンフィルタをかけるという画像処理を行った。メディアンフィルタは図 2-18 の命令により適用することができる。

```
254        medianBlur(mask_image, dst_img, 7);
```

図 2-18 メディアンフィルタの命令

それにより、図 2-12 から図 2-17 の右に示したような二値化画像 mask_image にメディアンフィルタをかけ、dst_image として出力する。

このフィルタは小さなノイズの除去に有効なフィルタであり、実際にこの処理を行った結果は図 2-19 のとおりである。

図 2-19　メディアンフィルタを適用する前（左）と適用後（右）の比較

2.4.3 検出物体の輪郭の描写と重心の測定

画像内での物体の位置を得るために、2.4.2 の結果を元に物体の輪郭の描写を行う。輪郭の描写のためのプログラムは図 2-20 のとおりである。

```
255             //描写する輪郭の設定値を指定
256             const int contour_index = -1;
257             const int max_level = 2;
258             /// 輪郭の検出
259             std::vector<std::vector<cv::Point> > contours;
260             std::vector<cv::Vec4i> hierarchy;
261             // 2値画像，輪郭（出力），階層構造（出力），輪郭抽出モード，輪郭の近似手法
262             cv::findContours(dst_img, contours, hierarchy, cv::RETR_TREE, cv::CHAIN_APPROX_SIMPLE);
263             Mat rightimg_tag;
264             if (contours.empty()) {
265                 std::cout << "輪郭画像が見つかりません" << std::endl;
266             }
267
268             else {
269                 /// 輪郭の描画
270                 // 画像，輪郭，描画輪郭指定インデックス，色，太さ，種類，階層構造，描画輪郭の最大レベル
271                 right_img.copyTo(rightimg_tag);
272                 cv::drawContours(rightimg_tag, contours, contour_index, cv::Scalar(0, 0, 255), 2, CV_AA, hierarchy, max_level);
```

図 2-20　輪郭描写のプログラム

ノイズ除去後の二値化画像 dst_img をもとに輪郭画像を作成している。

このプログラムでは輪郭が存在しない場合はエラーとして扱いそれ以外の場合は図 2-21 のように輪郭を描写する。

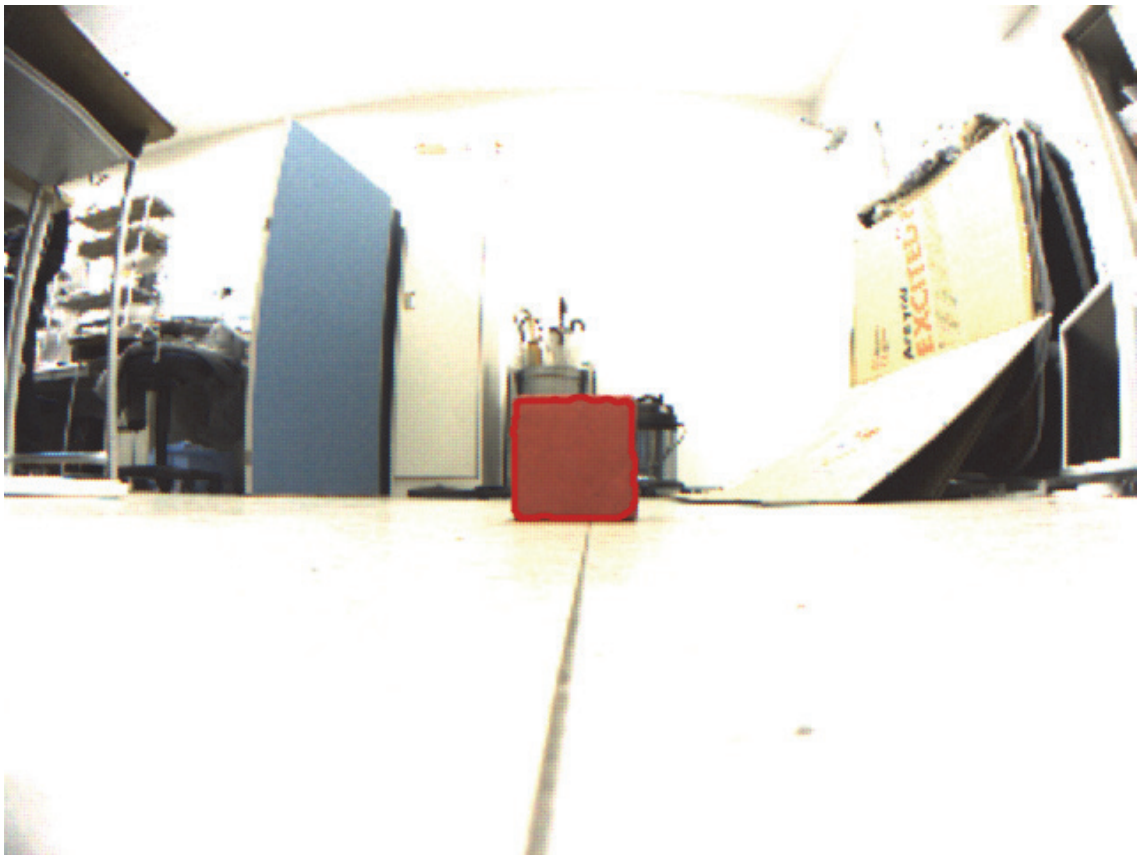

図 2-21　目標物の輪郭を描画した画像

次にこの輪郭に基づき重心位置の算出を行う。重心位置の計算方法は以下のとおりである。なお、輪郭が複数存在する場合、最も大きな輪郭に対して計算が行われる。

```
274                 //重心を計測
275                 double max_area = 0;
276                 int max_area_contour = -1;
277                 for (int j = 0; j < contours.size(); j++) {
278                     double area = contourArea(contours.at(j));
279                     if (max_area < area) {
280                         max_area = area;
281                         max_area_contour = j;
282                     }
283                 }
284                 int count = contours.at(max_area_contour).size();
285                 int x = 0;
286                 int y = 0;
287                 for (int k = 0; k < count; k++) {
288                     x += contours.at(max_area_contour).at(k).x;
289                     y += contours.at(max_area_contour).at(k).y;
290                 }
291                 x /= count;
292                 y /= count;
293
```

図 2-22　重心位置の算出プログラム

これによって重心位置を計算して出力したのが図 2-23 である。

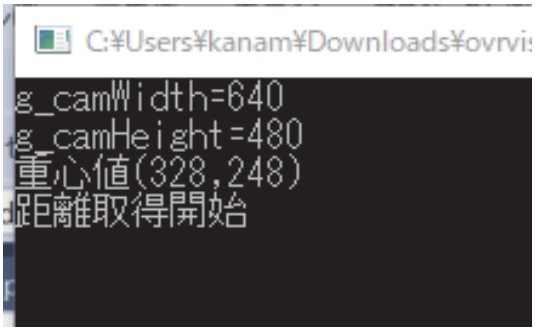

図 2-23　座標出力の例

これにより目標物の重心の座標が(328,248)であることが分かる。画像サイズは 640×480 であるから、目標物は画面のほぼ中央に存在することが分かる。

ここで出力された値は 3.2 における距離の算出や進行方向の決定の際に用いることとなる。

2.4.4 評価

2.4.3 から、座標の取得は概ね正確に行えていることが分かった。

しかし、現在の重心座標を算出する方法では、目標物よりも大きい輪郭を持つ物体が画面内に存在する場合、正しい座標が得られない。

そのため、指定した色に近い物体が周辺に存在する環境では、輪郭の大きさ以外に形など別の判定方法を用意する必要があり、今後の検討課題である。

第 3 章 距離の測定（相澤担当）

3.1 概要

3.1.1 目標

移動ロボットが実際に移動をする際、進むべき距離が必要である。本章では、目標物までの距離を算出する方法を解説する。

3.1.2 ステレオカメラによる距離測定の方法・原理

目標物までの距離を測定する方法には、ステレオマッチングと呼ばれるものがある。

ステレオマッチングとは、二つの並行して置かれたカメラの画像の視差から、目標物までの距離を求める方法である。

視差とは、二つの画像内における目標物の位置のズレのことを指す。視差を利用した実際の距離の計算方法を、図 3-1 に示す。

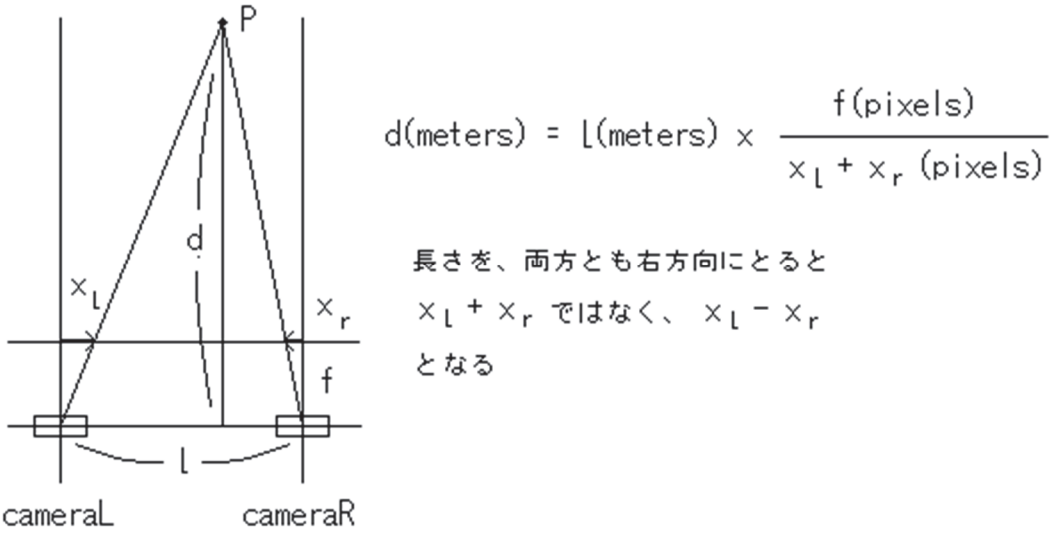

図 3-1　ステレオ法の計算方法[3-1]

