2016 年度（平成 28 年度）

ECPⅢ
Final Report

画像処理に基づく自動書類運搬装置の開発

チーフアドバイザー　：　金丸　隆志　准教授

サブアドバイザー　：　矢崎　敬人　准教授

チームメンバー

G1-13007　池　宏武

G1-13033　境　亮祐

G1-13049　中川　勇佑

G1-13063　布施　勇樹

G1-13064　堀井　孝亮

G1-13078　米山　耕介

G1-12009　石橋　知大

目次

第 1 章 緒言（布施・堀井担当）.....1
1.1 研究背景（堀井担当）.....1
1.1.1 研究テーマ.....1
1.1.2 車載カメラについて.....1
1.1.3 ステレオカメラ.....2
1.1.4 他のセンサとの比較.....3
1.2 研究背景（堀井・布施担当）.....4
1.2.1 製作物の決定（堀井担当）.....4
1.2.2 他社製品との比較（布施担当）.....5
1.2.3 研究の目的（堀井担当）.....8
1.2.4 研究の進行（堀井担当）.....9
第 2 章 設計（境担当）.....10
2.1 概要.....10
2.1.1 目的.....10
2.1.2 要求機能・制約条件.....10
2.2 自動書類運搬装置の製作.....11
2.2.1 アイディア.....11
2.2.2 ステレオカメラの設計.....12
2.2.3 自動書類運搬装置の構成要素.....24
2.2.4 機体仕様.....27
2.3 機体の完成.....28
2.3.1 動作確認.....28
2.3.2 考察.....29
第 3 章 目印認識と距離計測（池・中川・堀井担当）.....30
3.1 概要（堀井担当）.....30
3.1.1 目的.....30
3.1.2 要求仕様.....30
3.1.3 使用したソフトウェア.....31
3.1.4 プログラムのフロー.....32
3.2 使用する目印(中川担当).....33
3.2.1 概要.....33
3.2.2 目印の作成.....33
3.2.3 実験当初の目印.....34
3.2.4 実験最終段階の目印.....35

3.2.5 実験最終段階の目印でのルンバの改良 37
3.3 色認識による目印認識（池担当） 39
3.3.1 概要 39
3.3.22 値化するプログラムの作成 39
3.3.3 面積計算するプログラムの作成 40
3.3.4 特定範囲以外の面積を除外するプログラムの作成 41
3.3.5 目印を色でとらえる 41
3.4 目印を捉える実験 44
3.4.1 概要 44
3.4.2 実験結果と考察 45
3.5 距離計測の方法（堀井担当） 46
3.5.1 概要 46
3.5.2 原理・理論 46
3.5.3 視差画像の作成 48
3.5.4 視差の読み取り 51
3.5.5 動作確認テスト 52
3.5.6 動作確認の結果 53
3.5.7 考察 54
3.6 レンズ間距離と測定可能範囲の関係（堀井担当） 55
3.6.1 概要 55
3.6.2 測定手順 55
3.6.3 結果 56
3.6.4 考察 60
3.7 距離の計測（堀井担当） 61
3.7.1 概要 61
3.7.2 精度の目標 61
3.7.3 関数の作成方法 62
3.7.4 結果 63
3.7.5 考察 65
3.8 目印認識と距離計測の最終評価（池・堀井担当） 66
3.8.1 要求仕様との比較 66
3.8.2 考察 67
第 4 章 回路とそのプログラム（米山・石橋担当） 68
4.1 概要（石橋担当） 68
4.1.1 目的 68
4.1.2 要求機能 68

4.2 ルンバの接続と制御（米山・石橋担当）.....69
4.2.1 要求機能達成のための方法.....69
4.2.2 使用機器.....69
4.2.3 オープンインターフェース.....70
4.2.4 ルンバとの接続方法.....71
4.2.5 制御方法.....72
4.2.6 ルンバの動作テスト.....83
4.3 回路とそのプログラムの最終評価.....84
第 5 章 自動書類運搬装置の動作確認(中川担当).....85
5.1 概要.....85
5.2 実験方法.....86
5.3 実験結果.....87
5.4 考察.....88
第 6 章 結論（堀井担当）.....89
6.1 目標.....89
6.2 目標への達成度.....89
6.3 問題.....90
6.3.1 問題点.....90
6.3.2 解決法.....91
6.4 総合評価.....91
第 7 章 参考文献・URL.....92
謝辞.....94

第 1 章 緒言（布施・堀井担当）

1.1 研究背景（堀井担当）

1.1.1 研究テーマ

我々はＫ社から提供された「続・車載カメラをこんなものに付けたら」というテーマに取り組む。このテーマに取り組むにあたり、研究で使用するためのレンズを提供された。提供されたレンズは、車載のステレオカメラ用のものであるため、ステレオカメラによる距離計測を用いた応用例を模索していく。

1.1.2 車載カメラについて

本節では車載カメラについて解説する。

車載カメラは、用いられるレンズにより用途が分類される。主に用いられているレンズは以下の 2 つである。

1. 広角レンズ （焦点距離 24～35㎜）
2. 標準画角レンズ （焦点距離 50㎜ 前後）

1 のレンズは、画角が広いため、広範囲を確認することに適している。2 のレンズは、1 と比べ撮影できる範囲が狭いが、その分画像が鮮明になる。そのため、画像から人や車を検知したり、ステレオカメラを用いて距離計測をしたりするような、画像処理を施すことに適している。

今回我々が使用するステレオカメラ用のレンズはこの 2 つのレンズの内、2 の標準画角のレンズに分類される。

また、ステレオカメラについては次節 1.1.3 項で解説する。

1.1.3 ステレオカメラ

本項ではステレオカメラについて解説する。

ステレオカメラとは、図 1-1 のように 2 個のレンズを左右に並べ、同一の被写体を同時に 2 枚の画像に収めることのできるカメラのことである。この 2 枚の画像からステレオ法による距離計測を行うことで、ステレオカメラは距離計測用のセンサとして用いることができる。スバルのアイサイトはこの方法で前方の対象物までの距離計測を行っている。ステレオ法については第 3 章で解説する。

図 1-1　アイサイト［1-1］

1.1.4 他のセンサとの比較

1.1.3 項では、距離計測用センサとしてのステレオカメラついて触れた。本項では、他の一般的に使われている距離計測を行うセンサについて述べ、そのセンサとステレオカメラを比較する。

距離計測に用いられている一般的なセンサに測域センサがある。測域センサとは図 1-2 のような外観をしており、180°以上の周囲にレーザー光を放ち、光を当てた周囲の障害物や壁までの距離を測るセンサである。壁や障害物を 2 次元的に認識し、障害物までの距離を正確に計測することが可能である。しかし、測域センサは距離の情報しか得られないため、その障害物が何であるかまでは認識することが出来ない。更に、測域センサは非常に高価であり、例えば図 1-2 の「HOKUYO レーザー式測域センサ UXM-30LX-EW」の価格は 528,000 円である。

一方、ステレオカメラによる距離計測は、カメラを利用するため、障害物までの距離だけでなく色まで認識することが可能である。更に、カメラ自体はそこまで高価ではないため、製作も安価に済ませることが出来る。しかし、測域センサのように 180°以上の視野はなく、前方の測定しか出来ない。また、測域センサと比べると、測定した距離の精度が劣るのも欠点である。

測域センサとステレオカメラの性能をまとめると表 1-1 のようになる。

表 1-1　測域センサとステレオカメラの性能の違い

	測域センサ	ステレオカメラ
距離計測方法	180°以上の周囲にレーザー光照射	両眼視差を利用
障害物の情報	距離まで	距離、色まで認識可能
距離の正確さ	精密に測定出来る	あまり精密に測れない
視野	180°以上の周囲を測定可能	前方の物しか測定出来ない
価格	高価	安価

図 1-2　レーザー式測域センサ UXM-30LX-EW [1-2]

1.2 研究背景（堀井・布施担当）

1.2.1 製作物の決定（堀井担当）

前節で述べた通り、ステレオカメラは画像と距離の情報を得られることが可能である。この働きは、人間の目に似ている。我々は、このことに着眼し、ステレオカメラ以外に人間の手足となるものと、考える脳となるものがあれば、人間のように自動で走行する装置が製作できるのではないかと考えた。

そこで、我々が提案するものは「自動書類運搬装置」というものである。これは、書類のような荷物を運ぶ作業を、人間の代わりに行うシステムのことである。具体的な使用状況は以下の通りである。

・使用場所　　会社のオフィス。
・使用目的　　デスク間の書類などの受け渡しの簡略化のため、書類等の荷物を運搬すること。
・運搬する荷物　書類・軽い荷物（重制限量 5kg）。

広いオフィスの中には学校の体育館程の広さのものがあり、そのようなオフィスでは、例えば課長や部長などに書類を提出する際にかなりの距離を移動しなくてはならない。それにより、一時仕事が中断されてしまうという問題点がある。

そこで、オフィスで書類等を運搬するロボットを製作し、受け渡しの簡略化を目指すことにした。

なお、書類の運搬が目的なので、5kg という重量制限は機能上大きな問題にならないだろう。

1.2.2 他社製品との比較（布施担当）

本項では、我々が製作する自動書類運搬装置と既に市場に出ている他社の自走ロボットを比較する。

まず、アメリカの Simbe Robotics が開発した「Tally」である。Tally とは、商品陳列棚の状態をカメラで撮影し、商品の品切れや配置の間違い、値札の付け間違い、陳列の乱れなどを見つけ出すことの出来る自走ロボットである(図 1-3)。本体の側面にはカメラが二つ取り付けられており、一回で左右にある商品陳列棚を同時に撮影する事が可能である。Tally は自身が作成した店内の 3D マップを基に自走するため、あらかじめ 3D マップを作成しておく必要がある。また、障害物や人を検出するため、距離センサや音響センサなどのセンサを使用している。

図 1-3　Tally の概観 [1-3]

アメリカの Amazon Robotics が開発したのは、Amazon の倉庫で使われている自動運搬ロボットの「ドライブ」である(図 1-4)。ドライブは、商品棚を自動で特定の位置まで移動させることのできるロボットである。ドライブは人が入ることのできない柵で囲まれた範囲内でのみ動作し、床に一定間隔に配置された 2 次元コードを読み取りながら、自身の位置を認識し移動する。それぞれが独立して動いているのではなく、無線の Wi-Fi によってトラフィック制御されているため、ぶつかることはない。また、ドライブは運搬作業を通して適応化され、例えば人気製品は近くに、そうでないものは倉庫の奥に移動するなどの機能も備わっている。

図 1-4　ドライブの概観［1-4］

最後に、日本の Cerevo が開発した「Tipron」を紹介する(図 1-5)。Tipron とは、どんな壁のどんなところにもプロジェクターを写すことができる自走ロボットである。カメラと深度センサを備えており、今自分は部屋のどこにいるか、どこからまたはどの角度でプロジェクターを映せばよいのかを計算し移動することができる。Tipron はスマートフォンのアプリで動く仕組みになっており、その部屋のレイアウトをあらかじめスマートフォンを使って覚えさせておく必要がある。

図 1-5　Tipron の概要［1-5］

我々の自動書類運搬装置とこれら三つの自走ロボットの移動方法、目的、使用しているセンサで比較したものが表 1-3 である。

表 1-3　三つのロボットとの比較表

	自動書類運搬装置	Tally	ドライブ	Tipron
移動方法	目印を見つけ辿る	マップを作成する	2次元コードを辿る	マップを覚えさせる
目的	オフィスでの書類運搬	商品棚の品切れ等の確認	倉庫でのドライブ用棚の運搬	指定した場所へプロジェクターを写す
使用しているセンサ等	ステレオカメラ	イメージセンサ	2次元コードリーダー	深度センサ
		LIDAR		
		距離画像センサ		
		音響センサ		

表 1-3 のようにステレオカメラを利用して自走する市販のロボットはあまり存在していない。これは、ステレオカメラの欠点である近距離を認知しにくい点や測域センサより精度が低い点が原因であると考えられる。

その一方で、ステレオカメラは測域センサには不可能な色や形などを認識することが可能であるというメリットがある。また、距離を計測するセンサは大変高価であるのに対し、カメラのみを使用する我々の自動書類運搬装置は安価である特徴もある。

このことから、我々はステレオカメラで周囲を認識し、自走する新たな装置を製作できると考えた。

1.2.3 研究の目的（堀井担当）

自動書類運搬装置の実現に必要なものは、移動手段(ハードウェア)と対象認識(ソフトウェア)の二つである。

一つ目の移動手段に関しては、我々は家庭用お掃除ロボットルンバを使用することにした。ルンバは自作プログラムによる制御の仕様が公開されているためである。ルンバの制御には、PC とマイコンボードである Arduino を使用する。

二つ目の対象認識とは、「目的地を見つけること」および「目的地までの距離の把握」を行う手段のことである。目的地の距離の把握には、K 社から提供されるレンズを用いて製作するステレオカメラを用いる。また、目的地には色分けされた目印を設置するため、ステレオカメラの一方のカメラを利用して、色認識を行うことで目的地を見つけることができる。これらを実現することで、自動書類運搬装置は図 1-6 のようなイメージで機能する。まず、装置にはあらかじめ回るべき色が格納されている。例えば赤→青→緑などである。まず、装置は色認識により赤い目印を見つけ、さらにステレオカメラによりその目印までの距離を計測する。その距離をルンバが移動し、移動し終えたらルンバはその場で旋回して次の目印を見つけ、以下、受け渡しの目印に移動するまで繰り返す。この動作を実現するが本研究の目的である。

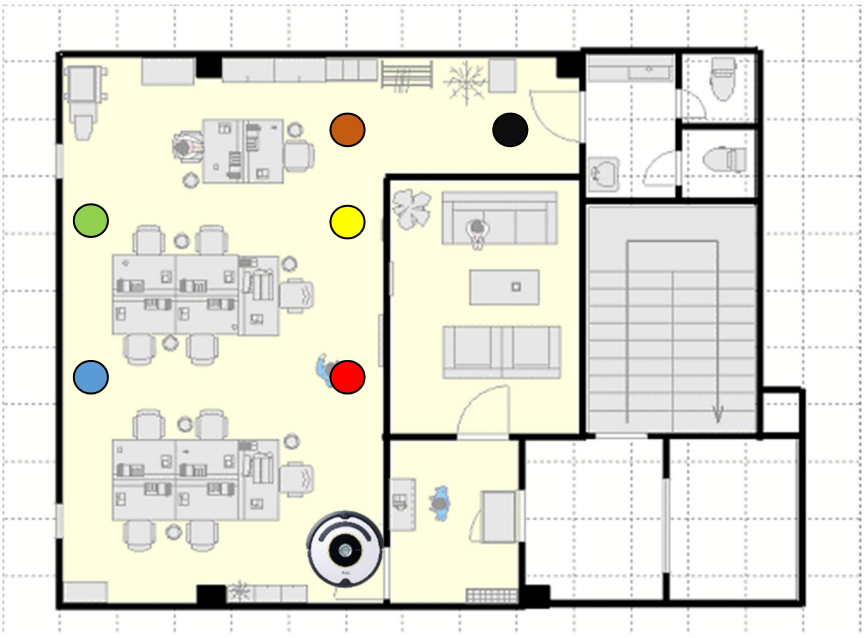

図 1-6　移動のイメージ

1.2.4 研究の進行（堀井担当）

我々は自動書類運搬装置を実現するにあたり、表 1-3 のようなチームに分かれた。

1 つ目の班は設計班である。設計班の役割は以下の 3 つである。

1. 距離計測を実現するためのステレオカメラの製作。
2. 自動書類運搬装置の外観の考案と製作。
3. 製作したステレオカメラ、収納部、及びルンバとの接合。

2 つ目の班は画像処理班である。画像処理班の役割は以下の 3 つである。

1. 目的地となる目印の製作。
2. 色認識を用いた目的地の把握。
3. ステレオ法を用いるのに適したレンズ間距離の算出。
4. ステレオ法を用いた自動書類運搬装置と目印までの距離の測量。

3 つ目の班はルンバ制御班である。ルンバ制御班の役割は以下の 3 つである。

1. PC、Arduino、及びルンバの接続とシリアル通信。
2. Arduino を用いたルンバ内のモータを操作、及びルンバの移動の実現。
3. ルンバの移動距離と旋回角度の取得。

上記の 3 班がそれぞれ作業を進行し、自動書類運運搬装置の完成を目指す。

表 1-3　班構成

班構成		
設計班	**画像処理班**	**ルンバ制御班**
境　亮介	池　宏武	布施　勇樹
	○中川　勇佑	米山　耕介
	◎堀井　孝亮	石橋　知大

◎：班長　○：副班長

第 2 章 設計（境担当）

2.1 概要

2.1.1 目的

本章では自動書類運搬装置の設計について解説する。

機体の設計ではルンバを用いて書類を運搬する装置の製作を行った。また、ステレオ法を実現するために 3.5 節で用いるステレオカメラの製作も行った。

2.1.2 要求機能・制約条件

自動書類運搬装置を製作するにあたって、要求される機能を以下に示す。

1. 距離計測を実現するためのステレオカメラの製作。
2. 自動書類運搬装置の外観の考察と製作。
3. 製作したステレオカメラ、収納部、及びルンバとの接合。

次にステレオカメラの制約条件を以下に示す。

1. レンズの固定部分が安定していること。
2. 整備がしやすいこと。
3. 自動書類運搬装置に取り付けやすい構造であること。

これらを踏まえ研究を進めていく。

2.2 自動書類運搬装置の製作

2.2.1 アイディア

本格的な製作に入る前に、2.1.2 項の要求を満たす自動書類運搬装置の構造についてアイディアを出した結果、図 2-1 のようになった。

下段には移動手段のルンバ、中段には書類を入れるための棚、上段にはステレオカメラが配置されている。このイメージを目指し、設計を行う。

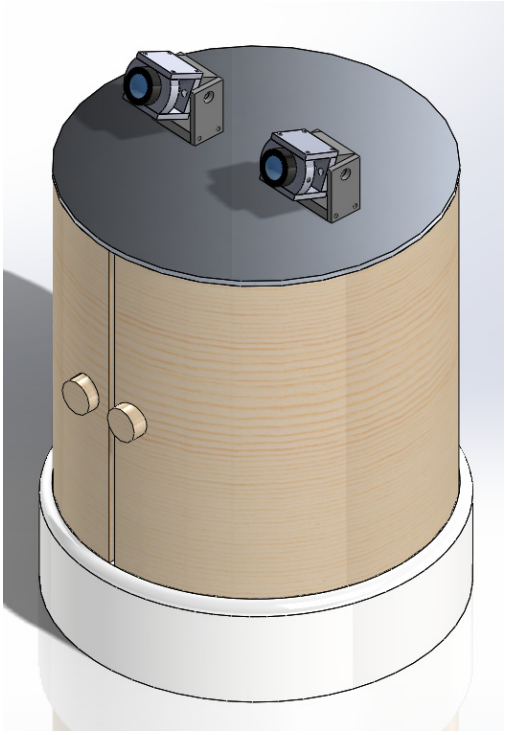

図 2-1　SolidWorks によるイメージ

2.2.2 ステレオカメラの設計

本節ではリエゾンから提供されたレンズを用いて、製作したステレオカメラについて解説していく。

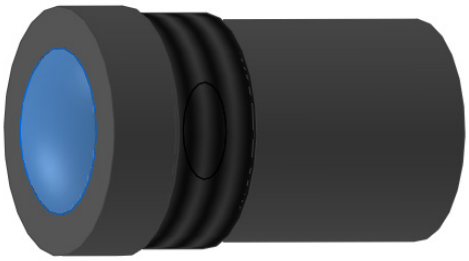

図 2-2　レンズ外観のイメージ

レンズと共に用いるセンサとして、Logicool 社の WEB カメラ C920 に用いられているものを使用した。

図 2-3　Logicool 社 C920［2-1］

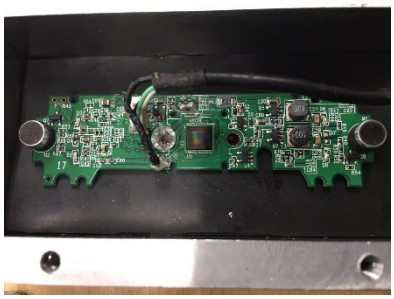

図 2-4　C920 のセンサ基板

2.2.2.1 既存のステレオカメラの改良

前述のステレオカメラを製作するにあたり、研究室で過去に製作された図 2-5 のステレオカメラを参考にすることにした。

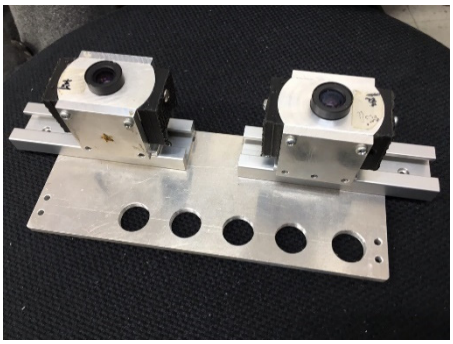

図 2-5　研究室で過去に製作されたステレオカメラ

このステレオカメラには以下の問題点があった。

1. 2 つのカメラがスライドレールよって独立して左右に動かせる構造のため、2 つのカメラが上下にぶれやすい(図 2-6)。
2. ステレオカメラに足がないため、レンズを垂直にし続けることが難しい。