二台のカメラ CameraL と CameraR が水平に配置され、向きが平行になっているとすると、視差 $x_l + x_r$ を用いて、目標物までの距離 d が求められる。ただし、カメラ間の距離 l と焦点距離 f が必要である。

この原理によると、視差が大きければ目標物が近くにあり、逆に小さければ目標物が遠いという意味を持つことが分かる。そして、この視差の値を直接視覚化し画像にしたものを視差マップ(デプスマップ)という。

本研究では、OpenCV を用いて視差マップを作成し、それを距離と対応づける。

図 3-2 は、SGBM(セミグローバルブロックマッチング法)を用いて視差マップを作成するプログラムである。

本研究ではこれを用いて視差マップを作成し距離を測定していく。

各種パラメータは参考文献[3-1]をもとに設定した。

```
212         // デプス画像の取得
213         cv::Mat disp, disp8, disp8uc4;
214         // SGBMによるデプス画像の作成
215         int window_size = 3;
216         int minDisparity = 0;
217         int numDisparities = 48;
218         int blockSize = 5;
219         int P1 = 8 * 3 * window_size * window_size;
220         int P2 = 32 * 3 * window_size * window_size;
221         int disp12MaxDiff = 1;
222         int preFilterCap = 0;
223         int uniquenessRatio = 10;
224         int speckleWindowSize = 200;
225         int speckleRange = 1;
226
227         cv::Ptr<cv::StereoSGBM> ssgbm = cv::StereoSGBM::create(
228             minDisparity,
229             numDisparities,
230             blockSize,
231             P1,
232             P2,
233             disp12MaxDiff,
234             preFilterCap,
235             uniquenessRatio,
236             speckleWindowSize, speckleRange, cv::StereoSGBM::MODE_SGBM);
237         ssgbm->compute(left_img_gray, right_img_gray, disp);
238
```

図 3-2　SGBM のプログラム

得られた視差マップの例を以下に示す。

図 3-3　視差マップの作成元画像

図 3-4　視差マップの例

図 3-4 のような視差マップの濃淡から、目標物までの距離に対応する数値を求められる。

この数値をデプス値とし、目標物までの実測値と共にグラフ化することで、デプス値から距離を算出する近似式が得られる。

3.2 距離の算出

3.2.1 必要データの収集方法

デプス値と実際の距離の値の対応を取得しグラフを作成する為、実験を行った。

実験の手順を以下に示す。

1. メジャーを使用してカメラと目標物の直線距離を求める
2. 目標物をカメラから 200㎜ の位置に設置する
3. 目標物とのデプス値を記録する
4. 目標物をカメラからさらに 50 ㎜遠ざけた位置に設置する
5. 手順 3 と 4 をデプス値が検出できなくなるまで繰り返す

以上の方法を三回繰り返し、デプス値の三回の平均値をグラフに用いる。

デプス値を取得するプログラムは以下のとおりである。

```
329                     //重心を中心に50*50の画素値を取得
330                     int xst = x - 25;
331                     int yst = y + 25;
332                     int xend = x + 25;
333                     int yend = y - 25;
334                     int average = 0;
335
336                     for (xst = x - 25; xst < xend; xst++) {
337                         if (xst >= 640 || yst >= 480) { continue; }
338                         for (yst = y + 25; yst > yend; yst--) {
339                             int pix = 0;
340                             if (xst >= 640 || yst >= 480) { continue; }
341                             pix = static_cast<int>(disp.at<unsigned char>(xst, yst));
342                             if (pix > 30) {
343                                 //sum計算
344                                 average += pix;
345                                 n = n++;
346                             }//画素値判定
347                         }//ｙライン
348                     }//xライン
349
350                      //平均値取得
351                     if (average > 0)
352                     {
353                         average = average / n;
354                         std::cout << "¥n" << "平均値" << average << "¥n";
355                         std::cout << "項目数" << n << "¥n";
356                         std::cout << "画像処理終了" << endl;
357                     }
```

図 3-5　デプス値の取得プログラム

取得を行うエリアは、物体検出に基づき算出された重心の座標とその周辺上下左右 25 ピクセル分の領域とする。

この計算に際し、デプス値が 30 以下の値は対象エリア内の測定がうまくいかなかったノイズとして存在すること多かった為、除外して計算を行うことにした。

3.2.2 算出式の作成

3.2.1 の実験をもとに作成したグラフが図 3-6 である。

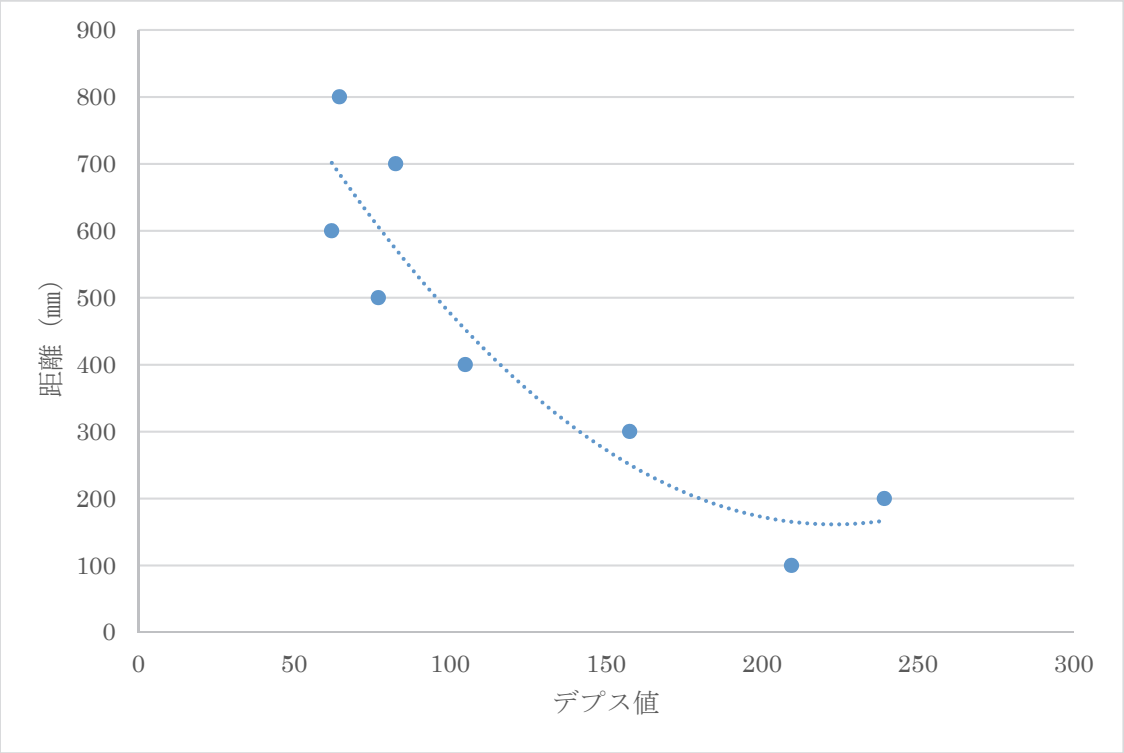

図 3-6　デプス値と距離の関係

今回取得したできたデータは 800 ㎜に対するデプス値までである。これより遠距離ではデプス値がほぼ 0 になってしまい、距離の判定が行えていない。

このようにして、取得したデータから近似式を作成した結果、デプス値 x、距離 y(㎜)に対し

$y = 0.0208x^2 - 9.2897x + 1197.5$

という式が得られた。

この式を用いて実際の距離を算出する。

3.2.3 評価・考察

図 3-6 のグラフから､デプス値から実際の目標物までの距離を得ることが可能になった。

しかし、デプス値は試行ごとの揺らぎが大きく信頼度がとても低い。

この信頼度の低さの原因は､カメラ自身の性能や特性に基づくものであると考えられ､市販のステレオカメラではあるものの、距離の測定に特化したものではない為このような結果を引き起こしていると考えられる。

それ以外には、SGBM の性能を生かせていない、視差マップ内のノイズが除去しきれていないなどの問題が考えられる。

しかし、我々が試行錯誤を行った結果、デプス値として 0 に近い値か 255 に近い値しか拾わない状況を多く確認しており、有効に測定が行える範囲が 1m もないことからも、ハードウェア的な問題の要因が大きいと考えている。

第 4 章 移動ロボットの制御（柿沼担当）

4.1 概要

4.1.1 目的

本章では、移動ロボットの制御を行うためのソフトウェア、回路およびプログラムについて解説する。

4.1.2 要求機能

本章にて解説する移動ロボットの制御を実現するには以下の機能が要求される。

1.PC、Arduino、ルンバの接続とシリアル通信

2.Arduino IDE を用いたルンバの動作確認

3.Visual Studio 2017 を用いたルンバの動作確認

上記の要求機能を満たす方法について詳しく解説していく。

4.2 ルンバの接続と制御

4.2.1 使用機器

今回の研究で用いるマイコンボードは図 4-1 に示す Arduino Mega 2560 である。ルンバ上部の面積と比べて極めて小さいため、ルンバに載せるのが容易である。

このマイコンは ATmega2560 チップを搭載しており、Arduino LLC および Arduino SRL が設計・製造を行っている。デジタル入出力ピンが 54 ピン、アナログ入力ピンが 16 ピンあり、USB 接続や電源ジャックなどを搭載している。USB ケーブルで PC と接続するか、電源を投入すればすぐに動作させることができる。また Arduino UNO のメモリを増やした仕様になっている。Arduino Mega 2560 の仕様を表 4-1 に示す。