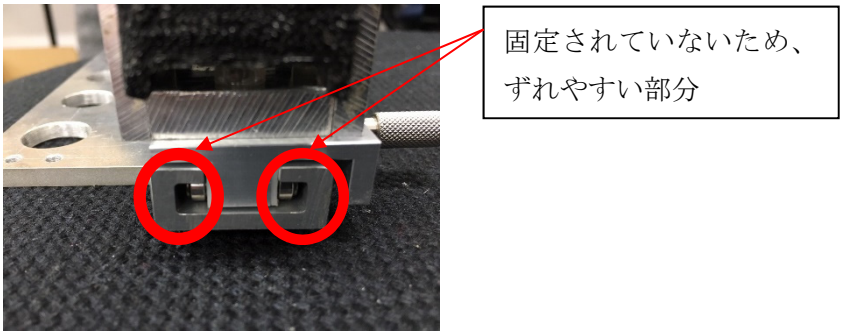

図 2-6　スライドレール

この問題点を解決した新しいステレオカメラの製作を目指す。

2.2.2.2 問題改善

本節では、新しく製作したステレオカメラについて解説する。2.2.2.1 で触れた上下にぶれやすいという問題点を解決するため、スライドレールは使用せず左右のレンズを１つの直方体に取り付ける形にした。この際、レンズ間の距離は 3.6 節で決めた 105mm とする。

加えて図 2-7 のような形状にすることでカメラを垂直に立たせることが可能となり、自動書類運搬装置への設置が行いやすくなった。

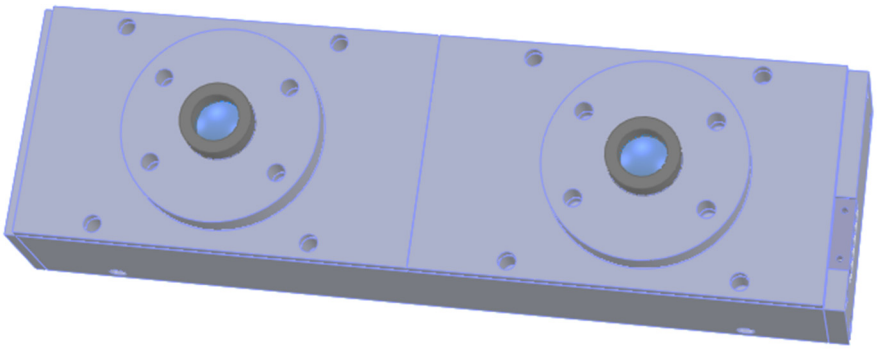

図 2-7　SolidWorks で新たに設計したステレオカメラ

なおレンズ固定用のアルミ板金を、図 2-7 のように左右それぞれ別にしている。これは、左右のレンズとセンサを独立して調節できるようにする事で、メンテナンスを容易にするためである(図 2-8)。

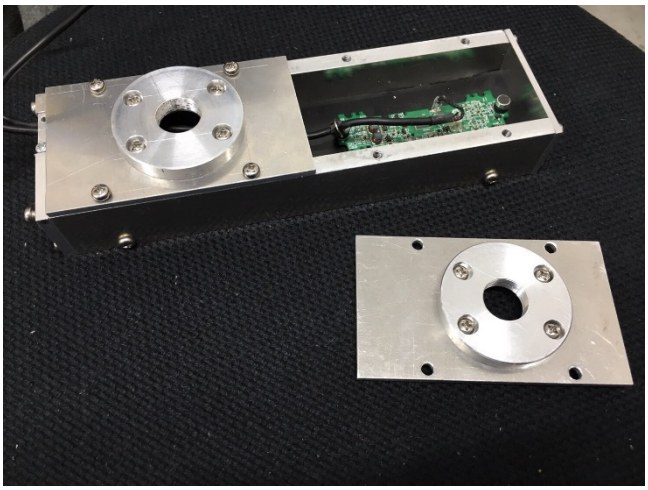

図 2-8　ステレオカメラ上面の分割

センサ基板を固定するためには、ゴムシートをセンサ基板とアルミ板金の間に敷き、そこにネジで取り付けるようにした(図 2-9)。これにより、回路にかかる負担を減らした。

図 2-9　センサ基板を固定した様子

センサ基板から映像が出力されるコードの固定部を拡大した図が、図 2-10 である。センサ基板と同様に、ゴムシートで挟んで固定するようにしている。このコード固定部を新しいステレオカメラの側面に取り付けることで、センサ基板とケーブルの接続部分に負担がかかることを防いだ。

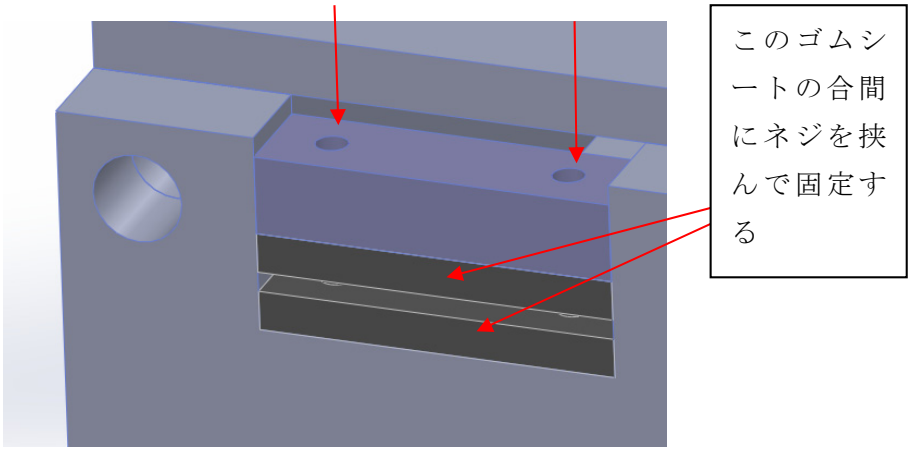

図 2-10　コード固定部

しかし、ゴムシートが予想以上に硬かったため、この大きさの隙間ではゴムシートでコードを挟むことが出来なかった。コード固定部のための隙間をこれ以上大きくすると、外からの光が内部の回路のセンサに当たり、正確な実験が行えなくなる。そのため図 2-11 のようにコード固定の端にコードを通し、隙間にゴムシートを入れ、片方をネジで固定する事でコードを固定する事が出来た。しかし、この固定ではコードや部品にかかる負担が大きく改善が必要である。

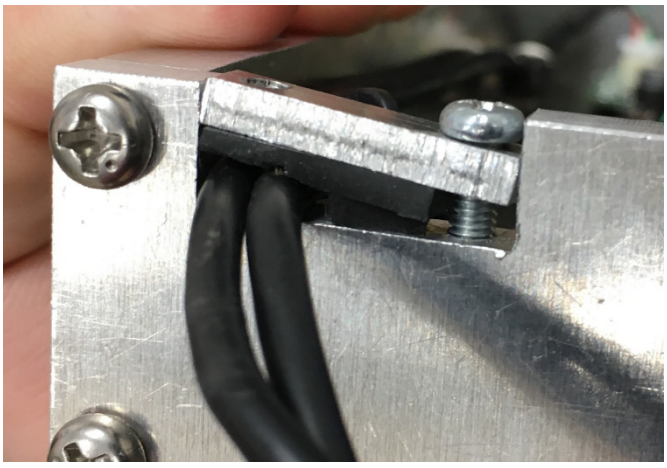

図 2-11　実際のコード固定部

2.2.2.3 完成したステレオカメラの詳細な構造

本節では完成したステレオカメラの構成について、図 2-12 と図 2-13 を用いて解説する。図 2-12 には、下記の 4 つの部品が示されている。

1. レンズ
2. レンズ固定
3. ステレオカメラ上面　その 1
4. ステレオカメラ上面　その 2

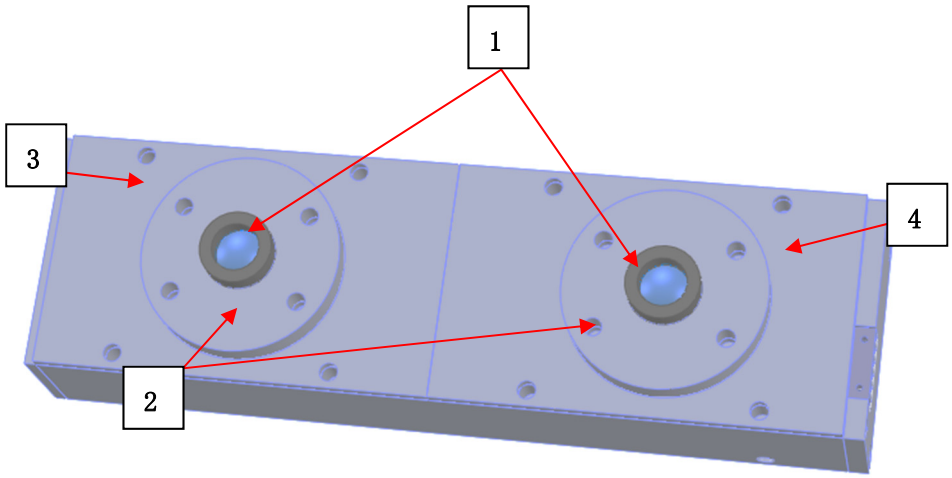

図 2-12　ステレオカメラ外観

図 2-12 から 3 と 4 の上面のアルミ板金を取り外すと図 2-13 のようになる。図 2-13 には下記の 5 つの部品が示されている。

5. ステレオカメラ側面
6. ステレオカメラ側面小　その 1
7. ステレオカメラ側面小　その 2
8. ステレオカメラ底面
9. コード固定

これらのパーツを固定するため、ネジとナットを使用した。

使用したものは主にネジ(M4×10)で、コード固定部はネジ(M3×10)、回路固定にはアクリルネジ(M3×10)を使用した。

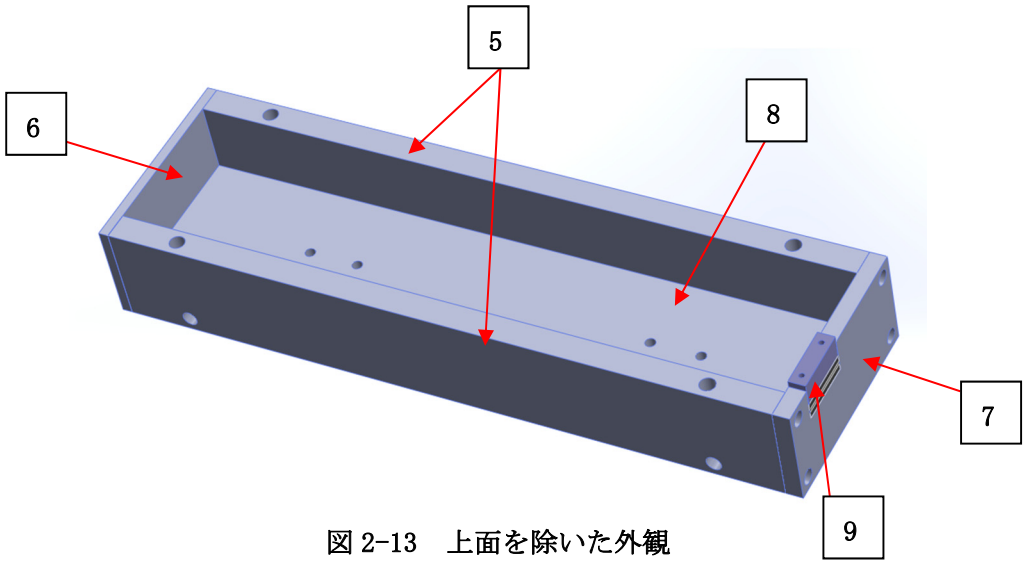

図 2-13　上面を除いた外観

これらのパーツの構造設計図は、3 次元 CAD ソフトウェアである SolidWorks を用いて作成した。

使用した設計図を図 2-14～図 2-20 で示す。

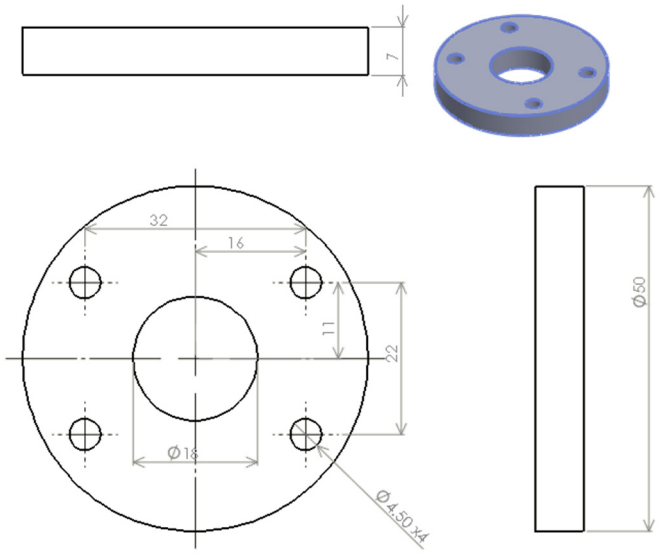

図 2-14　レンズ固定

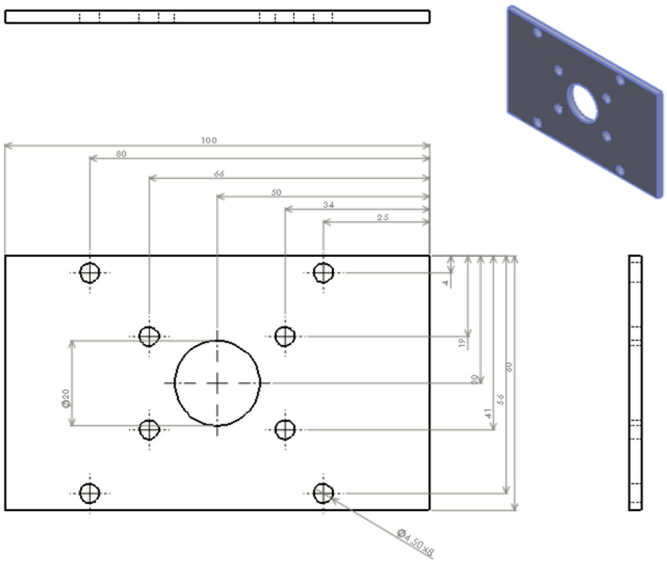

図 2-15　ステレオカメラ上面　その 1

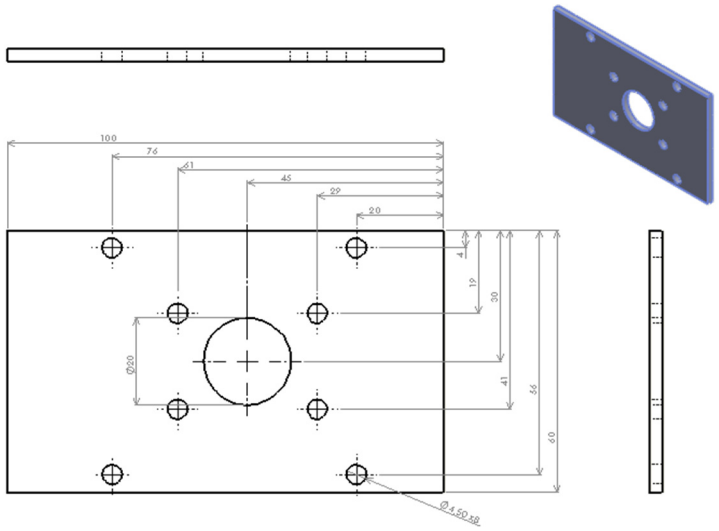

図 2-16　ステレオカメラ上面　その 2

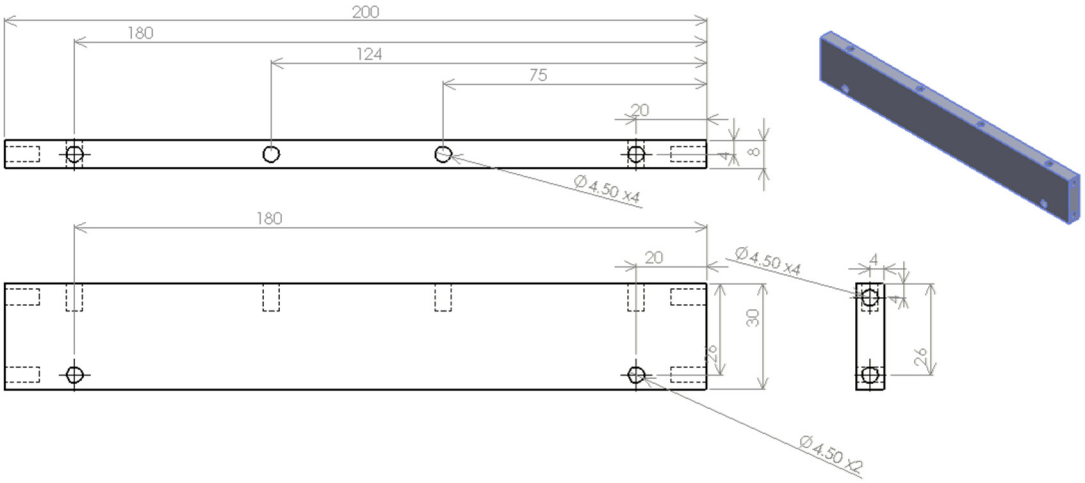

図 2-17　ステレオカメラ側面

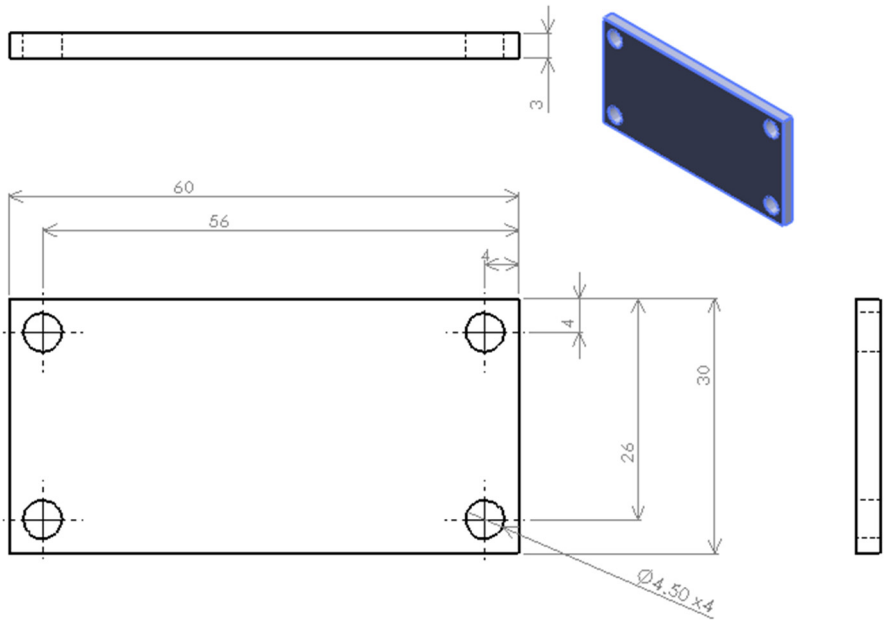

図 2-18　ステレオカメラ側面小　その 1

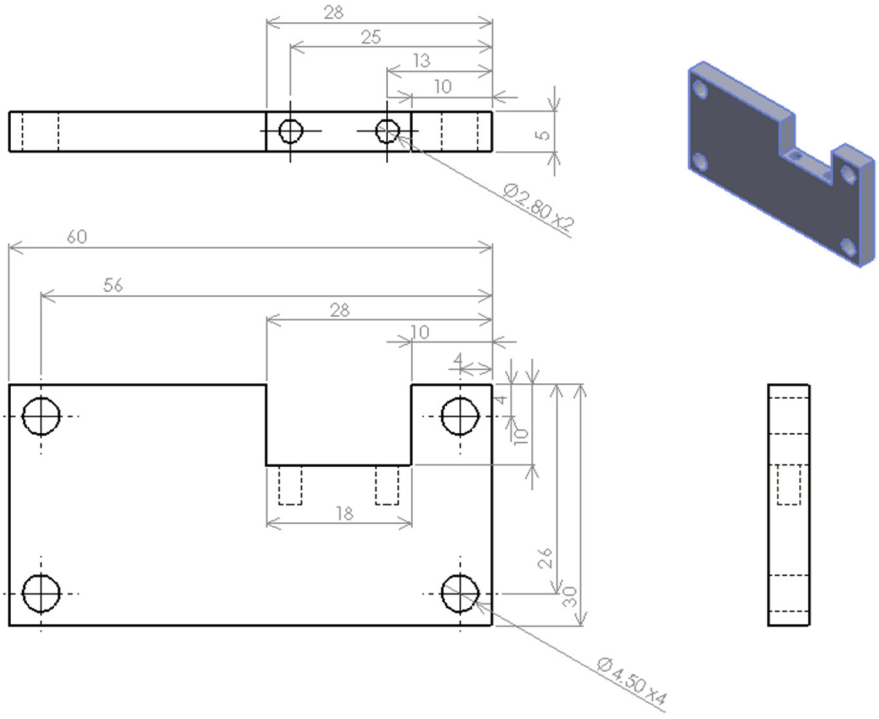

図 2-19　ステレオカメラ側面小　その 2

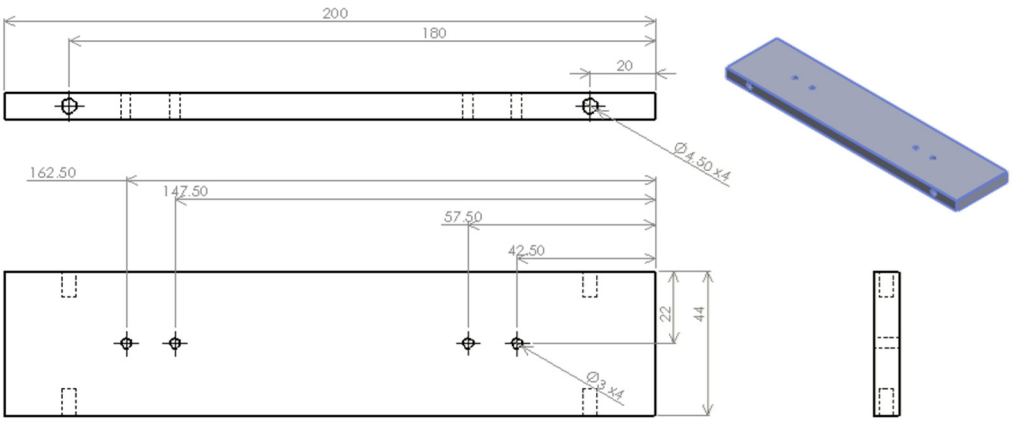

図 2-20　ステレオカメラ底面

2.2.2.4 完成したステレオカメラの評価

図 2-21 が実際に製作したステレオカメラである。棚に角度調整金具を用いて取り付けられている。角度調整金具の詳細については 2.2.3.3 において解説する。

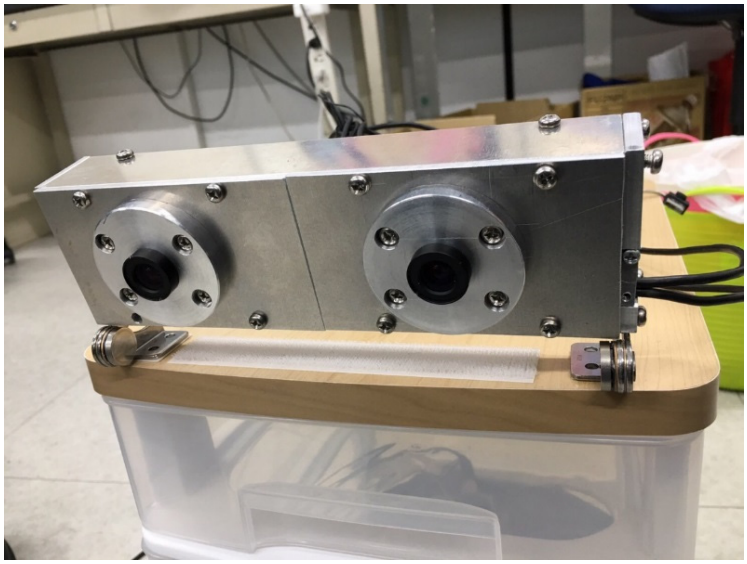

図 2-21　ステレオカメラの全体像

コード固定部以外は予定通り組み立てることが出来た。整備や調整は以前のステレオカメラと比べて行いやすくなった。しかし、製作したステレオカメラで画像を撮った場合、左右の画像に上下方向のずれが生じる。回路やアルミ板金を固定しているネジを締めたり緩めたりすることで、それらをなくすことができるが、時間がたつにつれ、板金の自重により画像がずれてしまう。視差は、左右の画像が 1㎜ 上下にずれていた場合確認できなかった(第 3 章参照)。この誤差は回路を固定しているネジを数十度旋回したり、板金の位置をずらしたりすることで、調整する事が出来る。