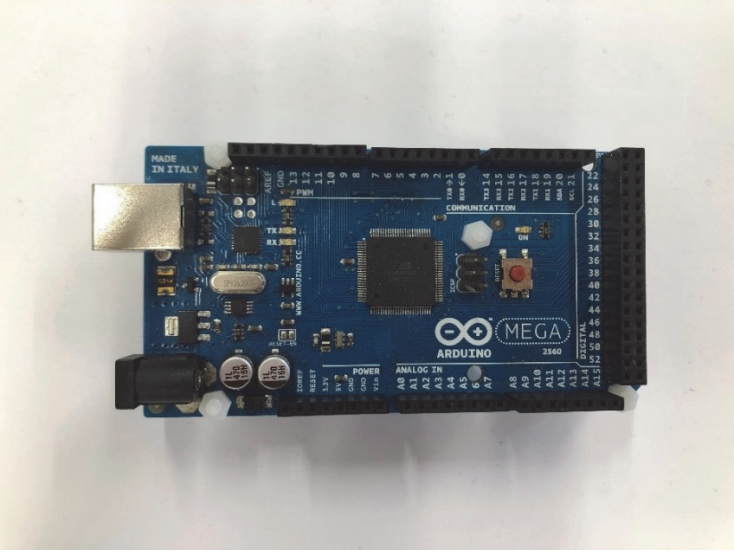

図 4-1　Arduino Mega 2560

表 4-1　Arduino Mega 2560 の仕様

動作電圧	5V
入力電圧（推奨）	7-12V
入力電圧（限界）	6-20V
デジタル入出力ピン	54 ピン
アナログ入力チャネル	16 ピン
I/O ピンでの直流電流	40mA
3.3V ピンでの直流電流	50mA
クロック速度	16MHz

4.2.2 ルンバオープンインターフェイス(ROI)

ルンバと Arduino Mega を接続するためには、図 4-2 と図 4-3 に示したルンバオープンインターフェイス(ROI)端子に接続する必要がある。ROI 端子上のピンに対して、外部の機器からシリアル通信でコマンドを送れば外部からルンバをコントロールできる。

図 4-4 と図 4-5 のようなコネクタを作成して接続する。

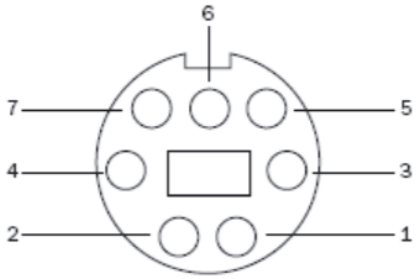

図 4-2　ROI 端子のイメージ図[4-1]

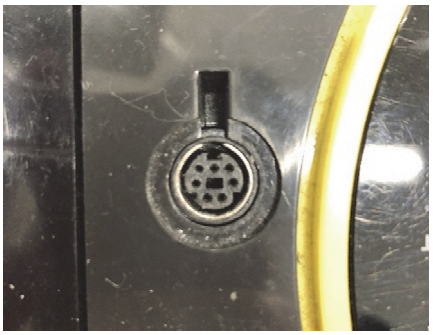

図 4-3　実際の ROI 端子

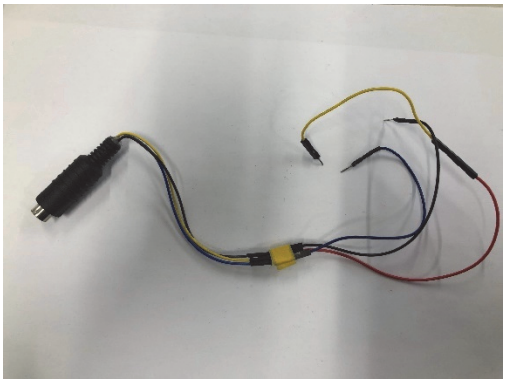

図 4-4　コネクタ

図 4-5　コネクタと ROI 端子の接続

4.2.3 接続方法

Arduino Mega 2560 と ROI 端子を表 4-2 のように接続する。マイコンボードとルンバの配線を図 4-6 に示す。マイコンボードと PC を USB ケーブルで接続することでマイコンボードの動作に必要な電力は PC から供給される。

表 4-2　接続方法

色	Arduino	ROI
黒	GND	GND
青	TX1	RX
黄	RX1	TX

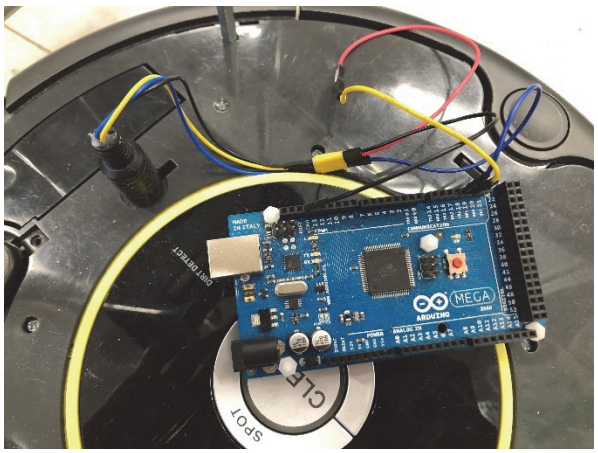

図 4-6　Arduino Mega 2560 とルンバの接続

4.2.4 制御方法

ルンバの制御を行うためにはコマンドをルンバに送信する必要がある。コマンドとは「コンピューターに特定の機能の実行を指示する命令」のことである。

今回使用したマイコンでは先行研究で使われていたプログラムをそのまま用いることにした。ルンバにはエンコーダーが搭載されており、進んだ距離や角度のデータを得られる機能がある。先行研究が用いていたのはエンコーダーから得られる距離と角度の数値と、指定した数値が同じになった時に停止するプログラムである。

本研究で用いたコマンドを表 4-3 に示した。この他にも数多くのコマンドがあるが、本研究で用いなかった為割愛する。表のように大文字のアルファベットの後に 4 桁の数字を打ち込み送信キーを押すことでコマンドを送信することができる。小文字の s で強制停止を行うこともできる。

なお大文字の S と 4 桁の数値(指定子)で速度の変更が可能だが、速度が速すぎると距離と角度の精度が落ちてしまうので注意しなければならない。

表 4-3　本研究で用いたコマンド

パターン	コード	指定子
前進	W	4 桁の数値
後退	X	
右旋回	D	
左旋回	A	
強制停止	s	なし

シリアル通信でコマンドをルンバに送信することで、ルンバを制御することができる。例えばコマンドの入力例は以下のようである。

例 1) 50cm 前進：W0050

例 2) 210° 右旋回：D0210

コマンドの後ろに入力する数値の単位は前進後退時が[cm]、回転時が[°]であり、3 桁の範囲の数値でなければならない。

4.2.5 ルンバの回転

我々の移動ロボットは直進を始める前に物体を探索するために回転を行う。物体検出に関しては 2.3 を参照すること。図 4-7 は対象物がルンバの右方向にある時、図 4-8 は対象物がルンバの左方向にある時の様子を示している。

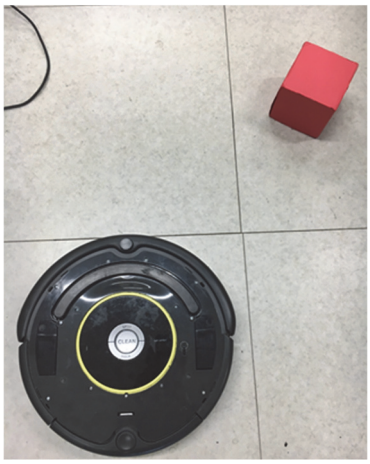

図 4-7　右方向に対象物

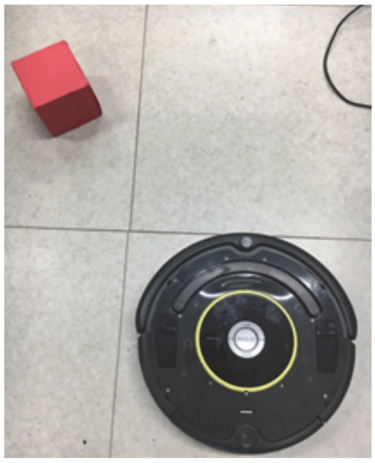

図 4-8　左左向に対象物

図 4-9　正面に対象物

ルンバは対象物を正面にとらえるように回転した後停止する。この際、カメラでとらえた映像から正面でとらえるまでの差を検出し右回転時には 5 度、左回転時には 10 度ずつ回転しながら図 4-9 のように対象を正面にとらえる。

4.3 動作確認

4.3.1 Arduino IDE を用いて動作確認

4.2.3 項で示した接続方法で PC、マイコンボード、ルンバを接続した。そして Arduino IDE を用いてルンバの動作確認を行う。

Arduino IDE の「ツール」から図 4-10 のようなシリアルモニターを開く。そして「W0050」といったコマンドを手動で入力して右の「送信」ボタンをクリックして動作確認を行った。なお、送信は PC の Enter キーで行うこともできる。今回は「前進」、「後退」、「右旋回」、「左旋回」の 4 種類のコマンドを入力した。コードだけではなく指定子も変えて実験したが、どの動作も問題なく作動した。

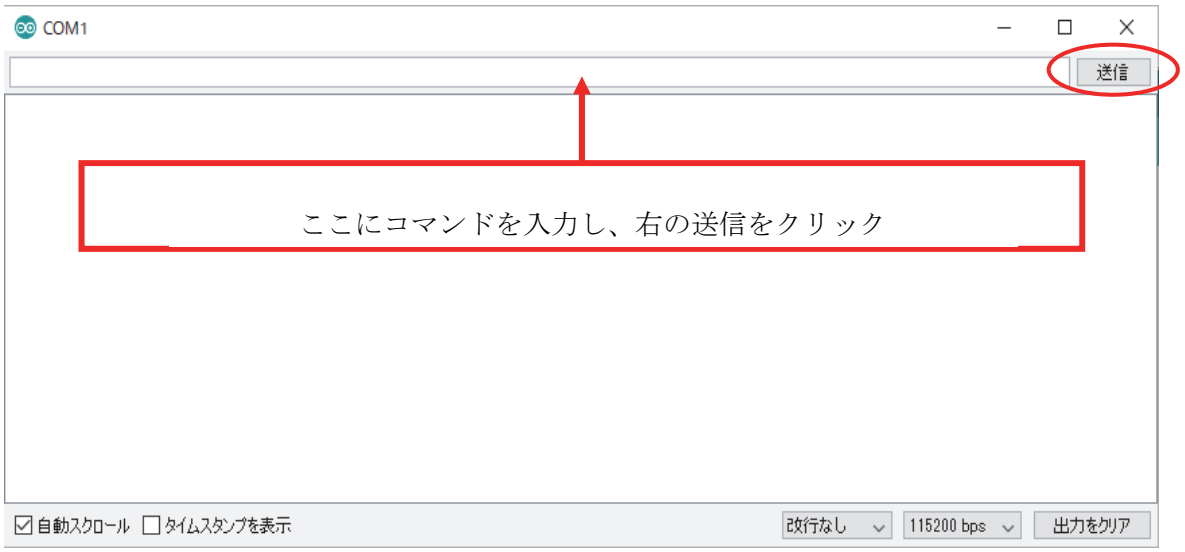

図 4-10　Arduino IDE のシリアルモニター

4.3.2 Visual Studio 2017 を用いて動作確認

次に Visual Studio 2017 を用いて動作確認を行った。Arduino Mega 2560 にコマンドを送信するプログラムを作成し、ルンバの動作を確認した。

以下のプログラムは最初に W0050(50cm 前進)を送信し、そのコマンドを送信してから 9.5 秒後に次の D0180(180° 右旋回)を送信するプログラムある。

このプログラムにより、ルンバが問題なく動作することが確認できた。

```
// ConsoleApplication1.cpp :
//

#include "stdafx.h"

#include <windows.h>
#include <string>

HANDLE arduino;
bool Ret;

int main()
{
char o;
//BYTE data = o;
//1.ポートをオープン
arduino = CreateFileA("COM4", GENERIC_WRITE | GENERIC_READ, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);

        if (arduino == INVALID_HANDLE_VALUE) {
        printf("PORT COULD NOT OPEN¥n");
        system("PAUSE");
        exit(0);
}
DWORD dwSendSize;
```

```
        }
        DWORD dwSendSize;

std::string Y; // ここでYを宣言
Y = "W0050";  // 最初に送信するコマンド

DWORD lengthOfSent = 5; // 送信するコマンドの文字数
Ret = WriteFile(arduino, Y.c_str(), lengthOfSent, &dwSendSize, NULL);
// コマンドをマイコンに送信
//Ret = WriteFile(arduino, Y.c_str(), lengthOfSent, &dwSendSize, NULL);
if (!Ret) {
        printf("SEND FAILED¥n");
        CloseHandle(arduino);
        system("PAUSE");
        exit(0);
}

Sleep(9500) // 次のコマンドを送るまで待つ時間、単位は[ms]
    Y = "D0180" // 次に送信するコマンド
Ret = WriteFile(arduino, Y.c_str(), lengthOfSent, &dwSendSize, NULL);

CloseHandle(arduino);
printf("FINISH1¥n");

rerurn 0;
}
```

4.3.3 カメラの測定可能距離を踏まえた動作確認

3.2 のように本研究で用いるステレオカメラには測定可能距離が 800㎜ までであるという限界がある。そこで図 4-11 の黄色の矢印のように距離が 800㎜ 以下ずつの移動を繰り返して対象物を探索するという方針で制御を行う。

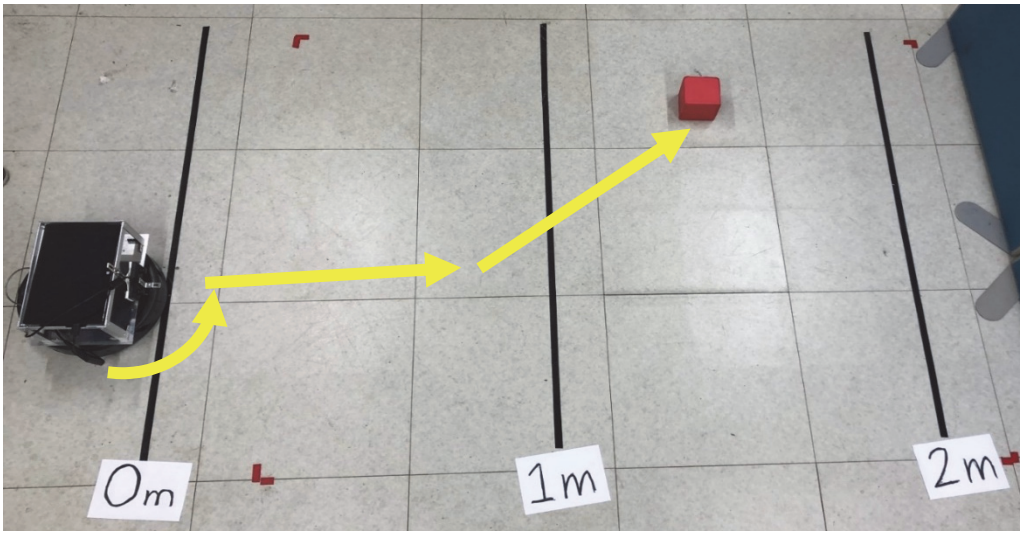

図 4-11　800㎜ 以下の距離ずつの移動

動作確認を行った結果、移動を繰り返しながら対象物を探索することに成功した。

4.4 最終評価

本章ではルンバの ROI 端子とマイコンボードを接続し、プログラムでルンバを制御できることを確認した。

ルンバに対して、Arduino IDE および自作のプログラムにより「前進」、「後退」、「右旋回」、「左旋回」を行わせることができた。

第 5 章 周辺パーツの制作（金子担当）

5.1 概要

5.1.1 目的

ルンバの上に Ovrvision Pro、Surface Pro 2、Arduino Mega2560 を設置し固定する周辺パーツの制作について説明する。

当初はルンバの上に物品をただ置いただけであったため、ルンバの移動中に配線の絡みや PC の落下などの問題がみられた。そこで Surface Pro 2 や Ovrvision Pro を固定するための部品の制作を行い、この問題を解決する。

5.1.2 要求機能・制約条件

周辺パーツを制作するにあたって、要求される機能を下記に示す。

1. Ovrvision Pro が固定できる部品である。
2. Surface Pro 2 を乗せた状態で移動するための台座である。
3. 制作した台座と Ovrvision Pro、ルンバとを結合できる。

移動ロボットとして動作させるにあたり、直進移動と回転移動による設置物への影響、距離測定と物体検出を用いるための Ovrvision Pro の設置位置を考慮するとさらに下記のことが制約される。

1. 移動の影響を受けず、Ovrvision Pro が固定される。
2. Surface Pro 2（275×173×13.46）を設置して台座が安定する。
3. 距離計測、物体検出に適した位置に Ovrvision Pro を設置できる。

これらを考慮し制作に取り組んでいく。

5.2 周辺パーツの設計

5.2.1 アイディア

制作に入るにあたり、周辺パーツを台座と Ovrvision Pro の固定部品に分けてイメージ図を作成した。

5.1.2 の要求を満たす台座のイメージ図が図 5-1 である。

図 5-2、図 5-3 は Ovrvision Pro の固定部品のイメージ図である。Ovrvision Pro を挟み込んで固定する方法を用いたので取り外すことも容易である。このイメージ図を基に設計を行う。