ステレオカメラは設計通りに製作する事が出来たことから、製作の精度は自分が工作機械を用いた製作で実現できる誤差である 0.05㎜ 未満であると考えられる。しかし手動の工作機械を用いてステレオカメラを製作する以上、0.05㎜ 以下の誤差を無くすことは難しいため、こうした形での実験の計測には限界があることが分かった。

また、コード固定部が不安定な構造であるため、使用するコードの径に合わせた専用のコード固定を作る必要がある。

2.2.3 自動書類運搬装置の構成要素

自動書類運搬装置の構成要素として「自走ロボット」、「書類収納用の棚」、「ステレオカメラ」があり、これらを用意した上で機体を組み立てる必要がある。その方法について以下で解説する。

2.2.3.1 自走ロボット

自走ロボットとして掃除ロボットのルンバ図 2-22 を使用することにした。白色のプラスチックカバーの部分はルンバの制御用端子を露出するために取り外した。

図 2-22　iRobot ルンバ 622 ホワイト [2-2]

2.2.3.2 収納部の棚

棚は図 2-23 のウッドトップテーブルチェスト ET-W430 を使用する事にした。

図 2-23　ウッドトップテーブルチェスト ET-W430［2-3］

この棚を選んだ理由は、一般的によく使われる使用しやすい形状の棚であり、A4 書類を入れる事が出来るためである。さらに上部が平坦で厚い木なので、安定して上に物を置くことができる。なお、ルンバに棚を搭載する際に、ルンバに接続された回路が棚の下段に干渉してしまうため、下段を取り除きルンバの調整や接続が行いやすくなるようにした(図 2-24)。

図 2-24　棚を設置した自走ロボット

2.2.3.3 ステレオカメラおよびその設置法

第 3 章で示すように、ステレオカメラは床に置かれた目印を認識する必要がある。そのため、レンズは正面ではなく前方の床が写るよう斜めに設置しなければならない。そこでステレオカメラを棚に設置する際に角度調整金具を用いることでレンズの角度調整を行えるようにした（図 2-25)。これに用いたのは図 2-26 のトルクヒンジ HG-TA 型 RL である。

図 2-25　トルクヒンジに取り付けたステレオカメラ

図 2-26　トルクヒンジ HG-TA 型　RL

トルクヒンジと棚は図 2-27 のようにネジでしっかりと固定した。ステレオカメラは取り外すことも多いため、図 2-25 のように強力両面テープを用いてトルクヒンジと固定した。

図 2-27　棚に取り付けられたトルクヒンジ

2.2.4 機体仕様

本章で解説した機体は表 2-1 に示した部品で製作した。

表 2-1　機体部品表

名称	用途
アルミ(A5052)切板 厚み 8mm	・ステレオカメラ側面 ・ステレオカメラ底面
アルミ(A5052)切板 厚み 3mm	・ステレオカメラ側面 その 1 ・ステレオカメラ上面 その 1 ・ステレオカメラ上面 その 2
アルミ(A5052)切板 厚み 5mm	・ステレオカメラ側面 その 2
アルミ丸棒(A2017) 外径 50Φmm	・レンズ固定
CNR ゴムシート	・回路の保護 ・コードの固定
ネジ(M4×10)	・各部の固定
ネジ(M3×10)	・コードの固定
アクリルネジ(M3×10)	・回路の固定
六角ナット　M4	・各部の固定
トルクヒンジ HG-TA 型	・レンズの角度調整
レンズ	・画像の認識
ウッドトップテーブルチェスト ET-W430	・書類の運搬
iRobot ルンバ 622 ホワイト	・機体の移動
マジックテープ	・棚とルンバとの固定
ナイスタック　強力両面テープ	・棚とステレオカメラとの固定

2.3 機体の完成

2.3.1 動作確認

図 2-28 は完成した全体の写真である。表 2-2 に仕様を示す。棚の上部に設置された PC は第 3 章で解説する目印認識と距離計測に用いられる。棚の最下段には第 4 章で解説するルンバ制御用の回路が設置されている。

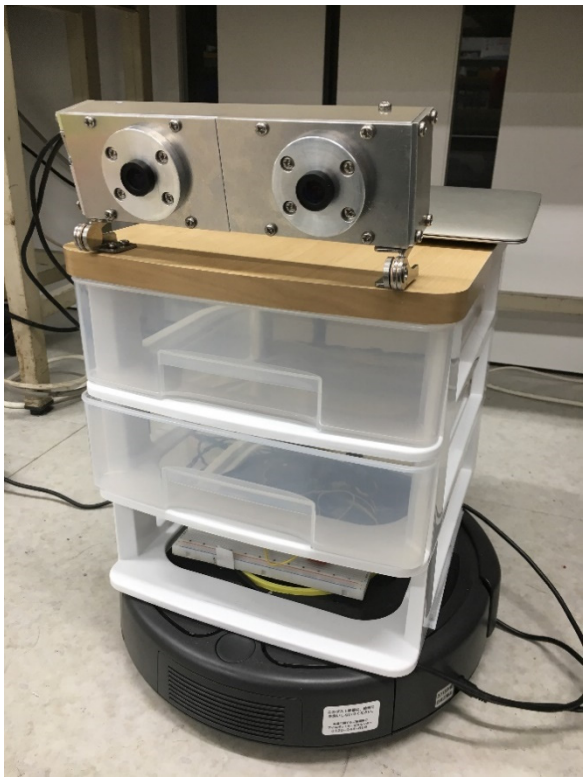

図 2-28　自動書類運搬装置

表 2-2　自動書類運搬装置の仕様

縦	440mm(PC を開いた状態では 550mm)
横	330mm
奥行き	330mm
全重量	6.5kg

すべてのシステムが完成した後に動作テストを行ったが、動作中に機体の構造が崩れることはなく、問題なく指示された行動を行うことが出来た。

2.3.2 考察

予定通り自動書類運搬装置を製作する事ができた。この機体は 2.1.2 項の要求条件、制約条件をすべて満たしている。

第 3 章　目印認識と距離計測（池・中川・堀井担当）

3.1 概要（堀井担当）

3.1.1 目的

自動書類運搬装置を自動で目的地まで移動させるためには、目印の場所を把握する必要がある。そこで、目印を把握するための目印認識のプログラムと、ステレオ法を利用して目印までの距離を計測するプログラムを作成する。本章はそれらのプログラムについて解説する。

3.1.2 要求仕様

目印の認識を実現するための要求仕様を以下に示す。

1. 目的地となる目印の製作。
2. 色認識を用いた目的地の把握。

ステレオ法による距離計測を実現するための要求仕様を以下に示す。

1. ステレオ法を用いるのに適したレンズ間距離の算出。
2. ステレオ法を用いた自動書類運搬装置と目印までの距離の測量。

なお、レンズ間距離とは図 3-1 のように、ステレオカメラの 2 つのレンズ間距離を表す。

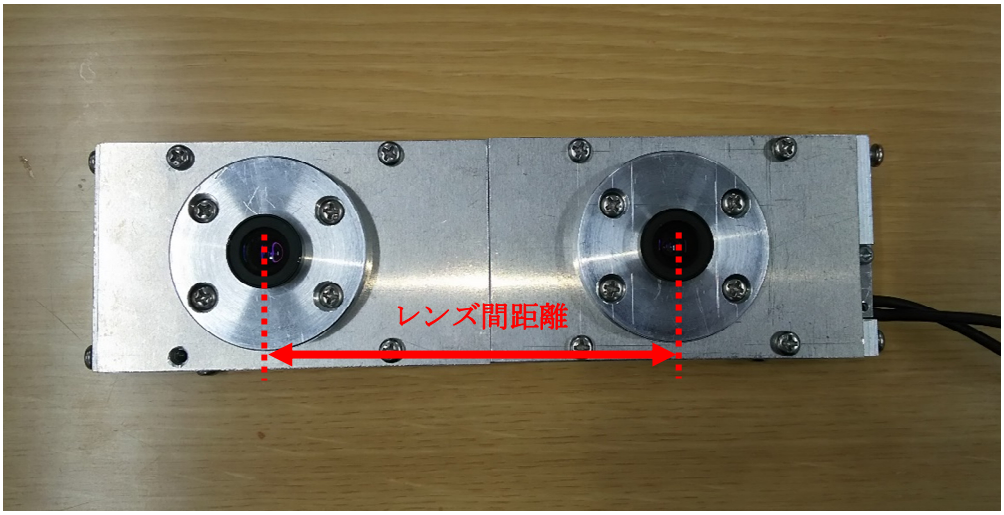

図 3-1　レンズ間距離

3.1.3 使用したソフトウェア

目印認識と距離計測を行うための開発環境を解説する。

3.1.3.1 Visual Studio 2015

プログラム作成の開発ツールとして、Visual Studio 2015 を使用する。Visual Studio 2015 はマイクロソフトのソフトウェア開発製品群および、それらを管理する統合開発環境である。

3.1.3.2 OpenCV 3.0

OpenCV とは、Intel 社が開発した画像処理・画像解析を行うライブラリである。言語は、C/C++、Java、Python が使用可能である。また、Windows、Linux、Mac OS、Android、iOS が公式にサポートされている。

3.1.4 プログラムのフロー

本章で作成するプログラムのフローについて解説する。

起動されたプログラムは、図 3-2 のように大きく分けて 5 つのフローを実行する。

まず、ルンバに 360° 旋回のコマンドを送信することで、旋回を始める。

旋回の最中に、ステレオカメラの片方のカメラから得た映像中から目印を探し、その位置を把握する。目印の位置を把握する為に、まずは特徴物の面積計算を用い対象の絞り込みを行い、その後色認識を用い目印の位置を確認する。

目印が確認されなかった場合は①に分岐して、また 360° 旋回を始める。目印が確認できた場合は②に分岐し、ルンバに旋回を停止するコマンドを送信し、停止する。

旋回停止後は見つけた目印までの距離計測を行う。この時、ステレオカメラから得た 2 枚の画像から視差画像を作成し、その視差画像から距離が算出される。

距離計測の終了後、得られた距離を基にルンバにコマンドを送信し、得られた距離だけルンバは前進する。

前進終了後、他に探す目印がある場合は③に分岐し、また 360° 旋回を始める。全ての目印を探し終えている場合は④に分岐し、プログラムが終了する。

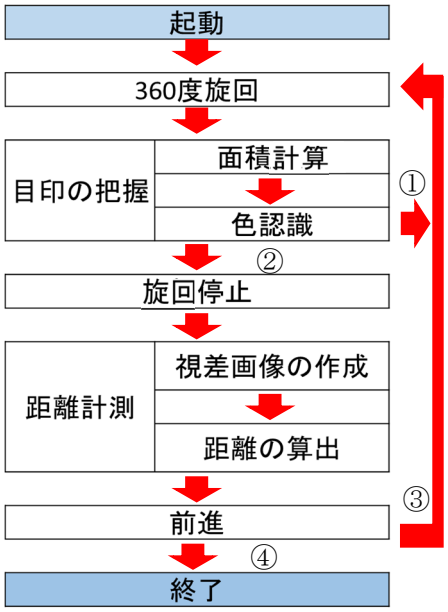

図 3-2　プログラムの流れ

3.2 使用する目印(中川担当)

3.2.1 概要

自動書類運搬装置が動作するためには、目的地となる目印が必要となる。そのために、小型の直方体の目印を作成した。ここではその目印を決定するまでのプロセスについて解説する。

3.2.2 目印の作成

ダンボールに色画用紙(図 3-3)を貼った縦縞状の目印(図 3-4)を作成し、3.5 節で解説する距離計測を行った。それにより、色画用紙の横幅何 cm が使用する目印に適しているかを調べた。

その結果、横幅が広いと中心の視差が確認されないことが分かった。これにより、目印の横幅は 5cm が距離計測を行うのに適していると判断した。

図 3-3　使用した色画用紙［3-1］

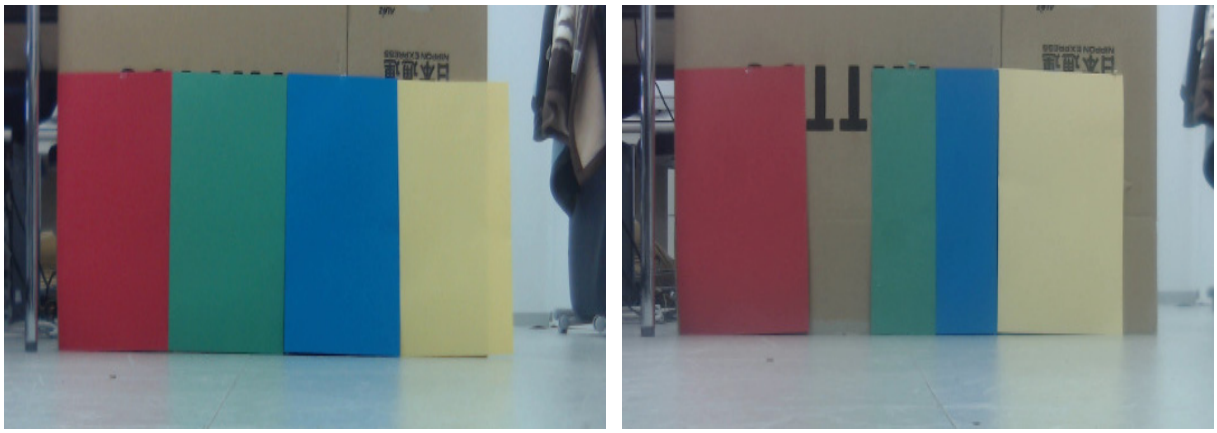

図 3-4　横幅を決定するために作成した目印(縦 210×横 297mm)

3.2.3 実験当初の目印

3.2.2 項の結果を基に、目印の横幅は 5cm が距離計測を行うのに適していると判断したため、図 3-5 のような横幅 5cm の目印を作成した。

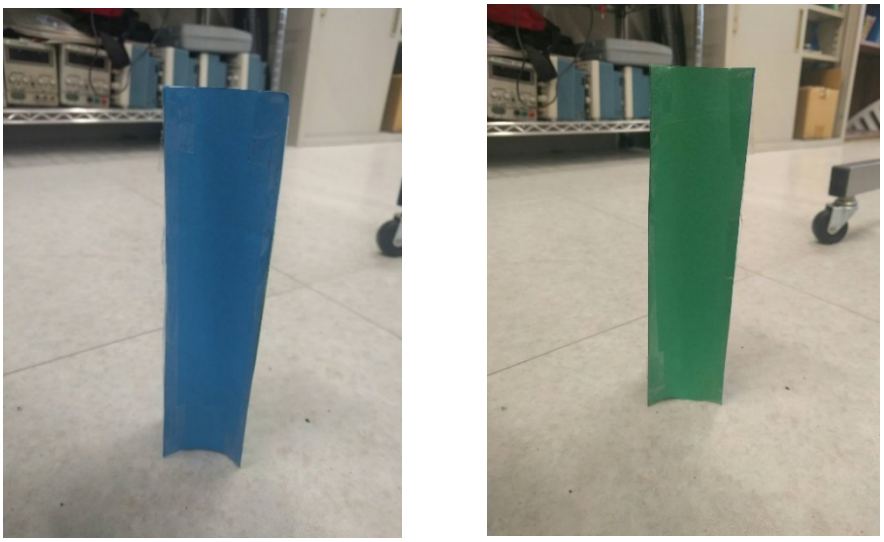

図 3-5　実験当初で作成した目印(縦 50㎜×横 50㎜×高さ 150㎜)

しかし、ルンバが走行する際、図 3-5 の目印では障害になり(図 3-6)、ルンバが迂回しないといけないので新たな目印を作る必要性があった。

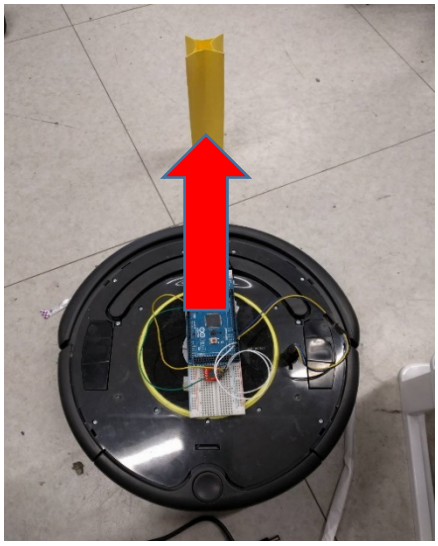

図 3-6　目印が障害になる場面

3.2.4 実験最終段階の目印

ルンバが目的地まで走行する際、目印は障害物にならないのが好ましい。そこでルンバが乗り越えることができる目印を作る必要がある。

我々が使っているルンバは、2cm までの厚みのある障害物を乗り越えることができるのでその範囲で目印を作成することにした。

ルンバが目印を乗り越える際、画用紙では耐久性がないことから、図 3-7 および図 3-8 のプラスチック板とばね座金を用いて作成する。

図 3-7　使用したプラスチック板

図 3-8　使用したばね座金

図 3-9 が作成した実験最終段階の目印である。

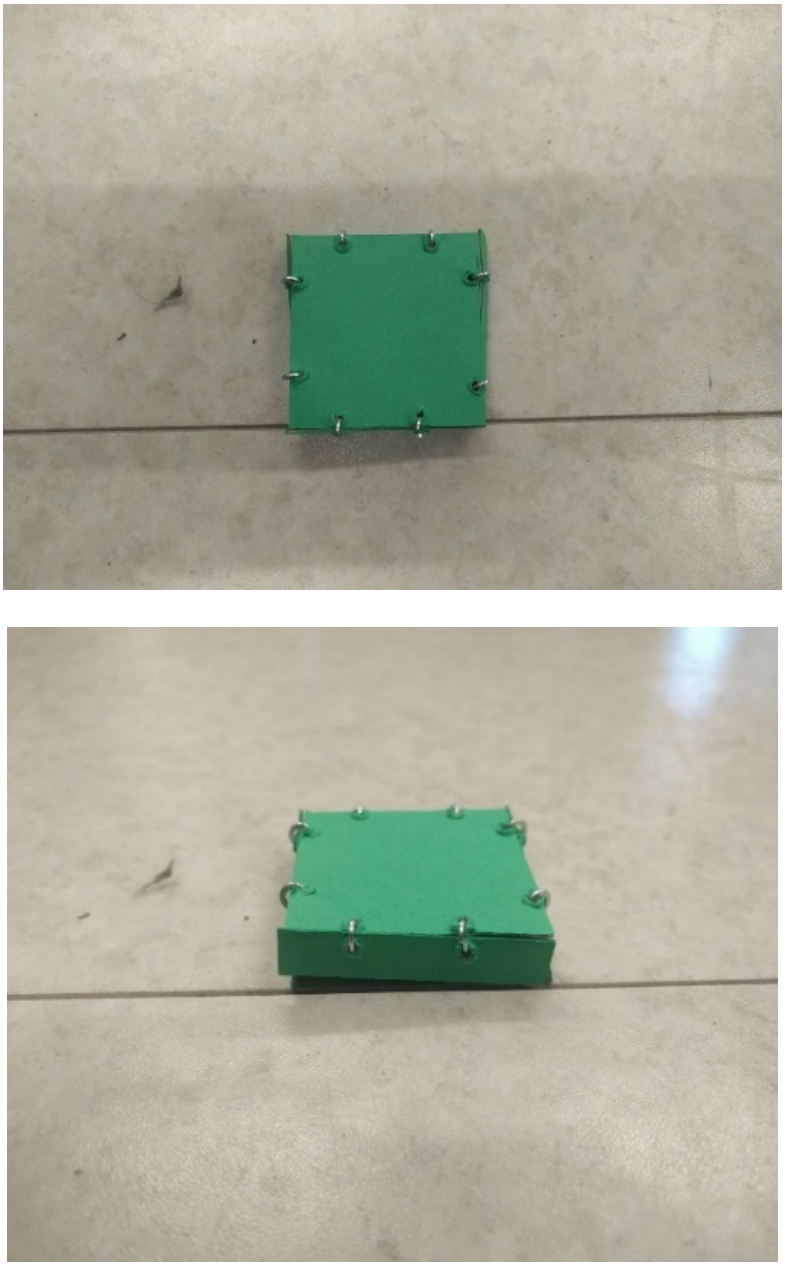

図 3-9　実験最終段階で作成した目印(縦 50㎜×横 50㎜×高さ 10㎜)