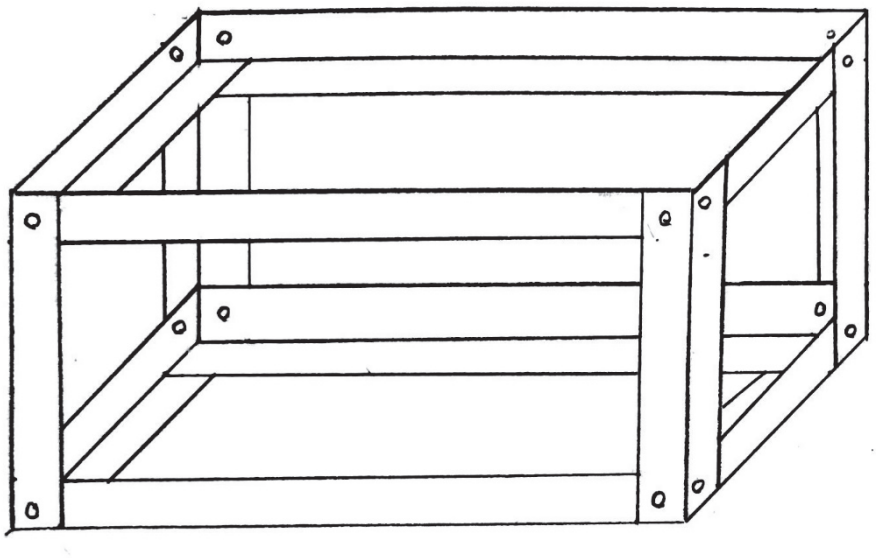

図 5-1 台座

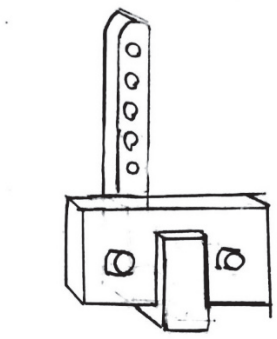

図 5-2 Ovrvision Pro の固定部品

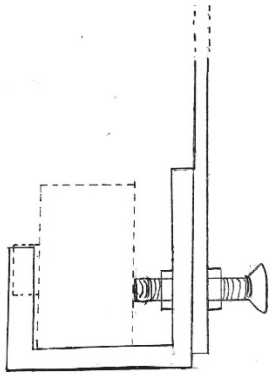

図 5-3 固定部品の側面図

5.2.2 設計

図 5-1～図 5-3 の台座と Ovrvision Pro の固定部品の図面を 3 次元 CAD ソフトウェアである SolidWorks を用いて作成した。

図 5-4～図 5-6 は図 5-1～図 5-3 のイメージ図をもとに作成した台座と Ovrvision Pro の固定部品のアセンブリである。

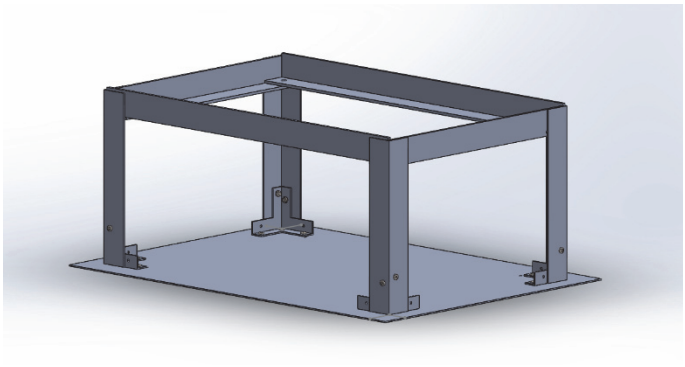

図 5-4 台座の完成図

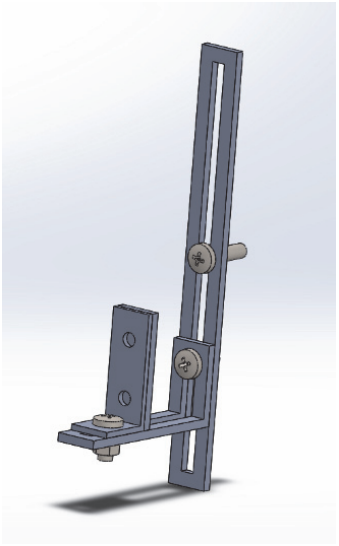

図 5-5 Ovrvision Pro の固定部品

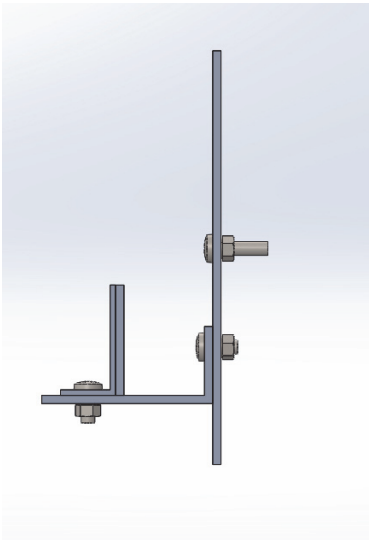

図 5-6 固定部品の側面図

5.2.3 台座の制作

初めに台座の設計について説明していく。

台座の制作に使用した部品は表 5-1 の通りである。

表 5-1　台座製作に必要な部品

部品名	寸法	制作会社	個数
アルミ等辺アングル	20×20×2×130	イマオコーポレーション	4
アルミ等辺アングル	20×20×2×185	イマオコーポレーション	2
アルミ等辺アングル	20×20×2×285	イマオコーポレーション	2
アルミ板	320×230×1	イマオコーポレーション	1
ユニクロ三方面	31×31×9	ハイロジック	4
リベット	直径 4.0mm	藤原産業	12
なべ小ねじ	M3×10	YAHATA	20

強度を上げるため、台座の底面の部分をユニクロ三方面（図 5-7）とアルミ板を用いることにした。図 5-8 はユニクロ三方面とアルミ板、アルミ等辺アングルを結合したものである。

図 5-7　ユニクロ三方面

図 5-8 ユニクロ三方面の接合部

また、Surface Pro 2 を設置する台座上面の四つ角にボルトを用いると移動による振動によってボルトが緩むことが考えられたため、ボルトと異なり簡単に取り外すことのできない図 5-9 のようなリベットを用いて固定した。これにより台座の安定性を高めることが出来た。

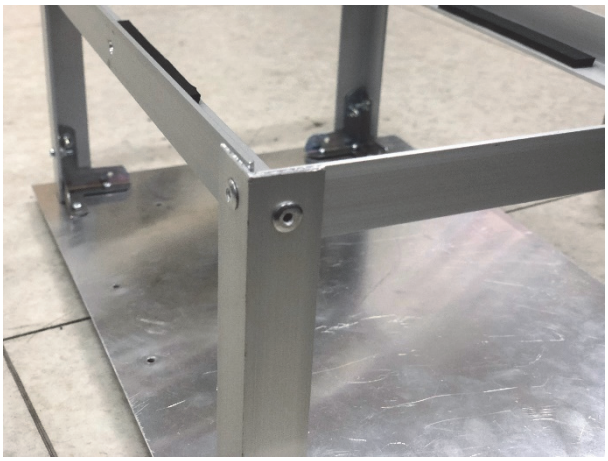

図 5-9 リベットで固定したアルミ等辺アングル

Surface Pro 2 とルンバの配線口を確保するため、図 5-10 と図 5-11 のように加工を加えた。

図 5-10 アルミ等辺アングル材加工部

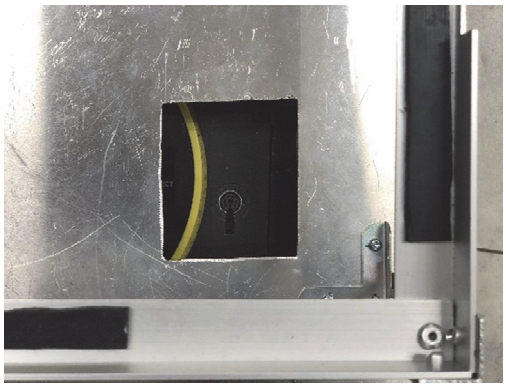

図 5-11 アルミ板加工部

図 5-12 は台座の完成図である。

図 5-12 台座完成図

台座に必要な部品の設計図を図 5-13～図 5-18 に示す。

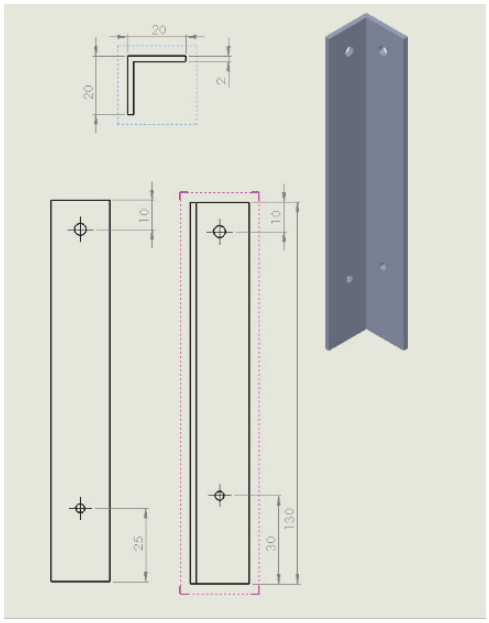

図 5-13 台座の柱

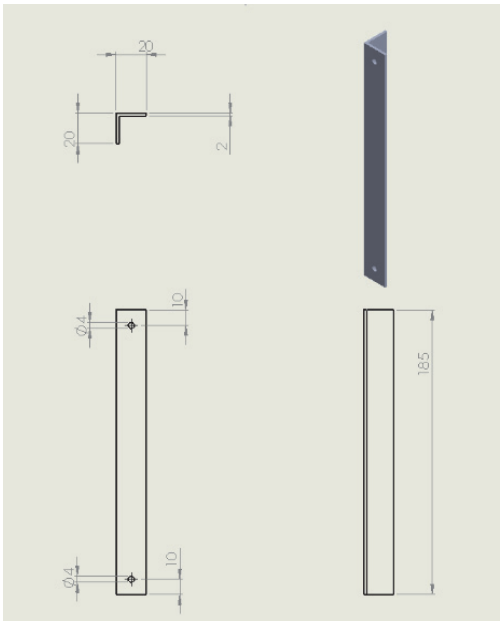

図 5-14 台座上面枠部品①

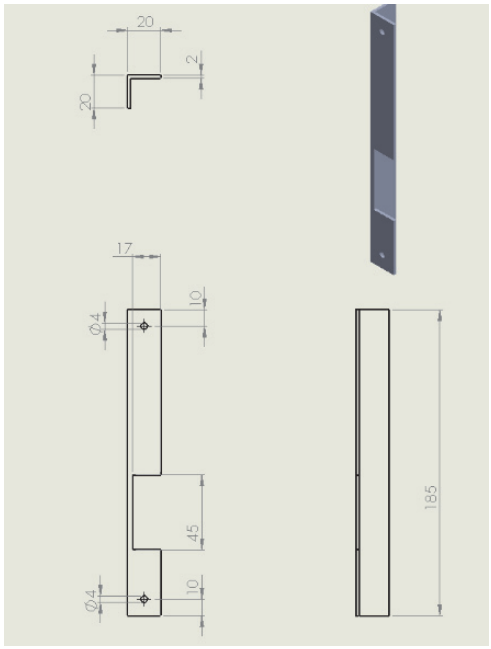

図 5-15 台座上面 枠部品②

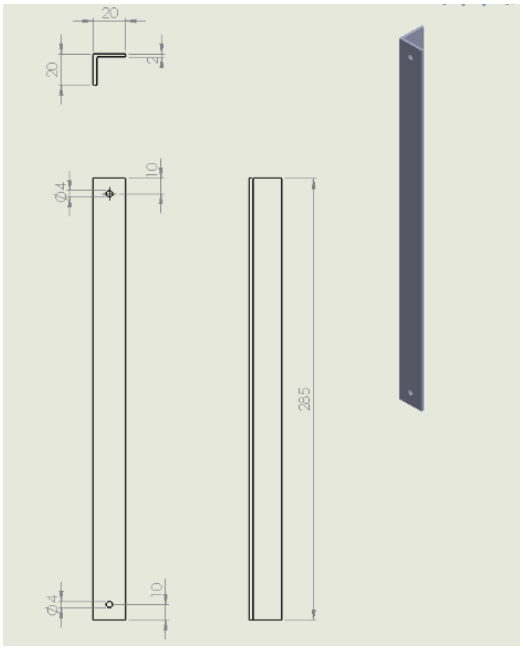

図 5-16 台座上面 枠部品③

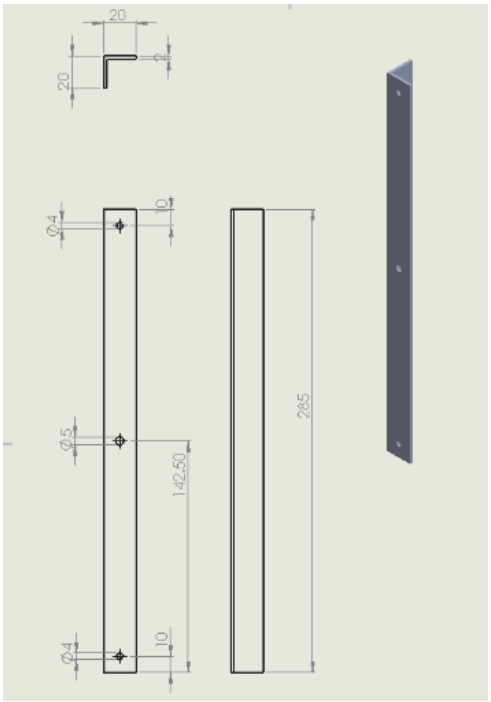

図 5-17 台座上面枠部品④

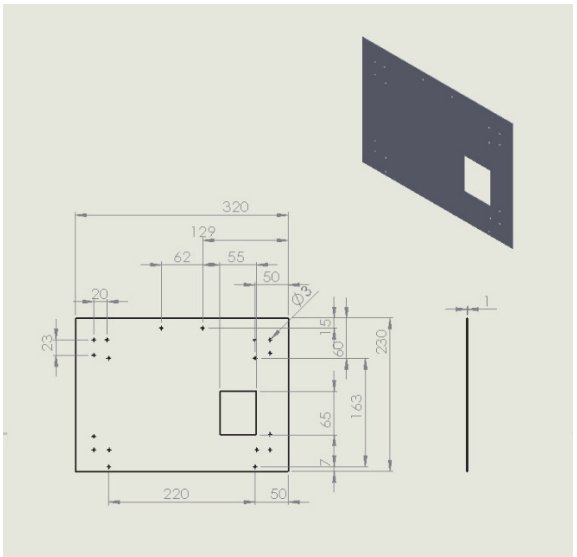

図 5-18 台座底面　アルミ板

5.2.4 Ovrvision Pro の固定部品の制作

次に Ovrvision Pro の固定部品の制作について説明する。

固定する部品の制作に使用した部品は表 5-2 の通りである。

表 5-2 Ovrvision Pro の固定部品に必要な部品

部品名	寸法	制作会社	個数
ミニ金具 HK-15	15×150	カメダデンキ	1
ミニ金具 LK-6	15×25×60	カメダデンキ	1
ST 特厚不等辺金折	2×15×2040	HIKARI	1
ゴム板	15×40×3	YAHATA	1
ゴム板	7×30×3	YAHATA	2
なべ小ねじ	M5×10mm	YAHATA	2
なべ小ねじ	M5×20mm	YAHATA	1

図 5-2、図 5-3 では Ovrvision Pro の後ろからボルトの締め付けで挟み込んで固定する予定であった。しかしミニ金具 HK-15 (図 5-19)、LK-6(図 5-20)を用いたことで後ろからの締め付けでは Ovrvision Pro を固定することができなかった。そこで図 5-5 のようにミニ金具 LK-6 とミニ金具 LK-15 で固定部品の土台を作り、ST 特厚不等辺金折と Ovrvision Pro にゴム板と張り付け、前からゴムで挟み込む方法に変更した。これにより、Ovrvision Pro の固定と高さ調節を両立できた。

図 5-19　ミニ金具 HK－15

図 5-20 ミニ金具 LK-6

図 5-21 と図 5-22 はゴム板を貼り付けた ST 特厚不等辺金折と Ovrvision Pro を示している。

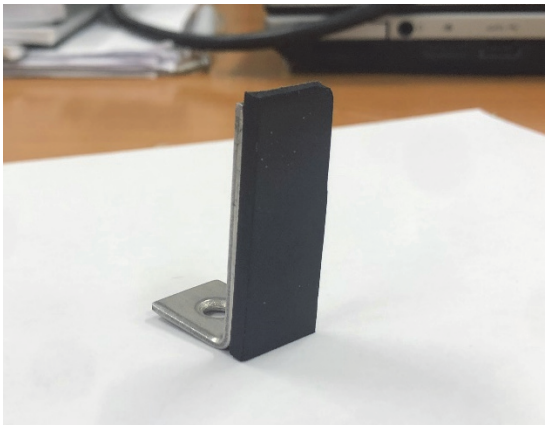

図 5-21 ST 特厚不等辺金折とゴム板

図 5-22　Ovrvision Pro とゴム板

図 5-23 は Ovrvision Pro を固定する部品の完成図である。

図 5-23　固定部品完成図

5.2.5 組立

最後に台座、Ovrvision Pro の固定部品とルンバを結合する。

台座と Ovrvision Pro の固定部品の組み立てにはボルトを用いる。

ルンバと台座の結合には図 5-25 のようにルンバとアルミ板の間に六角スペーサー（図 5-24）を挟み固定する。

図 5-24 六角スペーサー

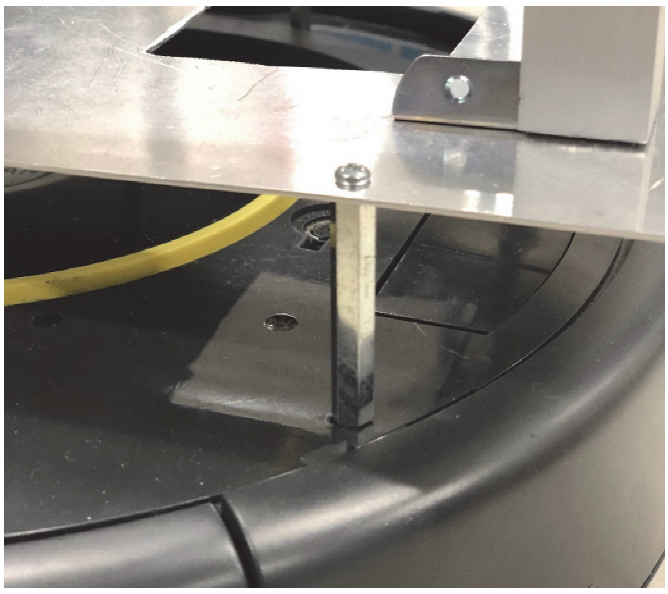

図 5-25 ルンバとアルミ板の結合部位

図 5-26 は組み立て後の完成した移動ロボットの写真である。

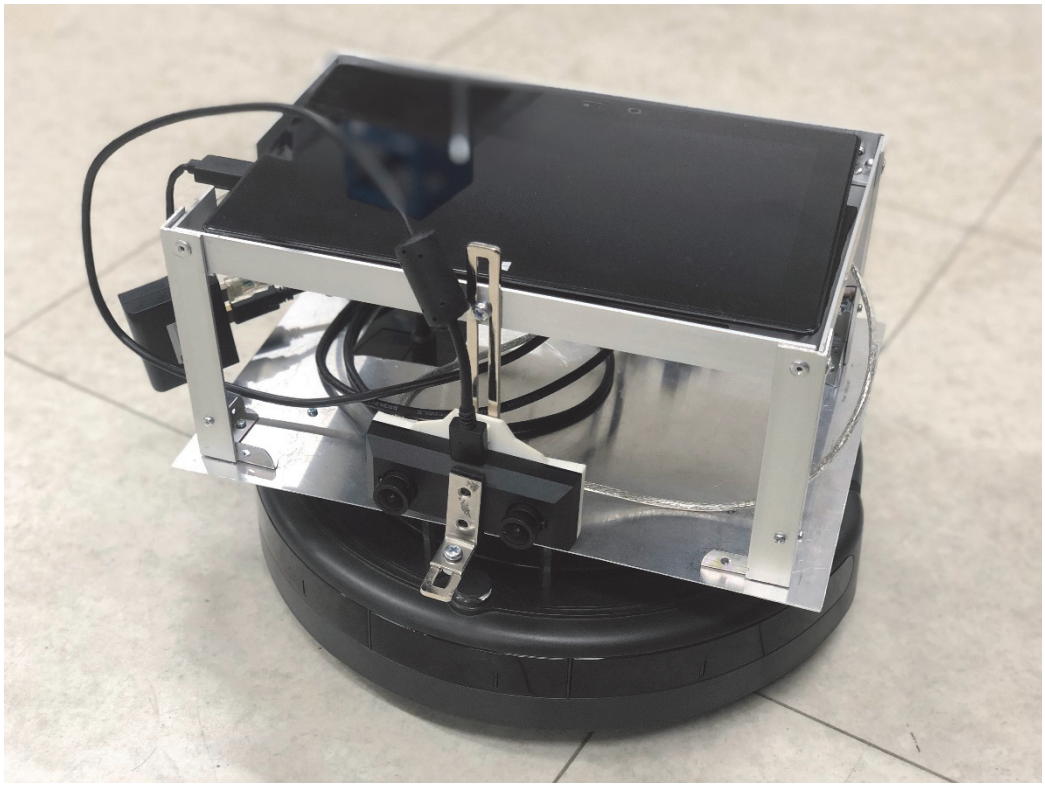

図 5-26 完成図

5.3 考察

要求機能や制約条件をすべて満たす制作物を作ることが出来た。また実際に制作してみると、予想していた以上の制作精度でできたため、しっかりと安定した移動ロボットを作ることが出来た。

第 6 章 動作確認（金子担当）

6.1 動作確認の目的

6.1.1 概要

本章では実際に Ovrvision Pro、Arduino Mega 2560、Surface Pro 2、ルンバを連携して動作確認を行うことで、要求機能が満たされていることをチェックする。

6.2 動作確認の評価方法

6.2.1 評価目標

移動ロボットの評価は機体、物体検出、距離計測、制御の要求機能をもとに行う。要求機能を表 6-1 に示す。

表 6-1 要求機能一覧

制作項目	要求機能
設計	・周辺パーツが安定している ・配線が移動の邪魔にならない
物体検出・距離計測	・目標を捉える ・距離計測が正確に行われる
制御	・出力された距離の値を基に移動・停止する ・回転移動により目標物の正面を向く

6.2.2 動作確認のロケーション

動作確認は研究室で行う。図 6-1 のように大まかな移動距離を把握するため、１ｍごとにマークする。また OpenCV を用いた Ovrvision Pro での距離測定の可能範囲が 200 ㎜～800 ㎜であることから、最大 2m の範囲内で動作確認を行うことにした。目標物は移動ロボットとの距離を１ｍ以上で、なおかつ移動ロボットの正面ではない位置に設置する。