3.2.5 実験最終段階の目印でのルンバの改良

図 3-9 の目印を作成しルンバを走行させた結果、ルンバの部品であるブラシなどが走行の障害になることが分かった。

そこで、ルンバのブラシなどの部分を取り外し、プラスチック板で蓋をすることで走行の障害を減らすことにした(図 3-10～図 3-13)。

その結果、ルンバは目印を乗り越えることができるようになり、容易に走行が可能になった。

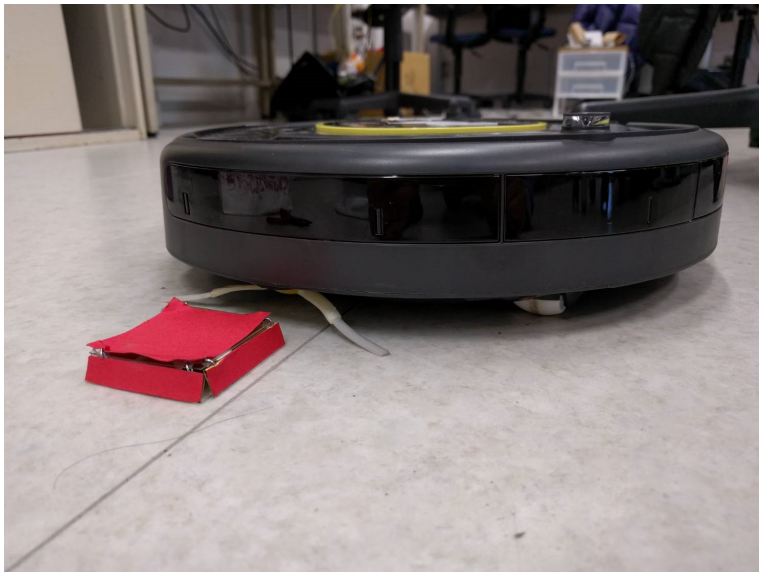

図 3-10　走行中にブラシが障害になる図

図 3-11　改良後のルンバ

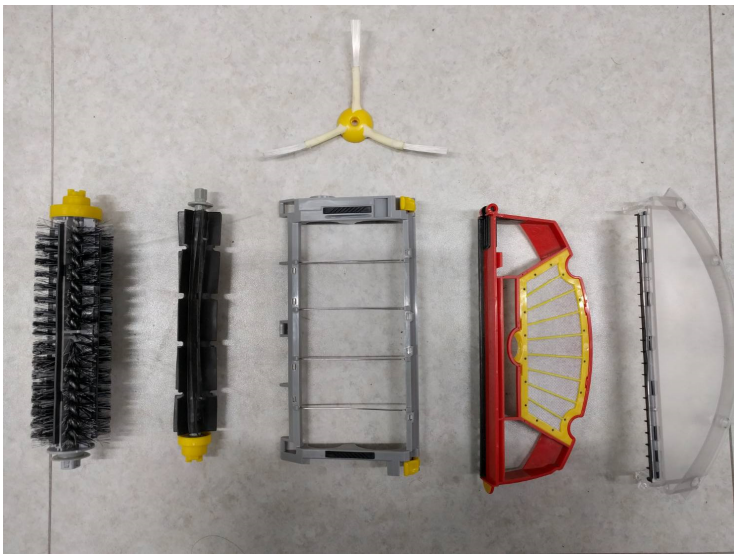

図 3-12　ルンバに取り付けられている部品(ブラシなど)

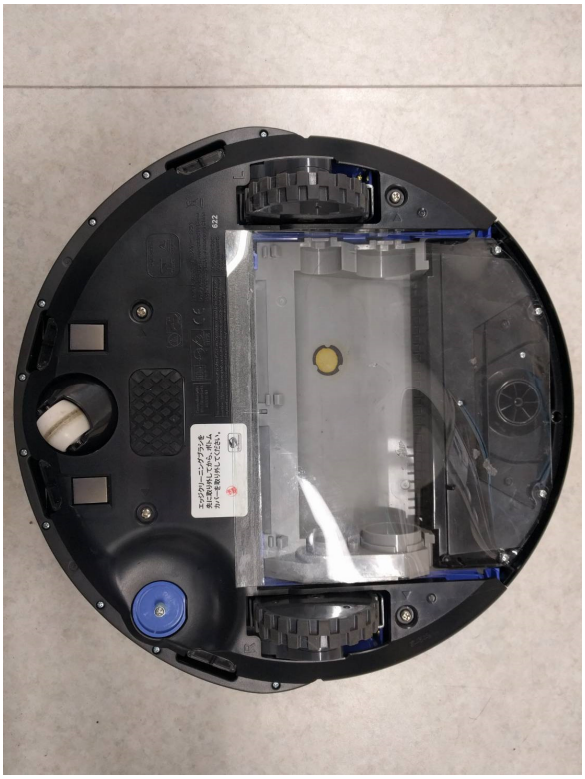

図 3-13　改良したルンバの裏面

3.3 色認識による目印認識（池担当）

3.3.1 概要

本節では図 3-2 のプログラムのフローのうち目印把握のプログラムについて解説する。これは、自動書類運搬装置に取り付けるステレオカメラの一方（右のカメラ）のカメラを使用して目印を探すために用いられる。この際、対象物の絞り込みにより候補を減らしてから目印を見つける。このプログラムに必要な条件は以下の 6 つである

1. カメラから取得した映像を 2 値化できること。
2. 目印の候補を面積でとらえられること。
3. 特定の大きさから外れる面積を持つ候補を除外できること。
4. 目印の重心を割り出せること。
5. 割り出した重心の位置における画素値を求められること。
6. 求めた画素値が、目印の色のどれに近いかを定められること。

3.3.22 値化するプログラムの作成

目印の候補をとらえるにあたり、まず画像の 2 値化を行う。2 値化とは、濃淡のある画像を白と黒の 2 階調に変換する処理のことである。閾値を定めて、各画素の値が閾値を上回っているか下回っているかで白黒に置き換えられる。

画像を 2 値化する際、下記のコードのように読み込んだ画像を cvtColor 関数でグレースケールに変換している。次に threshold 関数を用いて 2 値化する。THRESH_BINARY_INV により、画像中の明るい箇所が黒く、暗い箇所が白く変換される（図 3-14）。また、閾値は THRESH_OTSU を用いることで自動に設定される。目印を白色で表示する理由としては、この後のラベリング処理において目印が白色である必要があるからである。

```
cap >> cap_img;
cvtColor(cap_img, gray_img, CV_BGR2GRAY);
threshold(gray_img, grayinv_img, 0, 255, THRESH_BINARY_INV | THRESH_OTSU);
```

プログラムを実行した結果を図 3-14 に示す。これを見ると明るい床面は黒色に、相対的に暗い目印は白色になることがわかる。

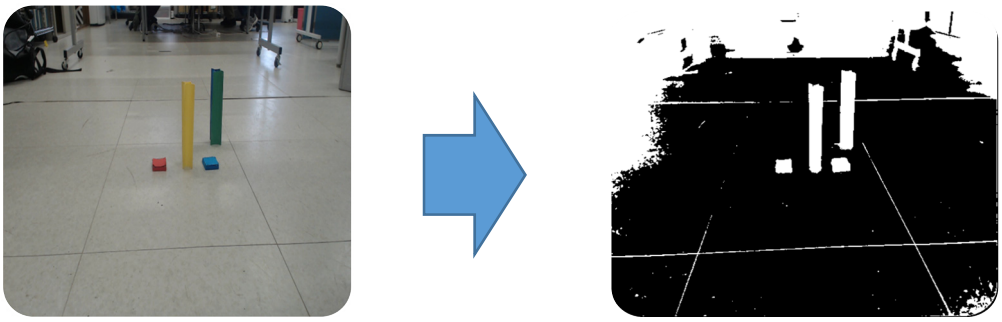

図 3-14　2 値化した画像

3.3.3 面積計算するプログラムの作成

次に 2 値化された画像から白色の部分の面積を計算する。そのためにまず画像のラベリング処理を行う。ラベリング処理とは、2 値化処理された画像において、白色の部分が連続した画素に同じ番号を割り振る処理のことである。通常、同じ番号ごとの面積（画素数）や幅、高さなどの特徴量を求めて欠陥検査や分類処理などに用いられる。このラベリング処理を用いて面積を計算する。ラベリング処理は connectedComponentsWithStats 関数を用いて行う。

```
int nLab = cv::connectedComponentsWithStats(grayinv_img,LabelImg, stats,
centroids);
```

ConnectedComponentsWithStats で得た stats 変数を下記のように param 変数に代入することで、i 番目のラベルがつけられた領域のデータを取り出すことができる

```
int *param = stats.ptr<int>(i);
```

この param 変数から param[cv::ConnectedComponentsTypes::CC_STAT_AREA]という記述により面積を取り出すことができる。

このソースコードを使用した際に得られる面積の模式図を図 3-15 に示す。これの場合、2 値化された青色の目印の面積値は 12 となる。

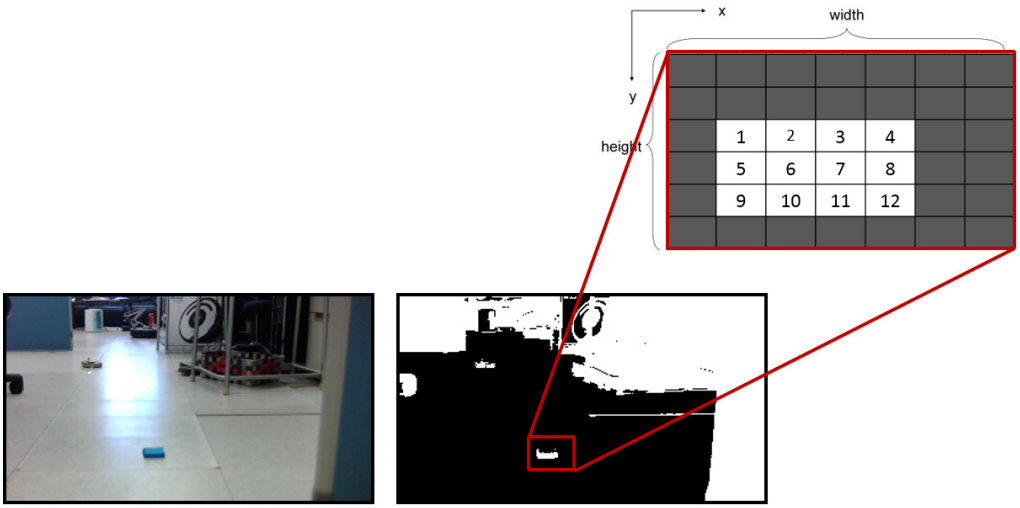

図 3-15　面積の計算の模式図

3.3.4 特定範囲以外の面積を除外するプログラムの作成

3.3.3 項で得た面積を用い、目印の候補の絞り込みを行う。下記のコードの内、○に面積の最大値、△に面積の最小値を記入することで、以後の色認識の処理に進む領域を限定できる。○と△には具体的には 20，50 という値を用いた。

```
if (param[cv::ConnectedComponentsTypes::CC_STAT_AREA] <= ○ &&
param[cv::ConnectedComponentsTypes::CC_STAT_AREA] >= △)
```

3.3.5 目印を色でとらえる

ここでは、複数の目印の候補のそれぞれの色を割り出すプログラムを作成する。3.3.3 項の ConnectedComponentsWithStats 関数により centroids という変数に重心の情報が格納されている。これに対し、下記のコードを実行することで、重心座標（X，Y）を得ることができる。

```
double *param1 = centroids.ptr<double>;
int X = static_cast<int>(param1[0]);
int Y = static_cast<int>(param1[1]);
```

得られた重心座標（X, Y）から、下記のコードで画素値を求める。HSV で画素値を得るために cvtColor 関数で画像を BGR から HSV に変換してから画素値を取得している。

```
int H, S, V;
cvtColor(cap_img, hsv_img, CV_BGR2HSV);
int a = hsv_img.step*Y + (X * 3);
H = hsv_img.data[a];
S = hsv_img.data[a + 1];
V = hsv_img.data[a + 2];
```

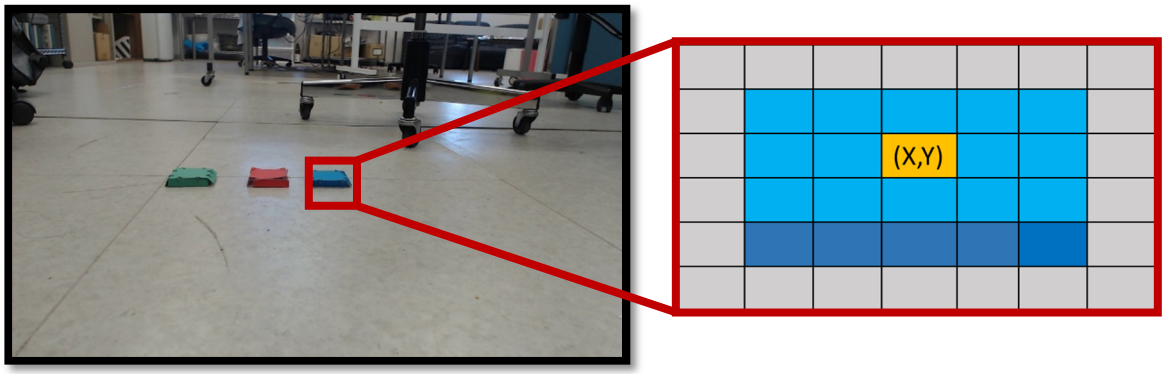

図 3-16　画素値を求める流れ

次に、得られた画素値が赤色、緑色、青色、紫色の目印のどの色に近いかを調べる。下記に示したソースコードは、4 種類の目印の画素値を格納している。dataHmax らの {} 内には得られた HSV の最大値が、左から赤色、緑色、青色、紫色の順番で格納されている。dataHmin {} 内には得られた HSV の最小値が、左から赤色、緑色、青色、紫色の順番で格納されている。

```
int dataHmax[] = { 140, 80, 150, 70 };
int dataSmax[] = { 255, 255, 255, 255 };
int dataVmax[] = { 255, 255, 255, 255 };
int dataHmin[] = { 90, 70, 110, 0 };
int dataSmin[] = { 60, 0, 5, 30 };
int dataVmin[] = { 60, 40, 40, 30 };
```

下記の条件より、得られた目印の画素値が、赤色、緑色、青色、紫色のどれであるか判定している。

```
if( H >= Hmin.front() && H <= Hmax.front() && S >= Smin.front() && S <= Smax.front()
&& V >= Vmin.front() && V <= Vmax.front());
```

得られた目印の色が、指定された色であるときのみ、次のステップである距離測定に進む。

3.4 目印を捉える実験

3.4.1 概要

本節では 3.3 節のソースコードを用いたプログラムを実行し、目印を捉えられるかを検証する。このプログラムが成功した場合の模式図を図 3-17 に示す。まず、図 3-2 にあるように自動書類運搬装置が 360 度旋回をし、目印を探す。2 値化、面積計算、候補の絞り込み、色認識と進み、目印を見つける。目印が見つかった場合、図 3-2 のプログラムの流れの図にある旋回の中止に移る。

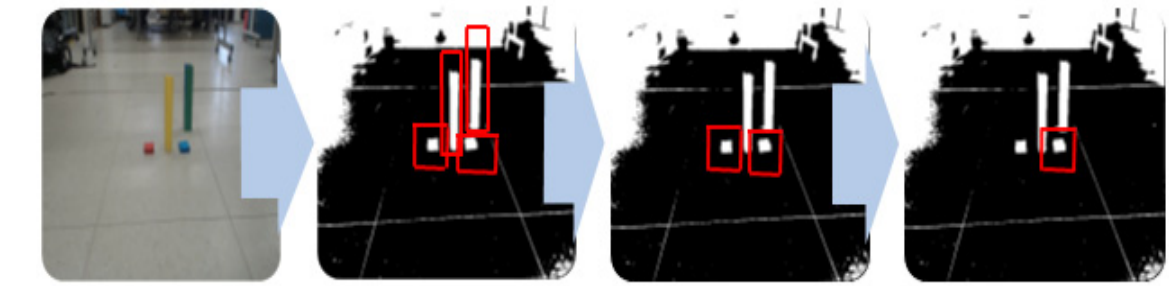

図 3-17　プログラム成功の模式図

3.4.2 実験結果と考察

実験の様子を図 3-18 に示す。目印を面積で絞り込み、色を認識することが出来ていることがわかる。この事から、カメラから得た画像を基に、目印のみを検知し場所を把握できることが分かった。この実験により 3.3.1 項の概要で示した求められる 6 点を満たすことができた。

処理の流れに沿って考察する。まず、2 値化については、処理に時間がかからずにリアルタイムで処理をすることができた。ここで問題として考えられことは、照明などの環境の違いで 2 値化の結果が変わることである。2 値化の閾値が自動で設定されるプログラムになっているため、場所により 2 値化の結果が変化することがあり、その結果で面積計算での目印の絞り込みがうまくできないことがある。解決法は、閾値決定のアルゴリズムを、自動書類運搬装置を用いるのに適した別の方法にすることが考えられる。次に、面積計算による絞り込みの問題点は、目印が近すぎたり離れすぎたりした場合、指定した面積値の範囲に収まらないことである。この問題の解決法は、指定する面積値の範囲を広げることである。広すぎても他の物体を認識してしまう恐れがあるが、前後 50 ㎝目印が移動している場合にもとらえられる範囲はすでに指定できているので問題ないと考えられる。

次に色認識についても、光の当たり方によって測定の値が変わるという問題がある。定めた目印の値は事前に調べた値であるため、手動で値を変えることができ、ある程度の暗さ、明るさには対応できると考えられる。

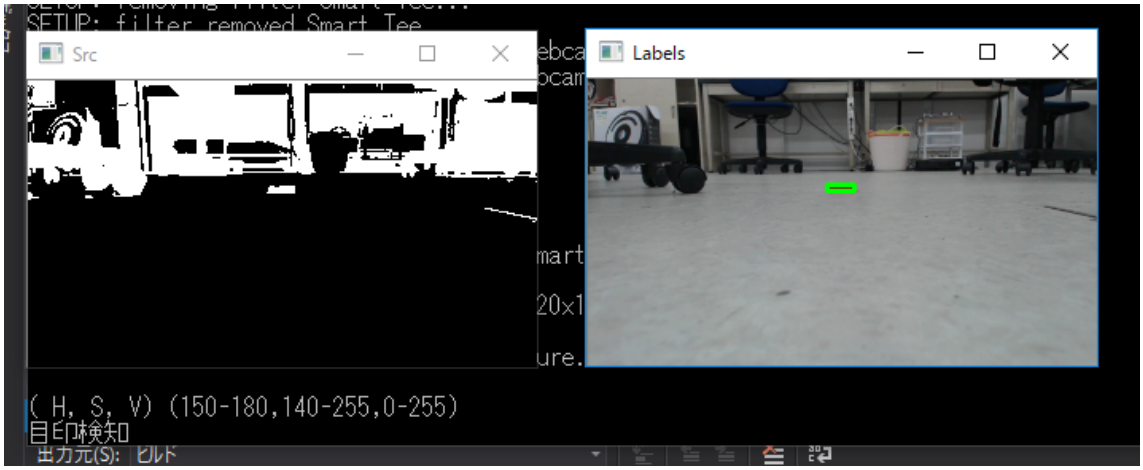

図 3-18　実験の様子

3.5 距離計測の方法（堀井担当）

3.5.1 概要

3.4 節で見つけた目印までの距離を計測する。そのために、ステレオ法による視差抽出が必要となる。本節ではその方法について述べる。

3.5.2 原理・理論

視差を抽出するためには、ステレオ法の原理が用いられる。

ステレオ法とは、三角形の相似の関係を使用して距離を測量する手法のことである(図 3-19)。2 つのレンズと被写体との距離 z は、あらかじめ決められたレンズの焦点距離 f と 2 つのレンズの間の距離 d に加え、被写体を 2 つのカメラで撮影した画像から被写体の像のずれ$|x_1 - x_2|$を用い計算することにより算出される。実際には、ステレオカメラで撮影された 2 枚の画像から、視差画像として対象までの距離が求められる。

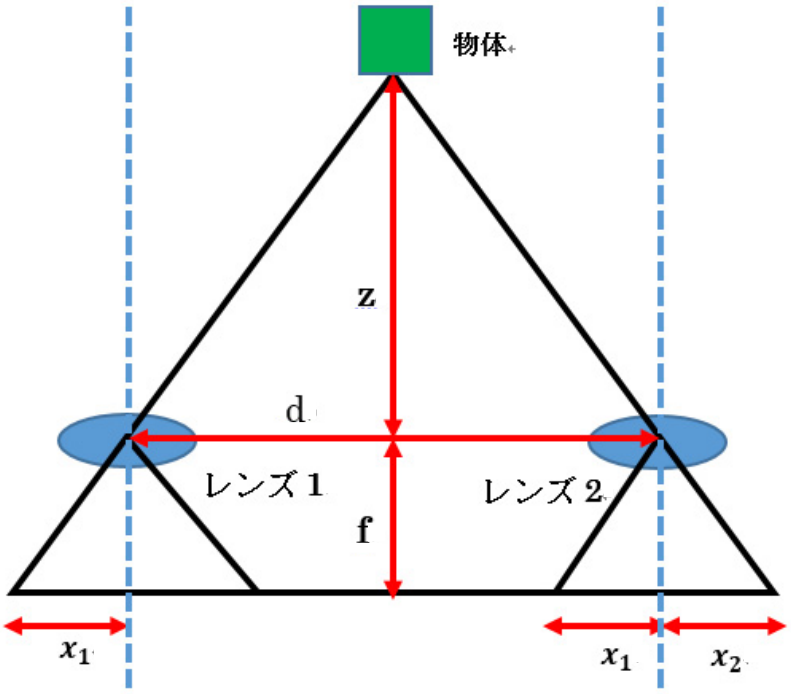

図 3-19　ステレオ法

図 3-20 が実際に得られた視差画像である。

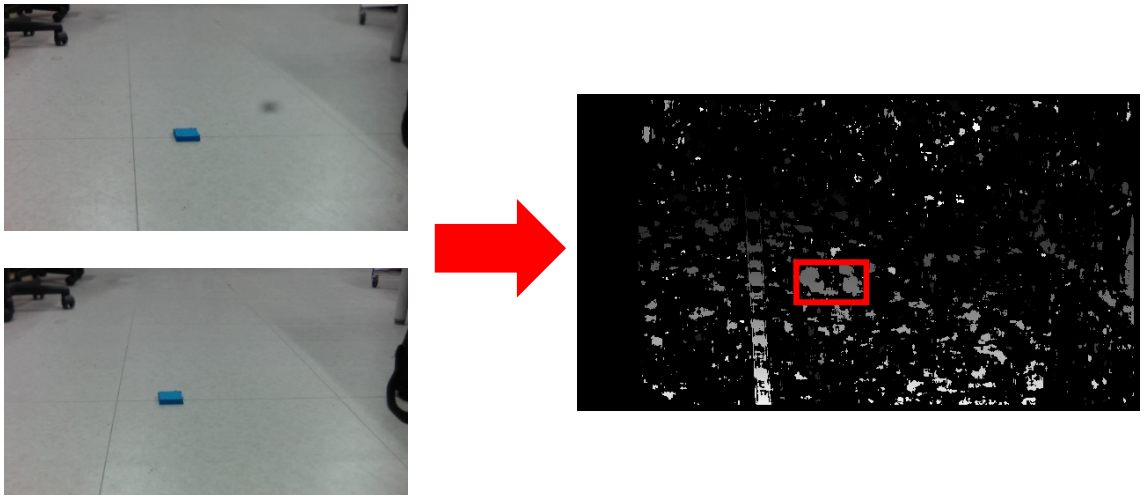

図 3-20　視差画像の作成

作成された視差画像の各ピクセルには視差が格納されている(図 3-21)。視差はその場所の奥行き、つまり距離に対応する。視差は近い距離ほど大きい値をとり、遠いほど小さい値をとる。

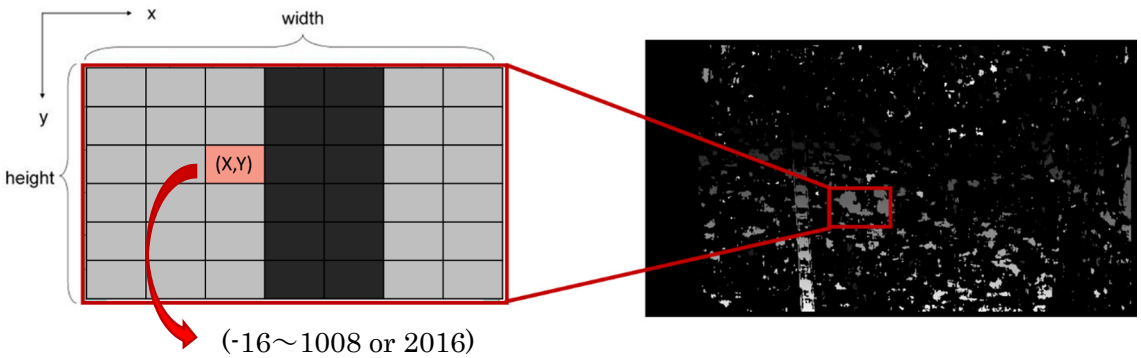

図 3-21　視差画像のデータ構造

3.5.3 視差画像の作成

3.5.2項で解説した原理・理論に基づき、視差を求めるプログラムを作成した。以下でその流れを解説する。

まず、視差画像を作成する場合は、カメラからキャプチャされた画像を、視差画像を作成することができる画像の形式にする必要がある。OpenCVで画像の管理をしているのがIplImage構造体である。この構造体にcvLoadImage関数を用い、一旦保存された左右のカメラの画像を読み込ませる。そのためここで作成される視差画像はカメラからリアルタイムに得た画像ではない。次に、作成される視差画像を準備するために、cvCreateImage関数を用いる。画像の参照先は、読み込みたい画像が保存されているアドレスである。また、IPL_DEPTH_16Sは1画素あたりのビット数のことである。16ビットに指定した理由は次のcvFindStereoCorrespondenceBM関数で入力する画像が16ビットでなければいけないためである。

```
IplImage* leftImage = cvLoadImage("画像の参照先", 0);
IplImage* rightImage = cvLoadImage("画像の参照先", 0);
IplImage* depthImage = cvCreateImage(cvGetSize(rightImage), IPL_DEPTH_16S, 1);
IplImage* dstImage = cvCreateImage(cvGetSize(rightImage), IPL_DEPTH_8U, 1);
```

2枚の画像の準備が終了したら、視差画像の作成を行う。視差画像の作成には、上記で準備した2枚の画像とcvFindStereoCorrespondenceBM関数を用いる。括弧内のleftImage、rightImage はそれぞれ左右の画像である。depthImageは出力される視差画像である。この処理を経て視差画像が作成される。

```
CvStereoBMState* state = cvCreateStereoBMState(CV_STEREO_BM_BASIC, 128);
cvFindStereoCorrespondenceBM(leftImage, rightImage, depthimage, state);
```

作成された視差画像は肉眼で視差が確認し難いため、確認をしやすくするために視差画像の正規化を行う。下記の cvNormalize 関数に作成した視差画像を入力し、dstImage という正規化された視差画像を作成する。正規化の詳しい内容は 3.5.3.1 に述べる。

```
cvNormalize(depthImage, dstImage, 0, 256, CV_MINMAX);
```

3.5.3.1 視差画像の正規化

プログラムによって作成された視差画像は図 3-22 の右上のように全面が灰色で視差が確認し難い。そのために正規化を行い、図 3-22 の右下のように確認しやすい画像に変換する。

全面が灰色になる理由は、視差画像の各ピクセルに RGB の色情報でなく-16～+1008 か-16～+2016 で表される視差が格納されているからである。

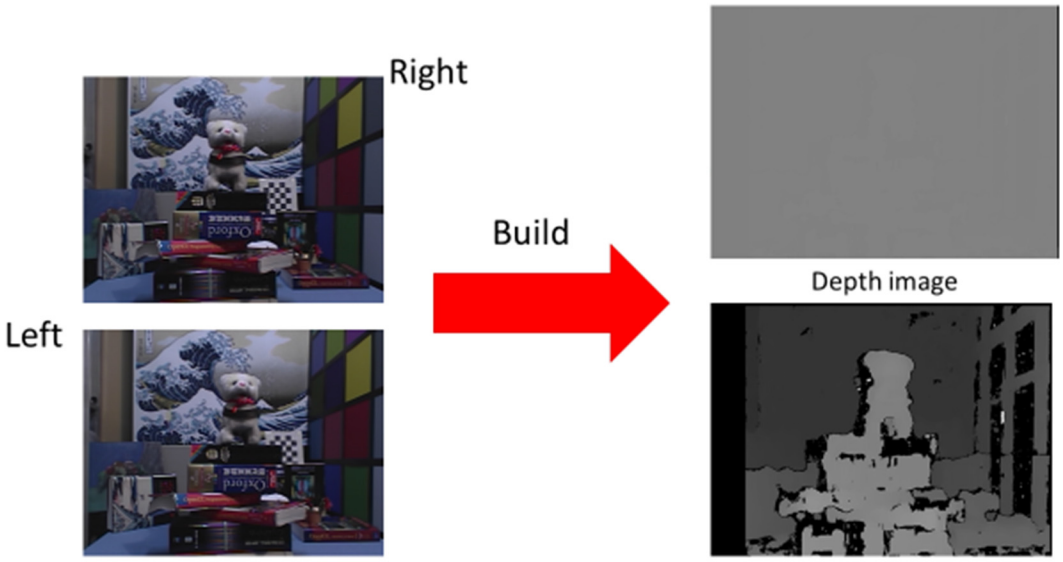

図 3-22　視差画像の作成［3-2］

視差画像が表示される際は、この視差が RGB に変換されたものが表示される。作成される視差画像は 16bit の符号付きである。これは 2^{16}=65536 階調に相当し最小値は-32768、最大値は+32767 となる。実際表示される画像はグレースケールの色の要素で表されるため、最小値は-32768 から 0、最大値は+32767 から 255 に変換される。作成された視差画像の視差の範囲は-16～+1008 なのでこれも同様に変換すると、表示できる色は 127 から 131 の 5 段階となる。このため、図 3-22 の右上のように表示される画像は灰色になる。

これを正規化すると余分な部分がカットされて+1008 が最大値に、-16 が最小値になる。よって、最小値は-16 から 0、最大値は+1008 から 255 となり、図 3-22 の右下の画像のように確認しやすい画像となる。

しかし、正規化をすると余分な部分はカットされるため、表示される距離と実際の距離が異なってしまう。そのため、正規化はあくまで肉眼でどのように視差が計算されているかを確認するものであり、距離を求める際には正規化前の値を使用する。

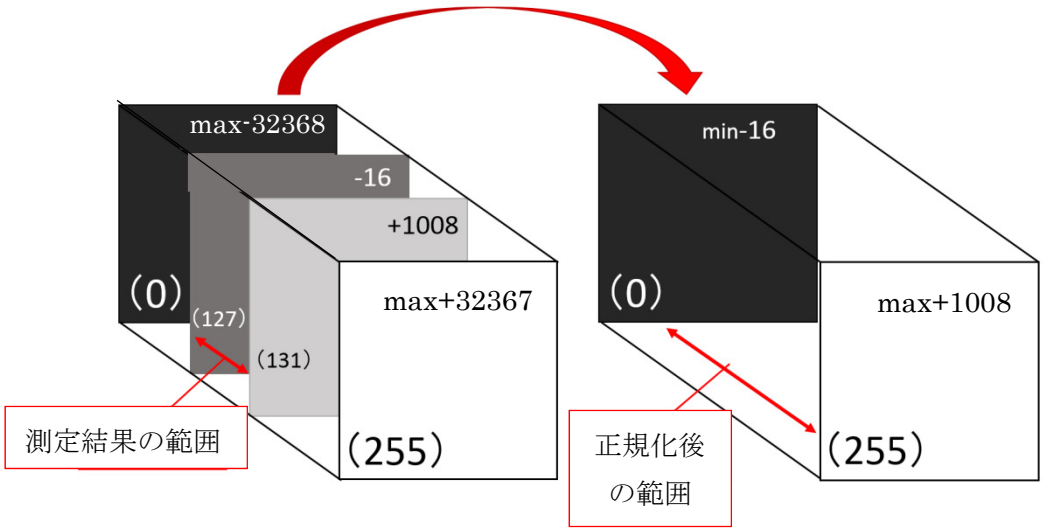

図 3-23　正規化のイメージ

3.5.4 視差の読み取り

作成された視差画像から、視差を読み取る。目印認識で得た座標における、視差を読み取る。

```
CvScalar data = cvGet2D(depthImage, x, y);
std::cout << data.val[0] << "¥n";
```

3.5.5 動作確認テスト

以上で解説したプログラムを用い、以下に記した測定手順で視差画像が作成されるかを確認する。また、このテストではレンズ間距離を変更することが可能なステレオカメラを用いて視差画像の撮影を行う(図 2-5)。

測定の手順は以下のとおりである。

1. ステレオカメラの左右のカメラに上下のずれがないかを確認し設置する。
2. ステレオカメラの軸と被写体の軸が合うように設置する。このとき被写体とステレオカメラの距離が 100cm となるように設置する。
3. ステレオカメラのレンズ間距離が 12cm の位置で被写体を撮影する。撮影が終了後、レンズ間距離 4cm の位置で被写体を撮影する。
4. 撮影された画像を 640×360 のサイズに縮小する。
5. 処理を行った画像を作成したプログラムに入力し視差画像を製作する。

図 3-24　視差を計測する被写体

3.5.6 動作確認の結果

得られた視差画像が、図 3-25 と図 3-26 である。

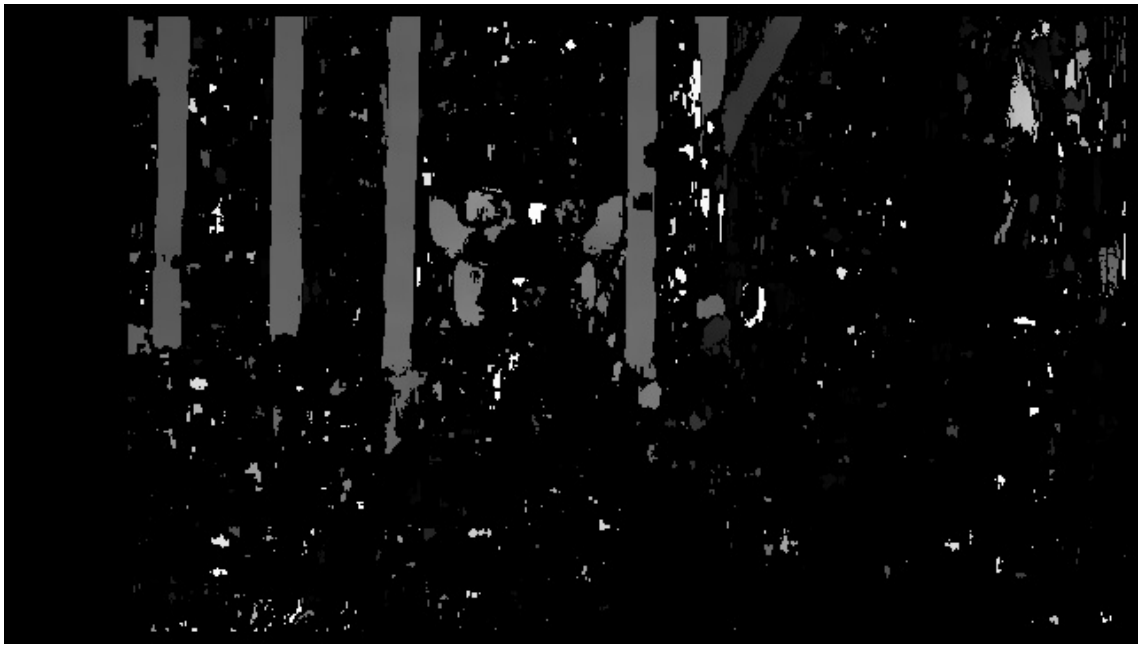

図 3-25　レンズ間距離 12cm の視差画像

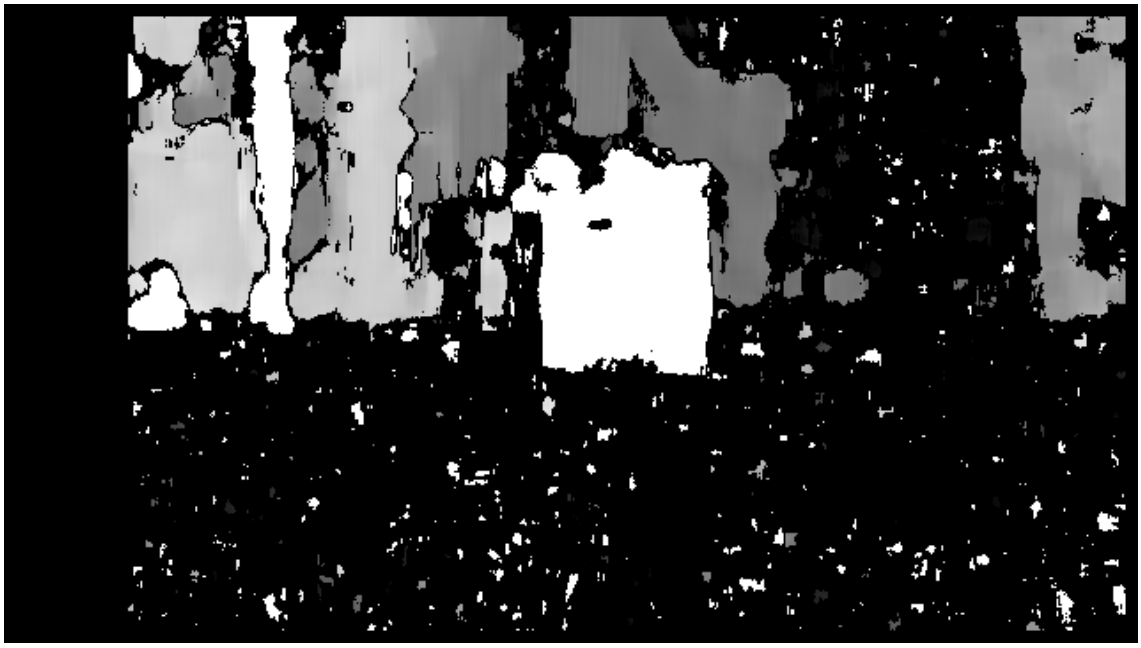

図 3-26　レンズ間距離 4cm の視差画像

3.5.7 考察

結果で得た視差画像からレンズ間距離によって映り方が変化することが分かった。レンズ間距離 4cm のときは被写体の全体が白く視差が出ているのに対して、レンズ間距離 12cm のときは被写体の一部しか視差が出ていないことが分かる(図 3-25、図 3-26)。

視差画像ではレンズ間距離が大きくなるほど被写体と背景の境界、つまり色の変化の境界のみ検出される傾向がある。これは、2 つの画像を比較した際、被写体の中央部が重なり色の変化がなく検出されにくいためではないかと考えられる。

3.6 レンズ間距離と測定可能範囲の関係（堀井担当）

3.6.1 概要

本節では、レンズ間距離による視差の変化について考察する。それにより、最適なレンズ間距離を決定する。実験は仕様決定前に仮作成した目印と第 2 章で紹介したレンズ間距離を変更できるステレオカメラを使用して行った。

3.6.2 測定手順

以下に示した手順で計測を行った。

1. ステレオカメラのレンズ間距離を 8.5cm の位置で固定する。このとき、ステレオカメラの左右のカメラに上下のずれがないかを確認し設置する。
2. 図 3-27 のようにステレオカメラの軸と被写体の軸が合うように設置する。このとき被写体とステレオカメラの距離は 50cm から 450cm まで 10cm 間隔の位置に設置し左右計 82 回撮影を行う。
3. 撮影終了後、レンズ間距離を 5mm 広くし、レンズ間距離 12cm まで 1 と 2 の作業を繰り返す。撮影される画像は 82×8 の 656 枚である。
4. 撮影された画像を 640×360 のサイズに縮小する。
5. 処理を行った画像を作成したプログラムに入力し、視差画像を製作し被写体の視差の測定結果を確認する。