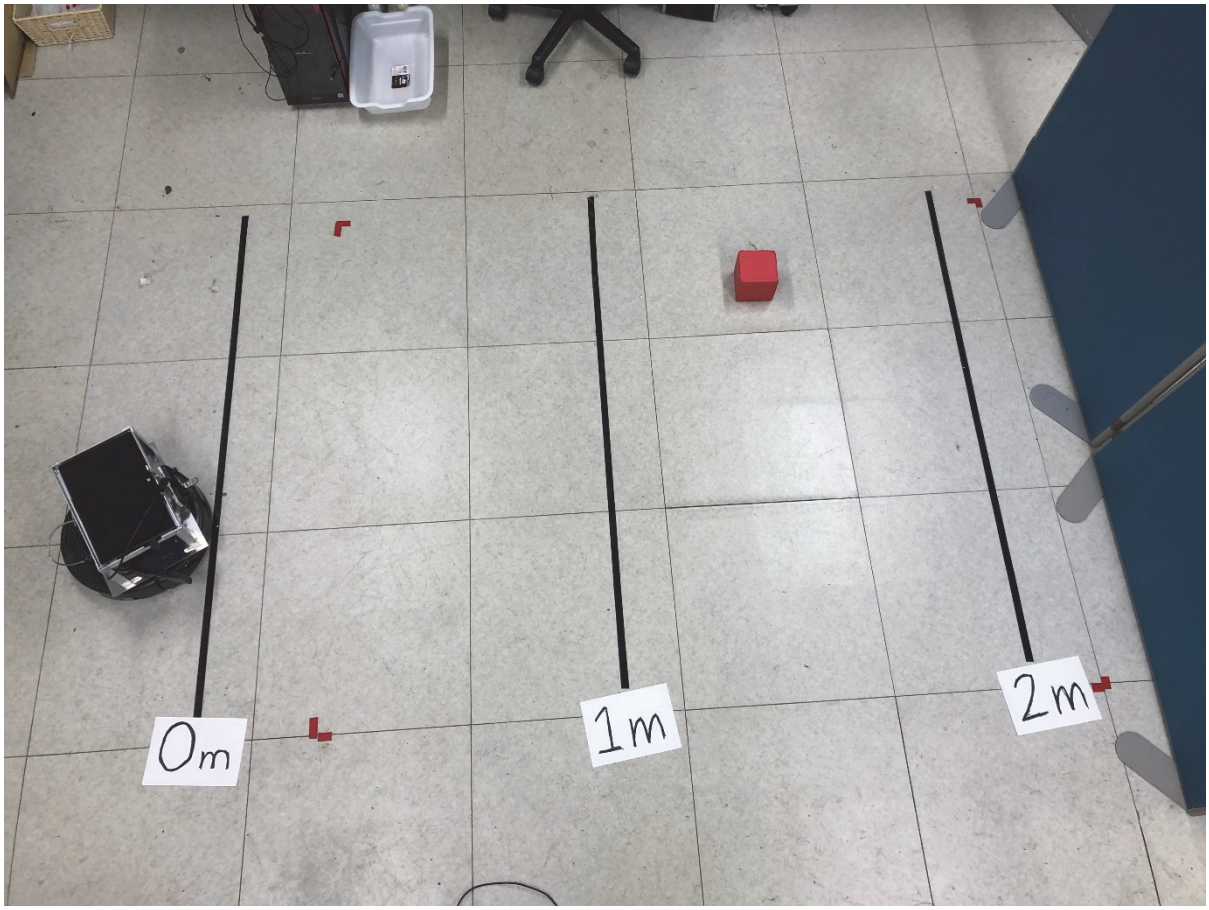

図 6-1 動作確認の準備

6.2.3 確認手順

動作確認は以下の手順で行う。

1. プログラムの起動を確認。
2. 物体検出が行われ、回転移動により目標物を移動ロボットの正面に捉えることが出来るかの確認。
3. 距離計測の算出が正常に行われていることの確認。算出された距離の値と実際の移動距離が一致するかを確認。
4. 目標地点での停止の確認。
5. 移動による影響で配線は邪魔になっていないか、機体が不安定になっていないか確認。

6.3 動作確認の結果

動作確認の結果を移動ロボットと Surface Pro 2 の出力画面と共に 6.2.3 の手順に沿って説明する。

1. プログラムを起動し、Surface Pro 2 上に Ovrvision Pro で取得した画像が出力されていることを確認するとともに、物体検出、距離計測のプログラムが正常に起動していることを確認した（図 6-2）。

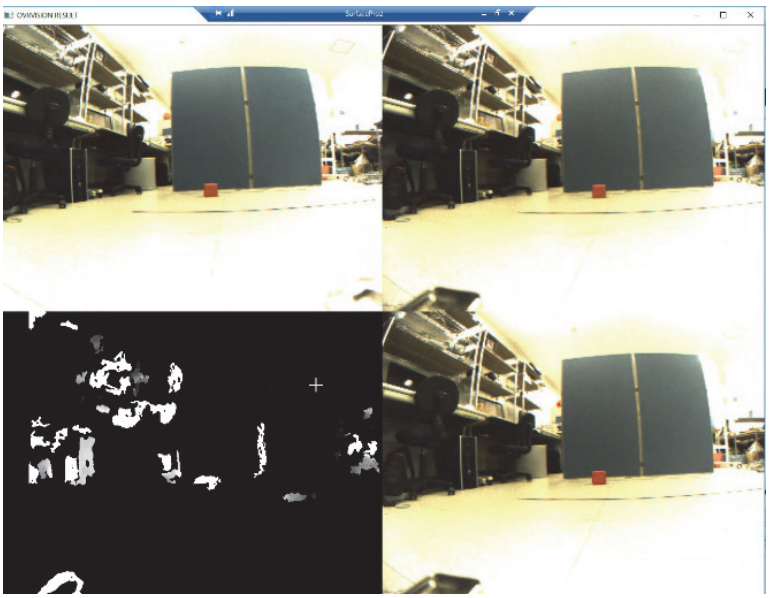

図 6-2　プログラム起動開始

2. 物体検出によって座標が取得されていることを確認した。また、Ovrvision Pro で取得した画像をもとに回転移動し、目標物を移動ロボットの正面に捉えたことを確認した（図 6-3、図 6-4)。

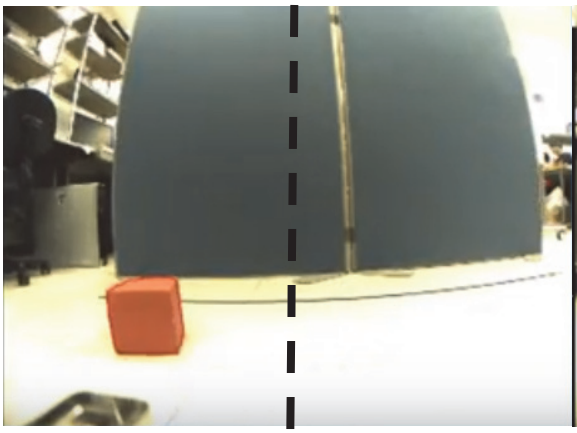

図 6-3 回転前

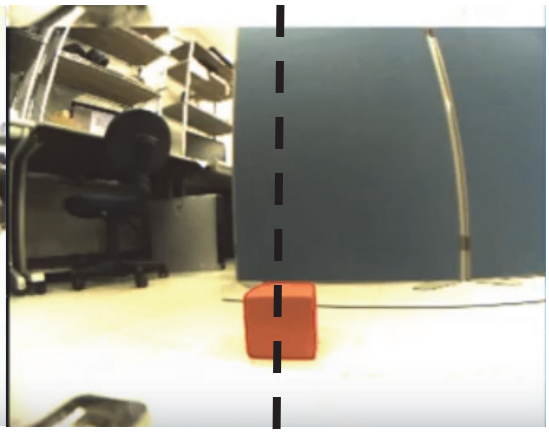

図 6-4 回転後

3. 距離測定によって目標物までの距離が算出されたことを確認した。しかし算出された距離の値は不安定なものだった。

4. 移動ロボットが目標地点で停止したことを確認した。

5. 配線が動作を妨げることはなく、移動中にもルンバの上の設置物が固定されていることを確認した。

6.4 評価

周辺パーツは回転移動、直進移動の影響も受けず固定され、安定していた。配線も走行の妨げになることはなかった。

3.2.3 で述べたとおり、デプス値は試行ごとに値の揺らぎが大きいため、算出された距離の値が不安定になった。

また利便性のある移動ロボットとして使用するには目標物に向かうまで時間がかかりすぎるため、移動速度や物体検出、方向転換などの各種動作をもっとスムーズにする必要がある。

以上から、距離計測と移動に関して要求を満たせなかった。

6.5 付加価値の提案

6.5.1 追尾機能の提案

我々が作成した物体検出は常時で目標物を認識できるため、移動ロボットの移動中に目標物が動いても対応することが出来る。

そのため、現状のプログラムで目標を追尾・追従することが可能である。

6.6 実験結果

6.6.1 実験方法

実験方法は 6.2.2 のロケーションで行い、図 6-5 のように人が目標を持ち、目標物に移動ロボットが近づいたときに目標物を移動させるという動作を繰り返すものである。