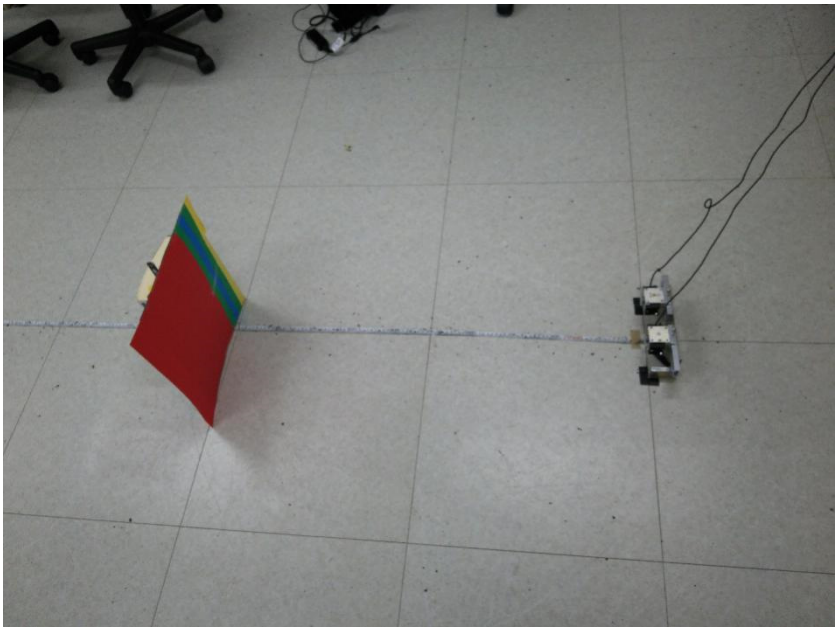

図 3-27　撮影風景

3.6.3 結果

8 種類のレンズに対し、目印までの距離と視差の関係を図示したのが図 3-28 から図 3-35 である。

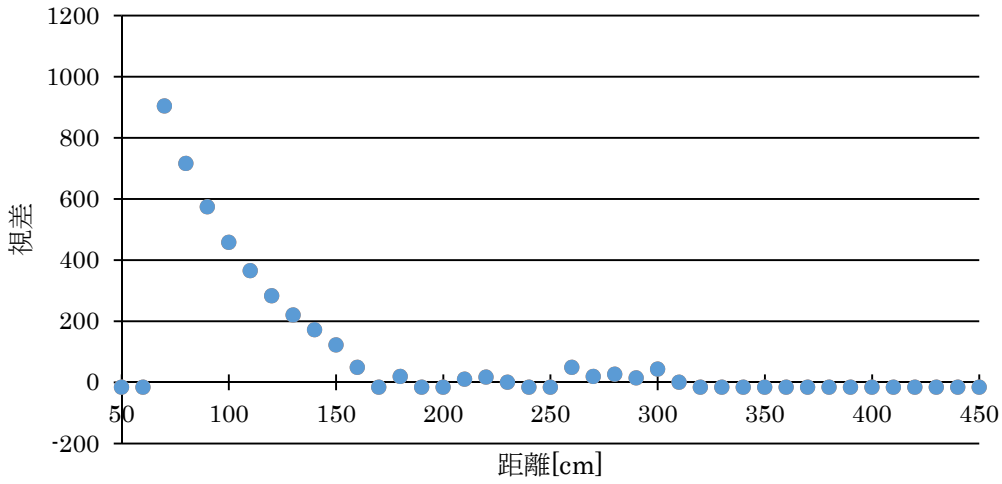

図 3-28　レンズ間距離 8.5cm

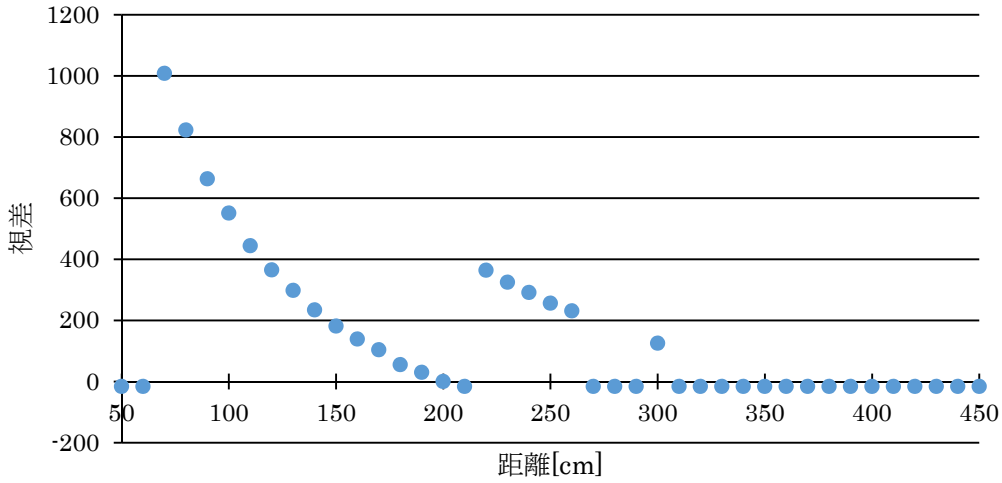

図 3-29　レンズ間距離 9.0cm

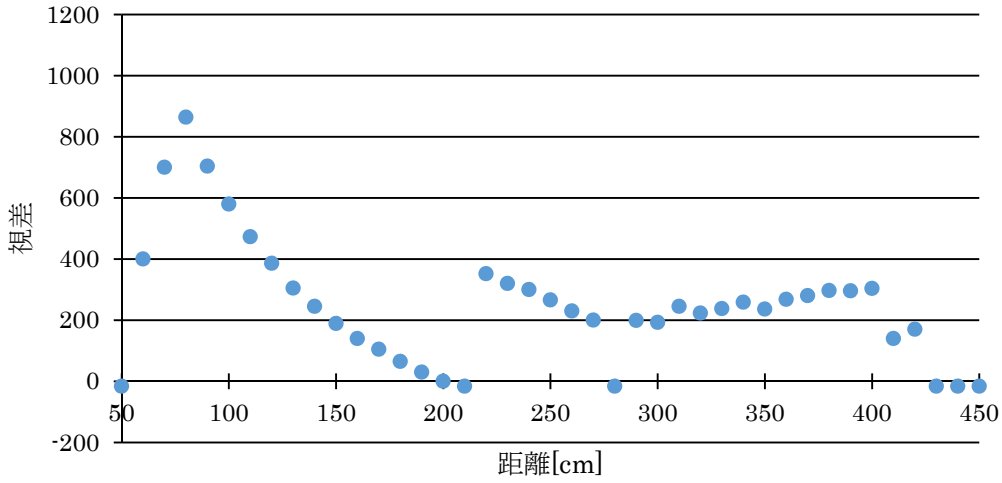

図 3-30　レンズ間距離 9.5cm

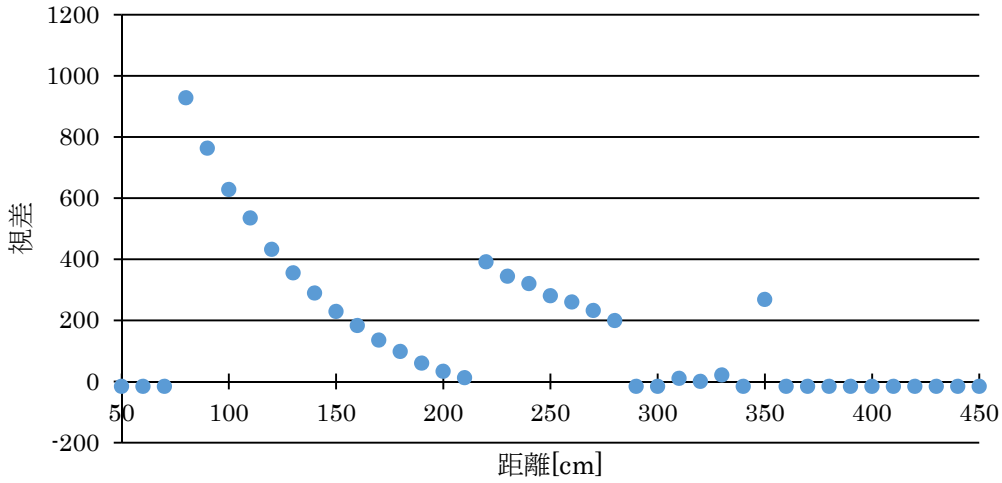

図 3-31　レンズ間距離 10.0cm

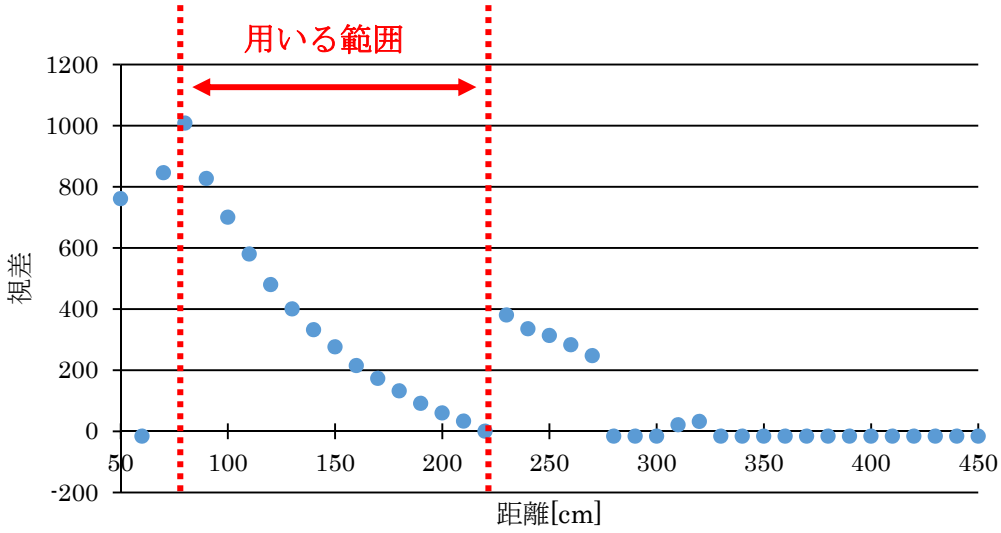

図 3-32　レンズ間距離 10.5cm

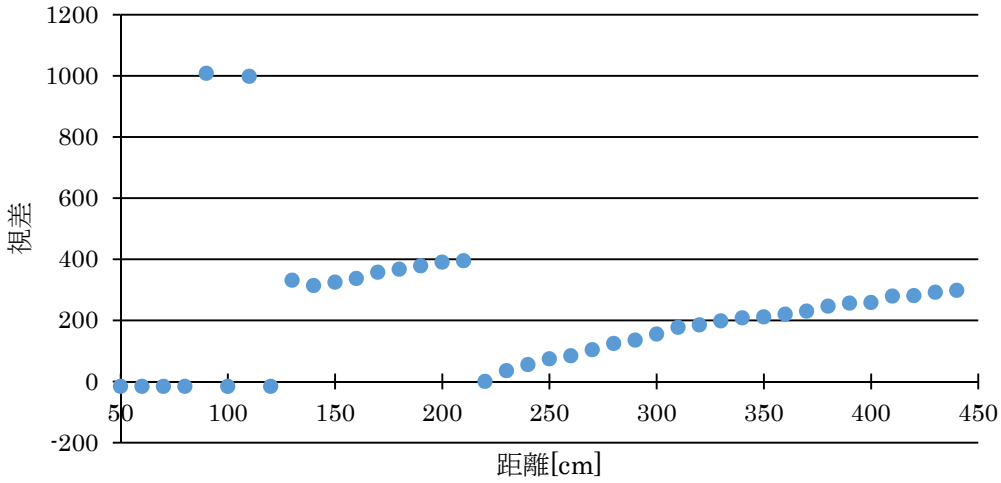

図 3-33　レンズ間距離 11.0cm

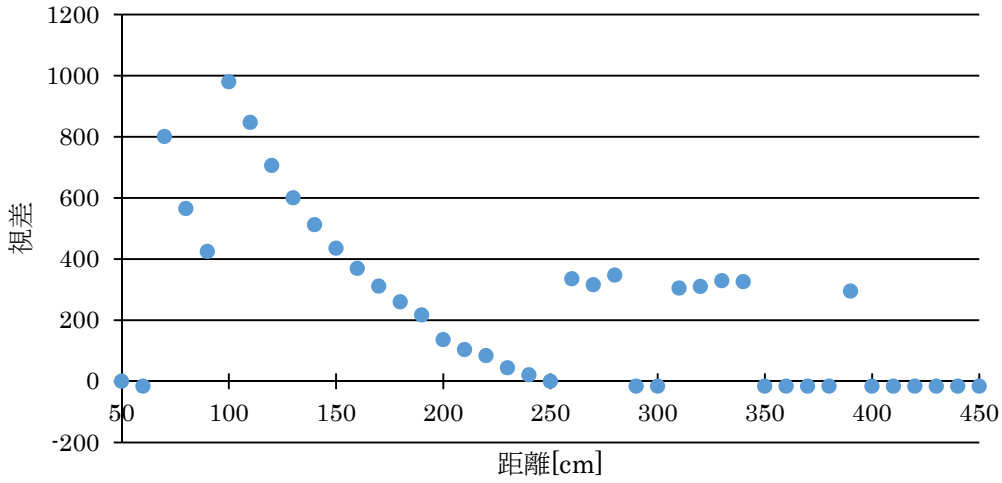

図 3-34　レンズ間距 11.5cm

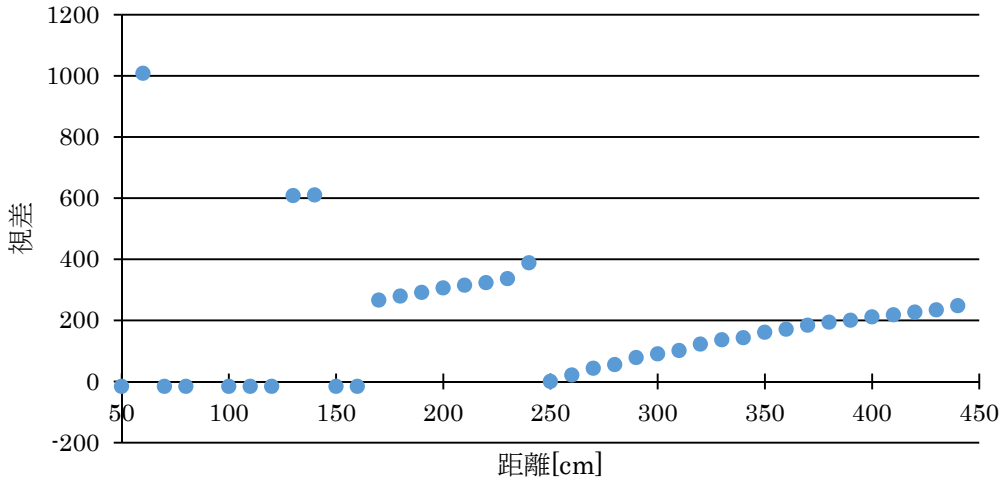

図 3-35　レンズ間距離 12.0cm

3.6.4 考察

レンズ間距離 11cm と 12cm を除いた場合、視差が計測される範囲は、おおよそ 70cm から 240cm の間であることが分かった。またその範囲の値も、近いほど値が大きく、遠いほど値が小さくなっているので、視差画像の特徴に沿っている。

レンズ間距離を決定する上で、重要なことは以下の 2 点である。

1. 広い範囲の距離で 0 より大きい視差を持つこと。
2. 距離と視差の値が 1 対 1 の対応をすること。

1 については、レンズ間距離が大きいほど範囲が広がることがわかる(図 3-32)。しかし、レンズ間距離が大きすぎると安定した視差が得られない(図 3-33)。一方 2 については、レンズ間距離が小さい方が 1 対 1 の対応が得られやすいことが分かる(図 3-28)。

本研究では、1 の距離範囲を重視し、図 3-32 のレンズ間距離 10.5cm を採用する。図 3-32 では視差 900 や 300 に対して複数の距離の候補があり得る。しかし、本研究の自動書類運搬装置はオフィスという限定された場所で用いるものであるので、視差から距離の対応として、90cm～220cm のみに限定して用いることで大きな問題はないと考えらえる。

3.7 距離の計測（堀井担当）

3.7.1 概要

3.6 節の測定結果及び考察をもとに、レンズ間距離を 10.5cm に固定したステレオカメラを製作した(2 章参照)。本節より、このステレオカメラを用いて実験を行う。そして、そのカメラを用いて目印の視差を読み取り、その値をもとに視差と距離の関係式を決定した。

3.7.2 精度の目標

ここでは距離計測の精度の目標について述べる。

目印から半径 35cm 以内、つまり目印までの距離を測定した際に、実際の距離と比べ±35cm の誤差で距離が計測できることを目標とする。この理由はルンバの直径は 34cm であり、図 3-36 のように誤差の範囲が半径 35cm 以内であれば目印から離れすぎない位置で停止できると考えたからである。

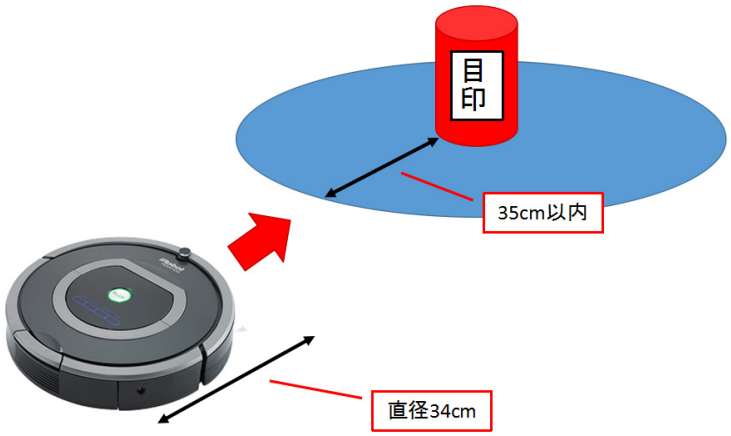

図 3-36　測定精度の設定

3.7.3 関数の作成方法

以下の測定手順で得られた結果をもとに、各距離とその時の視差のグラフを作成する。作成したグラフの結果から近似曲線を作成し、その近似曲線の式をもとに距離を算出する。

今回の実験では、ステレオカメラは自動書類運搬装置として運用することを考え実際に装置上に固定した(図 2-28)。

以下に示した手順で計測を行った。

1. 高さ35cmの位置にレンズ間距離が10.5cmのステレオカメラを下向き1度で固定する。このとき、ステレオカメラの左右のカメラに上下のずれがないかを確認し設置する。
2. ステレオカメラの水平方向の向きを被写体に合わせるように設置する。
3. 被写体とステレオカメラの距離は 50cm から 400cm まで 5cm 間隔の位置に設置し、左右計 80 回撮影を行う。
4. 撮影された画像を 640×360 のサイズに縮小する。
5. 処理を行った画像を作成したプログラムに入力し、作成された視差画像から目印の視差の測定結果を確認する。

3.7.4 結果

視差と距離の関係が図 3-37 である。これより、90cm～400cm の範囲で視差を確認することができた。このグラフから、視差 x から距離 y を算出する関数を以下のように決定した。

・使用した関数

$$y = 0.000000000000000299221x^6 - 0.000000000002738034714x^5 + 0.000000103570021499 75x^4 - 0.000207501131764597x^3 + 0.232687536997478x^2 - 139.047178021229x + 34993.0746106183$$

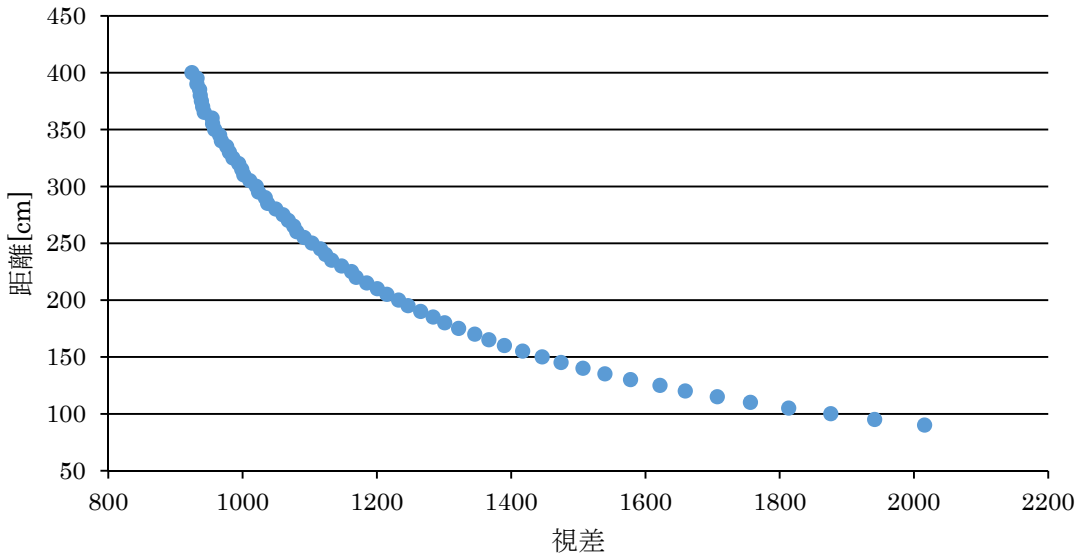

図 3-37　距離と視差の関係

次に、この関数から算出した距離が実際の距離と、どの程度異なるのかを調べた。それが図 3-38 である。平均の誤差は 1.65cm となった。また、最大の誤差は距離 395cm の位置で誤差 8.32cm。最小の誤差は距離 230cm 位置で誤差 0.021cm となった。目印までの距離が離れるにつれ誤差が大きくなる結果となった。