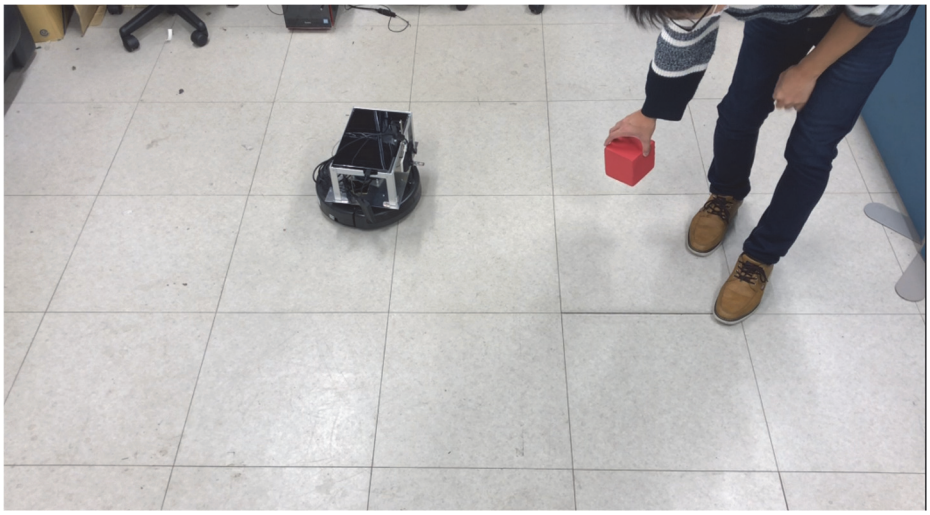

図 6-5 追尾・追従動作中

6.6.2 実験結果

実験の結果、目標を追従することが確認できた。しかし問題点として以下のことが上げられた。

・移動ロボットの移動速度が遅いため、移動ロボットより目標が速い場合追いつけない。
・直進移動中での目標の急激な方向転換に対応できず見失う。

これらの問題点を改善するために、移動速度は追従する目標物の移動速度に適切な速度設定をすることで改善されると考えられる。急激な方向転換に対応するためには現在では直進移動中に方向転換ができず、距離計測によって出力された距離を走行した後でないと、方向転換が行われなかったので、直進移動中に急停止し方向転換を行うプログラムを追加することで改善できると考える。

第 7 章 結論（松田担当）

7.1 目標

我々は Ovrvision Pro を用いて利便性のある移動ロボットを製作した。正確な物体の検出、目標物までの距離計測、ロボットの目標地点までの正確な移動と停止などといったシステムをステレオカメラのみを使って取り組んだ。

7.2 目標への達成度

設計の目標に対しては、マイコンボード、Surface Pro 2 を載せることが可能な台座を製作することで、急な前進後退や回転を行っても安定した動作を実現できた。

画像処理の目標に対しては、ステレオカメラにより取得した画像からデプス画像を作成し、距離を算出することができた。利便性のあるロボットとして追尾追従する機能を導入することができた。また、Ovrvision Pro の測定範囲に限度がある（200mm～800mm）ため、何度も探索を行い、目標物に向かうという手法を用いた。

ロボット制御では、Arduino IDE や Visual Studio 2017 を用いて、前進後退の動作の確認ができた。

7.3 総合評価

7.2 で述べたように、ステレオカメラを用いて物体検出や距離計測、追従追跡を行うという目標は達成できた。

しかし下記に示すようにいくつかの問題がある。

・デプス画像から得られる距離の精度が低く移動ロボットの目標設定に失敗することがある。

・何度も探索を繰り返すため移動に時間がかかる。

これらの問題を解決するために、より高性能なステレオカメラを用いれば探索回数や探索する時間は短縮でき、より効率よく作業が行えると考えられる。

第 8 章 参考文献・URL

[1-1]　Ovrvision Pro

http://ecx.images-amazon.com/images/I/51RVItvvXYL.jpg

[1-2]　アイサイト・安全 | フォレスター | SUBARU

https://www.subaru.jp/forester/forester/safety/safety2_1.html

[1-3]　ニューラルネットワークを組み込んだ ボッシュ のカメラモジュール…人とくるまのテクノロジー2018 | レスポンス (Response.jp)

https://response.jp/article/2018/05/23/310013.html

[1-4]　Ovrvision Pro | USB Open-Source Stereo Camera for AR/VR and Robot (Oculus Rift, HTC Vive)

http://ovrvision.com/

[1-5]　画像処理に基づく自動書類運搬装置の開発

https://brain.cc.kogakuin.ac.jp/research/201703_StereoThesis.pdf

[1-6]　パナソニックが自律搬送ロボット「HOSPI(R)」による空港、ホテルでのサービス実証実験を実施 | トピックス | Panasonic Newsroom Japan : パナソニック ニュースルーム ジャパン

https://news.panasonic.com/jp/topics/151281.html

[1-7]　自律搬送ロボット | ムラテック 村田機械

https://www.muratec.jp/corp/mono/robot.html

[1-8]　「ユーザーから"使いたい"と言ってもらえるロボットシステムを目指しています」～NEDO のステージゲートを通過・全方向移動自律搬送ロボット「MKR-003」を開発～【村田機械】 | ロボ・ステーション | J-Net21[中小企業ビジネス支援サイト]

http://j-net21.smrj.go.jp/develop/robot/entry/20090421-16.html

[2-1]RGB ウィキペディア (Wikipedia)

https://ja.wikipedia.org/wiki/RGB

[2-2]HSV 色空間ウィキペディア（Wikipedia）
https://ja.wikipedia.org/wiki/HSV%E8%89%B2%E7%A9%BA%E9%96%93

[3-1]　OpenCV と Visual C++による画像処理と認識（２０）----- Tsukuba 標準画像その他を使って視差マップ（disparity map）を作成する ----- -----
http://ishidate.my.coocan.jp/opencv_20/opencv_20.htm

[4-1]　うちのルンバに Twitter やらせてみた
http://kashiken.net/2016/08/02/%E3%81%86%E3%81%A1%E3%81%AE%E3%83%AB%E3%83%B3%E3%83%90%E3%81%ABtwitter%E3%82%84%E3%82%89%E3%81%9B%E3%81%A6%E3%81%BF%E3%81%9F/

謝辞

最後に、この場をお借りして本研究を進めるにあたり、研究テーマやステレオカメラに関して 2 年間ご指導ご鞭撻を頂きました金丸隆志准教授、研究において親身になり多くの助言を頂いた修士の先輩方、機構の設計や金属加工において快く対応していただいた 19 号館(ふらっと)の皆様方、協力していただいた皆様に心より感謝致します。これをもって謝辞とかえさせていただきます

ステレオカメラによる距離計測を用いた
移動ロボットの製作

指導教員　金丸 隆志　准教授
①S5-15001　相澤 優介　②S5-15019　柿沼 日向
③S5-15021　金子 悠　④S5-15058　松田 竜騎

1. 緒言

我々は「ステレオカメラを用いて利便性のある移動ロボットを作る」というテーマに取り組む.このテーマでは先行研究である 2016 年度卒業論文「画像処理に基づく自動運搬装置の開発」に続き,ステレオカメラを用いた移動ロボットの考案と製作を行うものである.先行研究に用いられたステレオカメラが自作のものであったのに対し,既製品のステレオカメラを用いた場合はどのような変化が得られるかを調べる.

ステレオカメラとは,被写体と同じ方向を向いた 2 つのカメラで撮影し物体との距離計測が可能なカメラである.本研究ではこのカメラに Ovrvision Pro を使用する.

本研究では利便性のある移動ロボットの製作にあたって「正確な物体の検出」,「ステレオカメラによる目標物までの距離計測」,「ロボットの目標地点までの正確な移動と停止」などの機能の実現を目指す.

2. 原理

本研究で製作する移動ロボットの製作には,移動手段に用いるハードウェアと物体認識に用いるソフトウェアの 2 つが必要である.移動手段については,お掃除ロボットのルンバを用いる.ルンバの制御に PC とマイコンボードを使用する.

物体認識には,「目標物を見つけること」と「目標物までの距離を計測すること」の 2 つの機能が必要である.これらを OpenCV による画像処理で実現する.

3. 移動ロボットの製作

製作した移動ロボットを図 1 に示す.PC とステレオカメラとマイコンボードを USB で接続し,マイコンボードとルンバを ROI 端子で接続する.PC ではステレオカメラから得た画像を基に目標物までの場所と距離を算出し,その情報をマイコンボードに送信する.マイコンボードは PC から受け取った情報を用いてルンバを制御する.

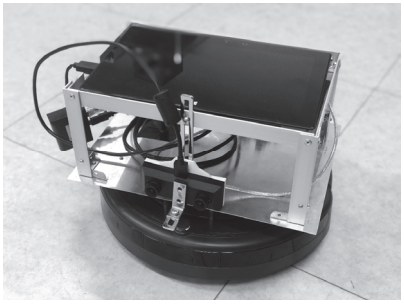

図 1　製作した移動ロボット

4. 実験

作成したプログラムを用いて,動作確認を行う.図 2 のように目標物を置きプログラムの起動,ハードウェアの接続,距離計測,目標地点での停止を確認する.距離の測定や進行方向の決定には多少の誤差があるものの,移動自体は正常に行い目標物の前まで移動し停止することができた.

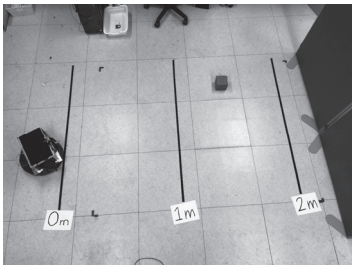

図 2　動作確認

次に追尾機能の確認を行う.図 3 のように人が目標物を持ち,それに移動ロボットが近づいたときに移動させるという動作を繰り返す.結果,目標を追従することが確認できた.しかし移動ロボットの移動速度が遅いため,目標が速い場合追いつけないことがあった.また移動中の目標物の急激な方向転換に対応できず見失うこともあった.

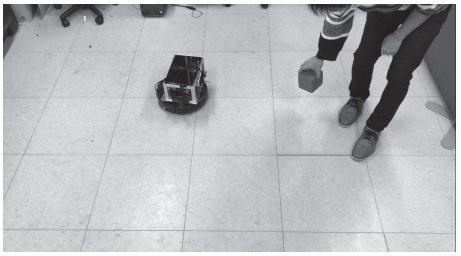

図 3　追尾機能の確認

5. 結言

(1) ステレオカメラで距離計測する移動ロボットを制作できた.
(2) マイコンボードとルンバを接続し,シリアル通信による移動を実現できた.
(3) 対象物の認識と距離計測のプログラムを作成できた.

〈参考文献〉
池 宏武・ほか 6 名,“画像処理に基づく自動書類運搬装置の開発” 工学院大学卒業論文,2017 年 3 月