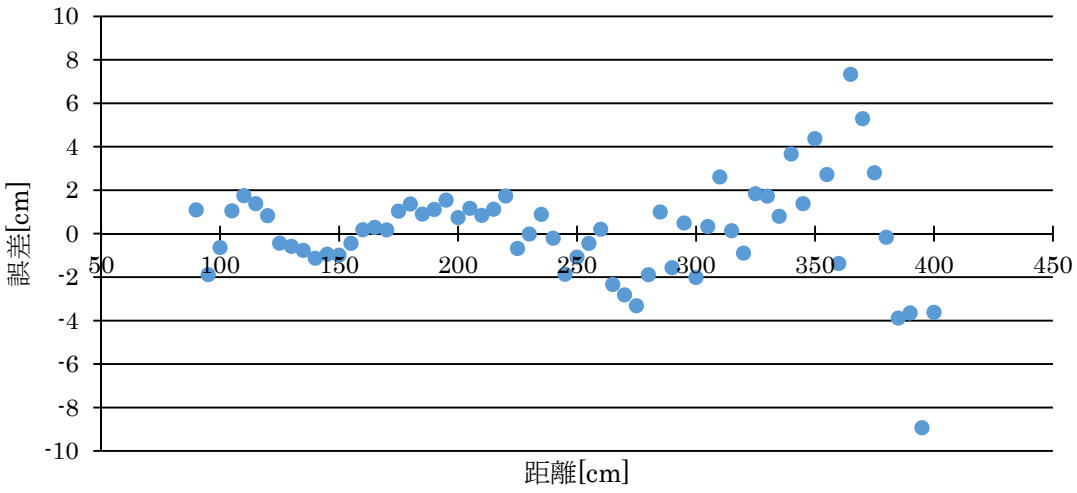

図 3-38　距離と誤差の関係

3.7.5 考察

新たに作成したステレオカメラでは 90cm～400cm の範囲で視差を撮影することができた。視差が撮影されている範囲は、3.6 節の結果より広くなっている。これは、実験に用いたステレオカメラが変わったためと考えられる。次に、距離計測の平均誤差は 1.65cm となった。目標の誤差は 35cm 以内であり、この目標を満たしていることから、自動書類運搬装置に用いる距離の算出方法として適していると言える。また、誤差の分布は 200cm 付近の誤差が小さく、距離が遠くになるにつれ大きくなる結果となった、このことから、誤差が小さかった 200cm の間隔で目印を設置することに決定した。

しかし、カメラのピントや上下のずれなどにより、視差が変化する場合があった。このことから、視差を計測する前にカメラのピントと上下のずれの補正や調整を行うプログラムを作成したり、より精度の高いステレオカメラを製作することが望ましいことが分かった。

3.8 目印認識と距離計測の最終評価（池・堀井担当）

3.8.1 要求仕様との比較

目印の認識を実現するための要求仕様を満たす必要があった。

1. 目的地となる目印の製作できること。
2. 対象を絞り込み、目印の位置を把握できること。

1 については、自動書類運搬装置の走行の障害にならない目印を作成することができた。

2 については、まず面積計測に関しては、2 値化した画像から面積を計算し、候補を絞り色認識で目印を捉えることができた。しかし、自動書類運搬装置を使う環境により目印をうまくとらえることができない場合があった。2 値化の閾値が自動で設定されるプログラムになっているため、面積の計測での目印の絞り込みがうまくできなかったことが考えられる。解決法としては、閾値決定のアルゴリズムを、自動書類運搬装置を用いるのに適した別の方法にすることが考えられる。環境の違いが影響すると考えられるが、指定する目印の画素値の範囲を変えることで対応できるだろう。

ステレオ法による距離計測を実現するためには以下の要求仕様を満たす必要があった。

1. ステレオ法を実現するために、カメラ 2 つの最適なレンズ間距離を求められること。
2. ステレオ法を使用して、レンズと被写体までの距離を算出できること。

まず 1 については、レンズ間距離の実験の結果、10.5cm と 11.5cm のレンズ間距離を用いると、視差が広い範囲で確認できた。レンズ間距離を大きくしすぎると、視差の値が安定しないことがあるため、レンズ間距離は 10.5cm を採用することにした。

次に 2 については、レンズ間距離 10.5cm のカメラを作成し、そのカメラを用いて視差画像を作成した。距離と視差のグラフを作成し、そのグラフの近似曲線を作成し距離と視差の計算式を作成した。この計算式を用い距離計測を行った結果は、90cm～400cm の範囲で平均誤差 1.65cm の精度で距離を測定することができ、距離計測の誤差の目標を満たすことができた。しかし、ステレオカメラの精度の調整により視差がうまく取れない場合もあった。このことから、ステレオカメラの精度を向上することにより、より正確な距離計測が出来ると考えられる。

3.8.2 考察

最終的な評価として、目印認識では 200cm 先に置いた目印の面積と色を認識することができた。前後 20cm～30cm 程度のずれなら問題なく認識することはできるがそれ以上に大きくずれると認識できないことからカメラの精度と移動の際の精度が重要と考えられる。
距離計測ではレンズ間距離 10.5cm の場合、90cm から 400cm 範囲で距離を計測することができた。誤差の平均値は 1.65cm で目標の精度に収まったことから、自動書類運搬装置に用いる距離の算出方法として適していると考えられる。しかし、カメラのピントや上下のずれなどにより、視差が変化することが多かった。このことから、視差を計測する前に、カメラのピントと上下のずれを補正や調整を行うプログラムの作成や、より精度の高いカメラの製作が必要であることも分かった。

第 4 章 回路とそのプログラム（米山・石橋担当）

4.1 概要（石橋担当）

4.1.1 目的

本章では、ルンバの動作制御を行うための回路とそのプログラムについて解説する。

4.1.2 要求機能

本章において解説する、自動書類運搬装置の稼動を実現させる為に必要な機能は以下の通りである。

1. PC、Arduino、及びルンバの接続とシリアル通信。
2. Arduino を用いたルンバ内のモータを操作、及びルンバの移動の実現。
3. ルンバの移動距離と旋回角度の取得。

上に挙げた機能を実現させる為に、マイコンボードを介して PC とルンバに接続することにした。

本章は、これらの実現方法について詳細に解説していく。

4.2 ルンバの接続と制御（米山・石橋担当）

4.2.1 要求機能達成のための方法

要求機能の 1 つ目である『PC、Arduino、及びルンバの接続とシリアル通信』という条件を達成する為に、Arduino という初心者向けマイコンボードを介してルンバと PC を接続する。

また、要求機能の 2 つ目と 3 つ目である『Arduino を用いたルンバ内のモータを操作、及びルンバの移動の実現』、『ルンバの移動距離と旋回角度の取得』という条件を満たすプログラムを作成するために、ルンバに内蔵されている機能を利用する。ルンバにはエンコーダーが内蔵されており、ルンバの進んだ距離および角度を得ることができる。これを利用して、エンコーダーから得られる距離および角度が、指定した距離や角度と同じ値になった際に停止するプログラムを作成すれば良い。4.2.5 項からそれらのプログラムについて解説を行う。

4.2.2 使用機器

本研究で用いるマイコンボードは図 4-1 に示す Arduino MEGA である。その仕様を表 4-1 に示す。

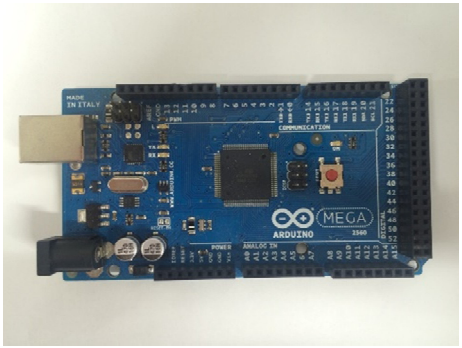

図 4-1　Arduino MEGA

表 4-1　Arduino MEGA の仕様表

使用マイクロコントローラ	ATmega2560
動作電圧	5V
入力電圧	7～12V
デジタル I/O ピン	54 本
アナログ入力ピン	16 本
長さ	101.52mm
幅	53.3mm
重さ	37g

4.2.3 オープンインターフェース

ルンバと Arduino MEGA を接続するために、ルンバオープンインターフェース（ROI）という端子に接続する(図 4-2、図 4-3)。ROI 端子とは、ルンバが外部機器と通信するための接続端子である。

図 4-4、図 4-5 のように自作したコネクタを使って接続すると、コマンド入力でルンバを制御することが出来る。

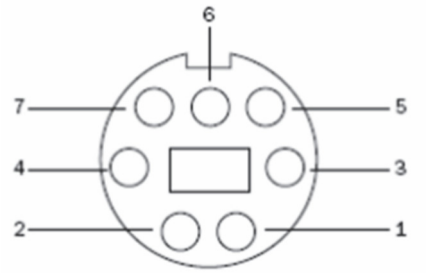

図 4-2　ROI 端子の形状［4-1］

図 4-3　ROI 端子の画像

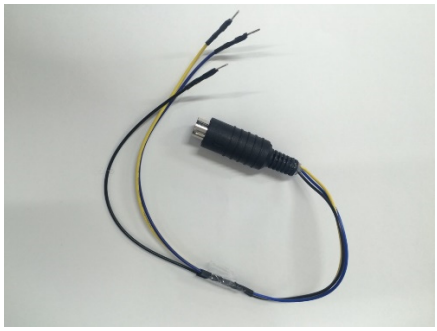

図 4-4　自作したコネクタ

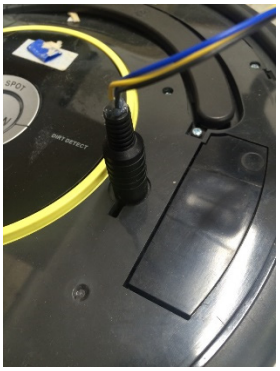

図 4-5　コネクタと ROI 端子の接続

表 4-2　ROI 端子の仕様表

1.電源（今回未使用）	5.ボーレート変更（今回未使用）
2.電源（今回未使用）	6.グランド(GND)
3.受信データ(RXD)	7.グランド(GND)
4.送信データ(TXD)	

4.2.4 ルンバとの接続方法

Arduino MEGA とルンバを図 4-6 のような配線で接続した。

PC とルンバを USB ケーブルで接続することによって Arduino に必要な電源は確保できる。

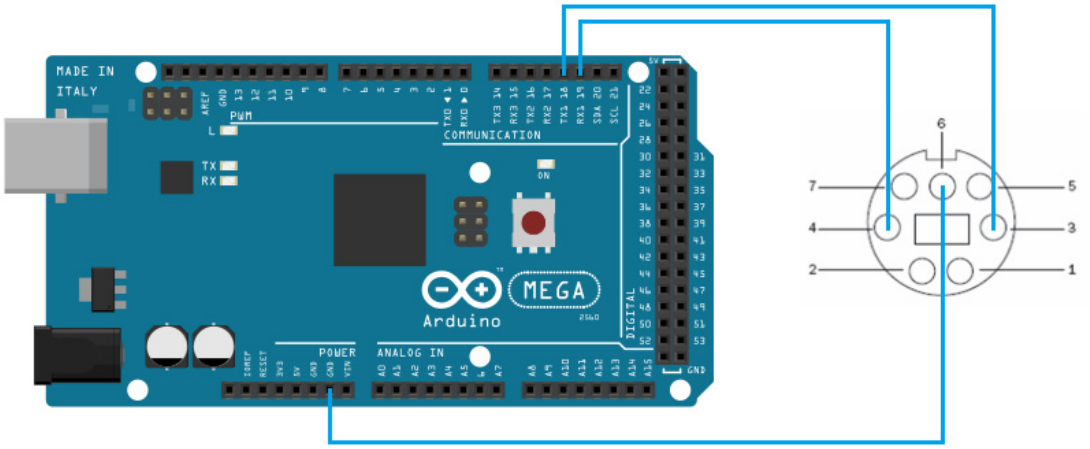

図 4-6　Arduino と ROI 端子の配線図 [4-1][4-2]

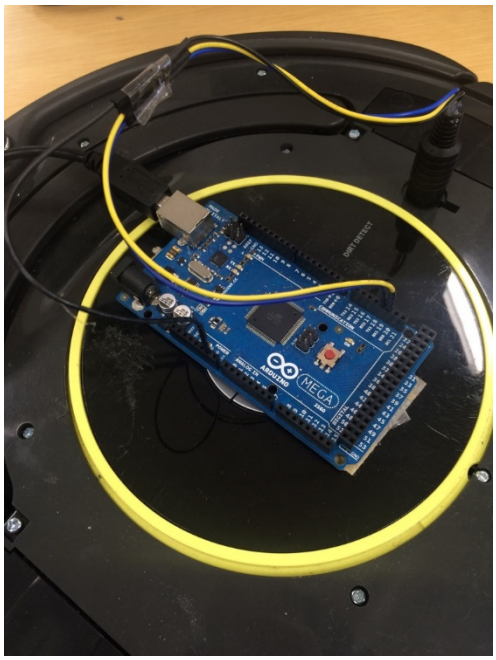

図 4-7　Arduino とルンバの配線

4.2.5 制御方法

ルンバの制御を行うためのコードは、「ルンバの ROI を使っての制御Ⅱ (Arduino ソフトウェア編)」(第７章[4-3])というサイトのプログラムコードを基に作成した。

変更点は以下の通りである。

1. 前進・後退を速度入力から距離入力に変更した(4.1.2 要求機能 2)。
2. 右・左旋回を速度入力から角度入力に変更した(4.1.2 要求機能 2)。

我々が用いるルンバの制御コマンドは表 4-3 の通りである。

表 4-3　制御コマンド一覧

行動パターン	行動コード	指定子	合計容量[Byte]
前進	w	4 桁の数値。1 桁目は 0 が+で 1 は-だが、本研究では 0 しか用いない。残りの 3 桁の数値は移動距離(mm)か角度（度）を表す	5
後退	x		5
右旋回	D		5
左旋回	A		5
停止	s	なし	1

上記のコマンドの後ろに指定した距離、角度を入力する。これらのコマンドをシリアル通信でルンバに送信する事で、ルンバの動作を制御することができる。

コマンドの入力例は以下の通り。

例１） 20cm 前進させる→「w0200」(入力時の距離の単位は mm である)

例２） 150 度右に旋回させる→「D0150」

ここからは表 4-3 のコマンドが入力された際に Arduino が行う処理について解説する。下記のコードは、表 4-3 に示されたコマンド一文字目の行動コードに対する、case 文による場合分けの記述である。

まず、ルンバに対する停止コマンドを受け取った際に実行されるコードは下記の通りである。

```
  //停止
  case 's': stopRoomba();
  break;
```

このコードによって呼び出される StopRoomba()という関数は下記の通りである。

```
//停止
void stopRoomba()
{
  // Stop
   RoombaSerial.write((byte)137);   // Drive
   RoombaSerial.write((byte)0x00);
   RoombaSerial.write((byte)0x00);
   RoombaSerial.write((byte)0x00);
   RoombaSerial.write((byte)0x00);
}
```

次は、ルンバに対する前進用のコマンドを受け取った際に実行されるコードである。

```
//前進
case 'w':
   foward( speedNum );
   break;
```

このコードでは、foward(speedNum)というルンバを前進させるための関数が呼び出される。引数 speedNum は実際には関数内で上書きされるので、引数としては用いていない。foward 関数は下記の通りである。foward 関数のコードは長いため、いくつかに区切りながら解説をする。

```
//前進命令
void foward(int speedNum)
{
   int StopNum;
   g_odometry_x = 0;
   StopNum = serialReadSpeed();
```

StopNum の値は PC 側から送信され、その値はルンバが停止するまでに動く距離である。g_odometry_x はルンバ自身が進んだ距離をエンコーダーから取得して格納する。g_odometry_x = 0 は、g_odometry_x の初期化である。

エンコーダーから取得したルンバの車輪の旋回数から移動距離を測定するのが下記のコードである。

```
//変化量
    float dX , //⊿X
          dY ,//⊿Y
          dAngle ,//⊿Y
          dL ,//左車輪の移動量
          dR;//右車輪の移動量


    int tempR, tempL , counterR , counterL;
    tempR = encoderOder(0);//右エンコーダー
    tempL = encoderOder(1);//左エンコーダー
    unsigned long updateTime = millis();
    if (      tempR - g_beforEncoderR < -32768 ) counterR = ((long)tempR + (long)65536 -
(long)g_beforEncoderR); //オーバーフローした場合
    else if ( tempR - g_beforEncoderR >  32768 ) counterR = ((long)tempR - (long)65536 -
(long)g_beforEncoderR); //マイナスになった場合
    else  counterR = tempR - g_beforEncoderR;
    if (      tempL - g_beforEncoderL < -32768 ) counterL = ((long)tempL + (long)65536 -
(long)g_beforEncoderL); //オーバーフローした場合
    else if ( tempL - g_beforEncoderL >  32768 ) counterL = ((long)tempL - (long)65536 -
(long)g_beforEncoderL); //マイナスになった場合
    else  counterL = tempL - g_beforEncoderL;
    g_beforEncoderR = tempR;//右エンコーダーの情報を更新
    g_beforEncoderL = tempL;//左エンコーダーの情報を更新
    dR = counterR * ENCODER_MM_PAR_PULSE;
    dL = counterL * ENCODER_MM_PAR_PULSE;
    dAngle = (double)(dL - dR) / L;
    g_odometry_Angl += dAngle / M_PI * 180.0;
    dX = ( dL + dR ) / 2.0 * cos( (g_odometry_Angl / 180) * M_PI ); //⊿X
    dY = ( dL + dR ) / 2.0 * sin( (g_odometry_Angl / 180) * M_PI ); //⊿Y
    g_odometry_x += dX;
    g_odometry_y += dY;
```

次はルンバに速度 70 で前進させる命令を送信するコードである。

```
byte sendSpeedData[2];
speedNum = 70;
sendSpeedData[0] = ( (byte *) ( (void*)(&speedNum) ) )[0];
sendSpeedData[1] = ( (byte *) ( (void*)(&speedNum) ) )[1];

RoombaSerial.write((byte)137);   // Drive
//速度指定
//配列の後ろから送信する
RoombaSerial.write(sendSpeedData[1]);   // Speed
RoombaSerial.write(sendSpeedData[0]);   // Speed
RoombaSerial.write((byte)0x00);  // Velocity: 0x00c8 = 200
RoombaSerial.write((byte)0x00);
```

RoombaSerial.write(sendSpeedData[1])、RoombaSerial.write(sendSpeedData[0])の部分がルンバの前進する速度を指定している部分である。ここの部分を speedNum = 70 という値に固定することにより、ルンバが前進する際の速度が一定になるようにしている。このコードによりルンバは前進し始める。

次はルンバが StopNum だけ前進した時に停止させるコードである。

```
if ( g_odometry_x >= ( StopNum ) && g_odometry_x <= (StopNum + 4) ) {
   Serial.print( StopNum ); Serial.print("¥n");

   RoombaSerial.write((byte)137);   // Drive
   RoombaSerial.write((byte)0x00);
   RoombaSerial.write((byte)0x00);
   RoombaSerial.write((byte)0x00);
   RoombaSerial.write((byte)0x00);
   delay(1);
   g_odometry_x = 0;   //オドメトリの X 座標
   g_odometry_y = 0;   //オドメトリの Y 座標
   g_odometry_Angl = 0;//オドメトリの角度
   break;
}
```

このコードは、PC 側から送信された値である StopNum とエンコーダーから得た値 g_odometry_x の値が一致した時に if 構文の中の停止コマンドが機能するようになっている。また、ルンバが確実に停止するように、StopNum～StopNum+4 の範囲でルンバを停止させている。

後退のプログラムは行動コードが'x'、「foward」が「backward」になっているだけで、後はほぼ同じである。

以上がルンバを前進させるためのコードである。

次は、右旋回及び左旋回の場合分けのコードである。

```
//右ターン
   case 'D':
   turnRight2( speedNum );  break;

//左ターン
   case 'A':
   turnLeft2( speedNum );   break;
```

このコードも前進や停止のコマンドと同様に、case 'D'で右旋回を行う turnRight2(speedNum)関数を呼び出し、case 'A'で左旋回を行う turnLeft2(speedNum)関数を呼び出す。左旋回と右旋回のコードは同様の構成であるため、ここでは右旋回のコードに絞って解説を行う。

右旋回のコードも長いため、区切りながら解説を行う。

```
void turnRight2(int speedNum)
{
  int StopNum;
  StopNum = serialReadSpeed() * (-1); //停止角度
  g_odometry_Angl = 0;
```

StopNum の値は PC 側から送信される旋回角度である。g_odometry_Angl は、ルンバに取り付けられているエンコーダーから取得したルンバ自身の旋回角度を格納する変数である。こちらも前進の命令の時と同様に、g_odometry_Angl = 0 で初期化している。なお、前進同様、引数 speedNum は後に上書きされるため引数としては機能していない。

エンコーダーから得られる情報によって、旋回角度を計算するコードは下記である。

```
//変化量
   float dX , //⊿X
         dY ,//⊿Y
         dAngle ,//⊿Y
         dL ,//左車輪の移動量
         dR;//右車輪の移動量


   int tempR, tempL , counterR , counterL;
   tempR = encoderOder(0);//右エンコーダー
   tempL = encoderOder(1);//左エンコーダー
   unsigned long updateTime = millis();
    if (      tempR - g_beforEncoderR < -32768 ) counterR = ((long)tempR + (long)65536 -
(long)g_beforEncoderR); //オーバーフローした場合
    else if ( tempR - g_beforEncoderR >  32768 ) counterR = ((long)tempR - (long)65536 -
(long)g_beforEncoderR); //マイナスになった場合
    else  counterR = tempR - g_beforEncoderR;
    if (      tempL - g_beforEncoderL < -32768 ) counterL = ((long)tempL + (long)65536 -
(long)g_beforEncoderL); //オーバーフローした場合
    else if ( tempL - g_beforEncoderL >  32768 ) counterL = ((long)tempL - (long)65536 -
(long)g_beforEncoderL); //マイナスになった場合
    else  counterL = tempL - g_beforEncoderL;
    g_beforEncoderR = tempR;//右エンコーダーの情報を更新
    g_beforEncoderL = tempL;//左エンコーダーの情報を更新
    dR = counterR * ENCODER_MM_PAR_PULSE;
    dL = counterL * ENCODER_MM_PAR_PULSE;
    dAngle = (double)(dL - dR) / L;
    g_odometry_Angl += dAngle / M_PI * 180.0;
```

次はルンバに旋回速度 15 で旋回させる命令を送信するコードである。

```
byte sendSpeedData[2];
speedNum = 15;
sendSpeedData[0] = ( (byte *) ( (void*)(&speedNum) ) )[0];
sendSpeedData[1] = ( (byte *) ( (void*)(&speedNum) ) )[1];
RoombaSerial.write((byte)137);   // Drive
//速度指定
//配列の後ろから送信する
RoombaSerial.write(sendSpeedData[1]);   // Speed
RoombaSerial.write(sendSpeedData[0]);   // Speed
RoombaSerial.write((byte)0xff);  // Radius: 0x0001 = Turn in place clockwise
RoombaSerial.write((byte)0xff);
```

RoombaSerial.write(sendSpeedData[1])、RoombaSerial.write(sendSpeedData[0])がルンバの旋回する速度を指定している部分である。これを speedNum = 15 という値に固定することにより、ルンバが旋回する際の速度が一定になるようにしている。ここのコードによりルンバは右旋回し始める。

次は、ルンバが StopNum だけ旋回した時にルンバを停止させるコードである。

```
   if ( g_odometry_Angl >= (-1) * ( StopNum + 4 ) && g_odometry_Angl <= (-1) * ( StopNum ) )
{
        RoombaSerial.write((byte)137);   // Drive
        RoombaSerial.write((byte)0x00);
        RoombaSerial.write((byte)0x00);
        RoombaSerial.write((byte)0x00);
        RoombaSerial.write((byte)0x00);
        delay(10);
        g_odometry_x = 0;   //オドメトリの X 座標
        g_odometry_y = 0;   //オドメトリの Y 座標
        g_odometry_Angl = 0;//オドメトリの角度
        break;
}
```

このコードも前進のコードと同様に、PC 側から送信された値である StopNum とエンコーダーから得た g_odometry_Angl の値が一致した時に if 構文の中の停止コマンドが機能するようになっている。また、ルンバが確実に停止するように、StopNum～StopNum+4 の範囲でルンバを停止させている。

以上がルンバを右旋回させるためのコードである。

最後に、表 4-3 の指定子を読み取り、距離や角度のデータとして返すコードを下記に示す。

```
 while ( true )
 {
  if (Serial.available() > 0)
  {
   temp = Serial.read();
   if (  ('0' <= temp) && (temp <= '9') )
    {
     switch ( readByte )
     {
      case 0:
         StopdNum += (temp - '0') * 1000;
         break;
      case 1:
         StopNum += (temp - '0') * 100;
         break;
      case 2:
         StopNum += (temp - '0') * 10;
         break;
      case 3:
         StopNum += (temp - '0') ;
         break;
      }
      readByte++;
    }
  }
  //4 文字受信したならばループ終了
  if ( readByte >= 4 ) break;
}
return StopNum;
```

4 文字の指定子を順に受け取っているが、それぞれ千、百、十、一の位の値を意味する。それらを 4 桁の数値として返している。これが、前述の forward 関数や turnRight2 関数内で移動距離や旋回角度として利用される。プログラム上は 4 桁の数値を与えられるようになっているが、表 4-3 に示したように、本研究では 3 桁までの数値しか用いていない。

4.2.6 ルンバの動作テスト

4.2.5 項で解説したプログラムの動作確認テストを行った。

図 4-8 の様に、PC からルンバに対して、全てのコマンドを入力し、動作が正常に行われるか一つずつ確認していった。どの動作もスムーズに行うことが出来た。

図 4-8　PC と Arduino を用いたルンバの操作テスト

4.3 回路とそのプログラムの最終評価

ルンバ制御班の作成すべき操作プログラムの動作を確認した。

任意の距離の前進・後退と、任意の角度の右・左旋回が出来た。また、同時にルンバ自身の走行距離や旋回角度を計測することが出来た。

だが、停止する距離や角度の範囲に収まらず、停止しないまま進んでしまう事も稀にあった。これを解決するための案は二つ考えられる。まず一つは、停止範囲の設定変更である。今回作成したコードにおいては、停止する範囲を設け、値がその停止範囲に入ったときに停止するように設定していた。しかし、前進・旋回の値はどちらとも単調増加であるため、停止範囲の上限を無くせば良いのではと考えられる。もう一つは、速度の変更である。一定に稼動するよう設定していた移動速度及び旋回速度を少し遅くすれば、より確実に停止出来るようになるのではと考えられる。

第 5 章　自動書類運搬装置の動作確認(中川担当)

5.1 概要

図 5-1 の完成した自動書類運搬装置の動作確認を行う。実験は工学院大学八王子キャンパス 4 号館 8 階の廊下で行った(図 5-2)。

図 5-1　完成した自動書類運搬装置

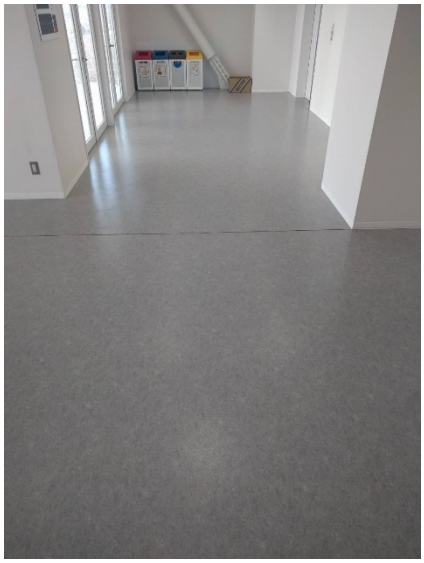

図 5-2　実験を行った廊下

5.2 実験方法

図 5-3 は実験を行った場所の模式図である。A 点を初期位置として、B 点、C 点の順に 200cm ずつ移動できるかを検証する。B 点、C 点にはそれぞれ目印が配置されている。A 点には自動書類運搬装置を配置する。このとき、自動書類運搬装置の向きは、B 点を正面にとらえた状態から、60° 右向きにずれた状態で配置する。

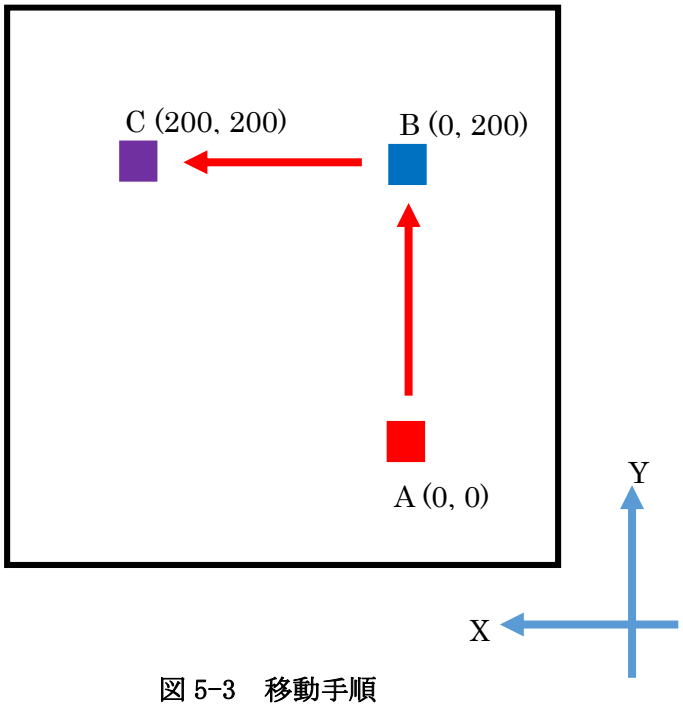

図 5-3　移動手順

5.3 実験結果

実験の結果、自動書類運搬装置は目印の場所を正確に計測し、A 点を初期位置として、B 点、C 点の順に 200cm ずつ自動で移動することができた。図 5-4 は動作中の自動書類運搬装置である。

しかし、以下の 2 つの誤作動を起こす場合もみられた。

1. 目印以外の物体を検知してしまう。
2. 目印までの距離を計測できず、目印を越えて走行してしまう。

1 は、床や壁を検知したことが原因である。2 は、ステレオカメラの上下とピントがずれてしまうことが原因である。

図 5-4　動作中の自動書類運搬装置

5.4 考察

5.3 節の実験結果から自動書類運搬装置が目標の動作を実行することが確認できた。このことから、製作した自動書類運搬装置は、目印を一定間隔にオフィスに配置すれば運用が可能であると考えられる。

しかし、1 と 2 のような誤作動を起こす場合もあった。1 は、夜に実験を行ったため、2 値化の閾値が自動で設定されるプログラムにおいて、白く映るべきところが黒く映ってしまったことが原因である。解決策としては、閾値決定のアルゴリズムを、自動書類運搬装置を用いるのに適した方法に変更することが考えられる。2 は、ステレオカメラのレンズの固定部で上下がずれたためピントが合わず、視差が得られなかったことが原因である。解決策としては、ステレオカメラのレンズの固定部の精度を上げることが考えられる。この誤作動は 2 の場合の方が発生する割合が高く、改善する優先度としてはこちらの方が高いと考えられる。

自動書類運搬装置を運用するためには、様々な場所で成功率を高めなくてはならない。よって、この 1 と 2 の誤作動を解決することが本システムの課題である。

第 6 章 結論（堀井担当）

6.1 目標

我々が製作した自動書類運搬装置は、画像処理の技術を用いて自動走行し、オフィス内で書類のような荷物を人間の代わりに目的の場所まで運搬するシステムである。

6.2 目標への達成度

我々は、目標の条件をすべて満すべく研究を行った。表 6-1 において、〇が達成、△が一部達成、×が未達成を表している。

表 6-1 目標を達成するための条件

距離計測を実現するためのステレオカメラの製作	△
自動書類運搬装置の外観の考案と製作	〇
製作したステレオカメラ、収納部、及びルンバとの接合	〇
目的地となる目印の製作	〇
色認識を用いた目的地の把握	△
ステレオ法を用いるのに適したレンズ間距離の算出	〇
ステレオ法を用いた自動書類運搬装置と目印までの距離の測量	△
PC、Arduino、及びルンバの接続とシリアル通信	〇
Arduino を用いたルンバ内のモータを操作、及びルンバの移動の実現	〇
ルンバの移動距離と旋回角度の取得	〇

設計班は画像処理班のレンズ間距離の実験を基に、レンズ間距離 10.5cm のステレオカメラを製作することができた。また、そのステレオカメラを用いて、画像処理班で距離を計測することが可能なことも確認できた。そして、ルンバや棚等をステレオカメラと組み合わせ、自動書類運搬装置の機体を製作することができた。しかし、ステレオカメラの精度により、距離が測定できないこともあった。このことから、ステレオカメラは距離計測を行うために求められる機械的な精度の向上が必要であると考えられる。

画像処理班は対象物を面積で絞り込んだ後、色を認識して目印を定めその座標を計測することができた。しかし、撮影環境により誤検知が発生する場合があった。また、レンズ間距離の実験を行い、本実験で利用するレンズ間距離を決定した。そして、その結果を用いた新たに製作したステレオカメラを用いて視差画像を作成し、距離の測定を行った。距離は平均誤差 1.65cm の精度で測定することでき、ステレオ法で距離を計測することができた。しかし、ステレオカメラの状態によっては、安定した視差の値を得られず、正しい距離を計測できない場合もあった。

ルンバ制御班は Arduino を使用し、任意の距離への前進・後退、任意の角度への右・左旋回が可能な動作プログラムを作成した。また、ルンバ自身の走行距離や旋回角度の計測も可能となった。

6.3 問題

前節で述べたように各班が目標を達成したが、自動書類運搬装置にはいくつかの以下の問題点もあった。本節では、これらの問題点とその解決法について記述する。

6.3.1 問題点

自動書類運搬装置の問題点は以下の 3 点である。

問題 1. ステレオカメラで撮影される画像の上下とピントがずれることがある。

これは、ステレオカメラの機械的な精度が求められる精度に達していないためである。

問題 2. 目印以外の物体を認識してしまうことがある。

これは、自動書類運搬装置を用いる場所の照明の環境によって、面積の絞り込みと色認識の設定がうまくいかないことがあるためである。

問題 3. 目印までの距離を計測できず、目印を大きく越えて走行してしまうことがある。

これは、ステレオカメラの左右のカメラの画像のピントがずれている場合、視差が得られないことがあるためである。

6.3.2 解決法

前項の問題点に対して考えられる解決法はそれぞれ以下の通りである。

問題 1 の解決法

ステレオカメラのレンズの固定部の精度を高めることで、画像のずれがなくなり安定した視差が得られると考えられる。

問題 2 の解決法

閾値決定のアルゴリズムを、自動書類運搬装置を用いるのに適した別の方法にすることが考えられる。

問題 3 の解決法

この問題についても問題の解決法 1 で解決すると考えられる。

6.4 総合評価

我々が製作した自動書類運搬装置の動作確認を行ったところ、自動書類運搬装置は目的地まで自動で移動することが確認できた。しかし、ステレオカメラの状態や、照明の環境により誤作動を起こす場合もあった。これは、ステレオカメラの精度を改善することと、2 値化のアルゴリズムを変更することにより解決すると考えられる。また、それにより、使用環境に関わらず、高い成功率で目的地まで移動できると考えられる。よって、今回の研究では、誤作動を起こす場合もあるが、目的地まで書類を運搬することが可能だったことより、目標を達成できたと考えられる。

第 7 章 参考文献・URL

[1-1] SUBARU EYESIGHT（アイサイト）のすごさ｜NAVER まとめ
https://matome.naver.jp/odai/2134836475594154801/2134849641908610003

[1-2] データ出力タイプ/UXD-30LX-EW｜HOKUYOU 北陽電気株式會社
http://www.hokuyo-aut.co.jp/search/single.php?serial=14

[1-3] 日経ビジネス ONLINE ｜品切れを見つける自走ロボ、その名は「Tally」
http://business.nikkeibp.co.jp/atcl/report/15/061700004/060100113/?SS=nboimgview&FD=1125424127

[1-4] GIZMODO ｜これが物流センターの最前線。コンピュータが管理する「Amazon Robotics」潜入レポート
http://www.gizmodo.jp/2016/12/amazon-robotics.html

[1-5] U-NOTE ｜壁や天井がディスプレイに！Cerevo がプロジェクター搭載ロボット「Tipron」を発売
http://u-note.me/note/47506530

[2-1] HD Pro Webcam C920r ｜ Logicool
http://www.logicool.co.jp/ja-jp/product/hd-pro-webcam-c920

[2-2] ルンバ 622(アイロボット)の比較と評価 ｜ ロボット掃除機　比較ガイド
http://okaimono-navi.info/irobot/622/

[2-3]ウッドトップテーブルチェスト ET-W430 ｜ アイリスプラザ
http://www.irisplaza.co.jp/Index.asp?KB=SHOSAI&SID=K218682

[3-1] サンスター文具　色画用紙　B4　1 冊｜ASKUL
http://www.askul.co.jp/p/7804515/

[3-2] Point Cloud Readering｜OpenCV.jp
http://opencv.jp/opencv2-x-samples/point-cloud-rendering

[4-1] Arduino から Roomba を Software Serial で動かす
http://qiita.com/DUxCA/items/36a13e68722c51c72927

[4-2] Fritzing
http://fritzing.org/download/

[4-3]ルンバの ROI を使っての制御
http://flanker711.blog49.fc2.com/blog-entry-97.html

使っている人いるのかな？OpenCV3.0 から入ったラベリング処理について｜Qiita
http://qiita.com/wakaba130/items/9d921b8b3eb812e4f197

OpsnCV HSV で肌色検出｜mintu’s プログラミング日誌
http://d.hatena.ne.jp/mintsu123/20111123/1322065624

第 5 日目：SLT②｜一週間で身につく C++言語の基本
http://cpp-lang.sevendays-study.com/ex-day5.html

カメラキャリブレーションと 3 次元構成｜open v2.1 C リファレンス
http://opencv.jp/opencv-2.1/cpp/camera_calibration_and_3d_reconstruction.html

ルンバの ROI を使っての制御Ⅱ（Arduino ソフトウェア編）
http://flanker711.blog49.fc2.com/blog-entry-98.html

Arduino を用いた移動位置推定ノードの開発
http://www.net.c.dendai.ac.jp/~nkubota/sotsuron.html

建築発明工作ゼミ 2008: Arduino デジタルコンパス／HMC6352
http://kousaku-kousaku.blogspot.jp/2008/08/arduinohmc6352.html

Arduino 日本語リファレンス
http://www.musashinodenpa.com/arduino/ref/

みんなの Arduino 入門　株式会社リックテレコム（2014）

たのしい電子工作　Arduino で電子工作をはじめよう！[第２版]　秀和システム（2013）

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適したレンズの提供等のご支援いただいたK社、並びに、2年間ご指導ご鞭撻を頂きました金丸隆志准教授、矢崎敬人准教授、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。

画像処理に基づく自動書類運搬装置の開発

指導教員　金丸 隆志 准教授　　副指導員　矢崎 敬人 准教授
G1-13007 池 宏武　　G1-13033　境 亮祐　　G1-13049　中川 勇佑　　G1-13063 布施 勇樹
G1-13064　堀井 孝亮　　G1-13078　米山 耕介　　G1-12009 石橋 知大

1.　緒言

本研究ではK社から提供して頂いた「続・車載カメラをこんなものに付けたら」というテーマを基にステレオカメラによる距離計測を用いた開発を模索していく。

開発するにあたり、ステレオカメラは画像から距離の情報を得られることが可能であることに着眼し、ステレオカメラ以外に人間の手足となるものと考える脳となるものがあれば、人間のように自動で走行する装置が製作できるのではないかと考えた。

そこで、我々は「自動書類運搬装置」を開発することに決定した。これは、書類のような荷物を運ぶ作業を、人間の代わりに行うシステムのことである。

2.　原理

本研究で開発する自動書類運搬装置の実現には移動手段(ハードウェア)と対象認識(ソフトウェア)の二つが必要となる。一つ目の移動手段に関しては、我々は家庭用お掃除ロボットのルンバを使用することにした。ルンバの制御には、PCとマイコンボードであるArduinoを使用する。二つ目の対象認識とは、「目的地を見つけること」および「目的地までの距離の把握」を行う手段のことである。目的地を見つけるには、K社から提供されるレンズを用いて製作するステレオカメラを用いる。目的地には色分けされた目印を設置するため、ステレオカメラの一方のカメラを利用して、色認識を行う。同じステレオカメラを用いて、目印までの距離を計測することもできる。図1が本システムの動作イメージである。

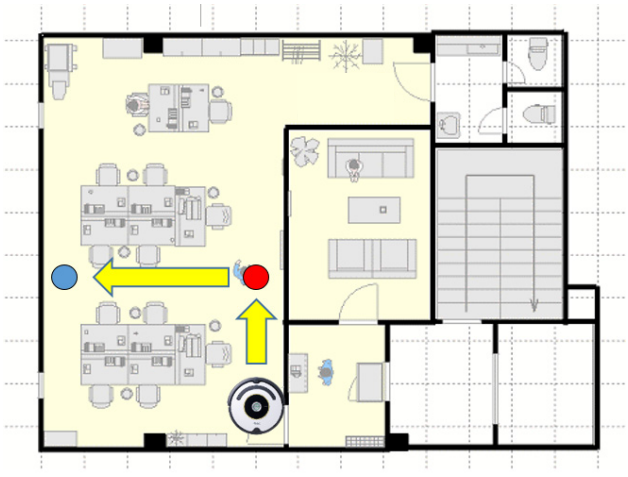

図1　システムが動作するイメージ

3.　自動書類運搬装置の製作

図2は開発した自動書類運搬装置の全体図である。PCとステレオカメラ、ArduinoはUSBで接続し、Arduinoとルンバは ROI 端子で接続する。PCはステレオカメラから得た情報を基に目的地までの場所と距離を把握し、その情報をArduino送信する。Arduino は受け取った情報を用いてルンバを制御する。

図2　開発した自動書類運搬装置

4.　実験

作成したプログラムを用いて、一連の流れの動作確認の実験を行った。目印把握の流れは図3の通りである。

まず、ルンバに 360° 旋回のコマンドを送信する。目印の位置を把握する為に、面積計測を用いて対象を絞り込み、その後色認識を用いて目印の位置を確認する。目印が確認されなかった場合は、また 360° 旋回を始める。目印が確認できた場合は、ルンバに旋回を停止するコマンドを送信する。旋回停止後は視差画像を作成して距離を算出する。得られた距離をルンバに送信し前進させる。

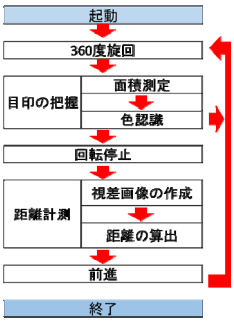

図3　システムの動作フロー

5.　結言

(1) 距離計測を実現するためのステレオカメラと自動書類運搬装置の製作。
(2) 色認識により目印を見つけることができた。
(3) ステレオ法を用いた自動書類運搬装置と目印までの距離の測量するプログラムを作成した。
(4) Arduino とルンバを接続してシリアル通信と移動を可能にした。