修 士 論 文

移動ロボット群向けの個体識別可能なロボット間位置計測システム

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

岩鼻　利幸

2007 年 3 月

修 士 論 文

移動ロボット群向けの個体識別可能なロボット間位置計測システム

指導教官　DEFAGO Xavier 助教授

審査委員主査　DEFAGO Xavier 助教授
審査委員　片山卓也 教授
審査委員　丁洛榮 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

510012 岩鼻　利幸

提出年月: 2007 年 2 月

概 要

近年，レスキューやサッカーロボットの開発など, 移動ロボット群システムの開発が盛んに行われている．移動ロボット群システムは，大規模なタスクの一部を担う単純な機能を備えた複数のロボットで構成されるが，1 台の高機能なロボットに比べ，よりロバストなシステムになると期待される．また，本質的に複数のロボットを必要とするタスクも数多くある．

移動ロボット群がタスクを協調して実行するためには，ロボット間の相対的な位置情報を実時間で認識することが重要である．また，カメラの視覚的情報を持っていてロボット間の通信が可能で各ロボットの IP の情報は持っているロボット群システムにおいて，ロボットを指定して通信したいとき，カメラでロボットは見えるが識別できないので，見ているロボットと通信しているものを一致することができないという問題がある．そのため，ロボットの個体識別が必要となる．

本研究では，各ロボットの個体識別と他のロボットの位置（角度と距離）を計測するシステムを提案する．各ロボットに全方位カメラと固有のカラーコードを搭載する．全方位カメラでこのカラーコードを読み取り，解析して各ロボットの個体識別と他のロボットの位置を測定する．本手法を実装し，POV-Ray で作った simulation 画像を用いて本システムの有効性を考察した．

目 次

第1章　はじめに

1.1　研究背景

近年，レスキュー，サッカーやロボットの開発など, 移動ロボット群システムの開発が盛んに行われている．ロボット群システムは，大規模なタスクの一部を担う単純な機能を備えた複数のロボットで構成されるが，1 台の高機能なロボットに比べ，よりロバストなシステムになると期待される．また，本質的に複数のロボットを必要とするタスクも数多くある．移動ロボット群がタスクを協調して実行するためには，ロボット間の相対的な位置情報を認識することが重要である．また，各ロボットの特性が異なる場合や大規模のロボット群でタスクを実行する場合適材適所にロボットを動かす必要性が生じ各ロボットの個体識別機能が要求されることが考えられる．実際に移動ロボット群のフォーメーション制御の研究において，ロボットの性能に近傍のロボットを識別でき，相対位置を正確に計測することができることを求めている．[1] これ以外でも，ロボット間で相手を指定して通信したいときにもロボットの個体識別が必要となる．カメラの視覚的情報を持っていてロボット間の通信が可能で各ロボットの IP の情報は持っているロボット群システムにおいて，ロボットを指定して通信したいとき，カメラでロボットは見えるが識別できないので，見ているロボットと通信しているものを一致することができないという問題がある．そのため，ロボットの個体識別が必要となる．今後，ロボット群システムの発展に伴い汎用性の高いロボット間位置計測システムの要求はさらに高まると考えられる．

移動ロボット群システムでのロボット間位置計測システムには，コストが安く環境のインフラストラクチャ(以下インフラ) に依存しない汎用性の高いシステムが求められる．また機能としては，障害物や周囲の環境の変化にロバストであることや実時間に一度で周囲の複数台のロボット位置を計測できることが重要である．これまで，超音波，光学センサやステレオカメラを用いてロボット間距離を計測する手法が一般的である．しかし，前者は干渉によるエラーが多いことや障害物とロボットの識別ができないこと，後者は一度に全方向のロボットを測定するのが困難であることより，これらの条件を満たすものではない．

そこで本研究では，移動ロボット群が一般的な協調動作をする際に使用するロボットの位置計測システムの構築を目的とする．具体的には，他のロボットの個体識別と自分自身を基準とした他のロボットまでの距離と角度の計測，以上 3 点を全方位カメラとロボット固有のコードを利用してインフラに依存せず実時間で同時に行える手法を提案しシステムを構築する．

1.2 関連研究

移動ロボット群の位置情報システムは，大きく分けてグローバルとローカルなシステムの２つに分けることができる．

グローバルなシステムとは，ロボットがある環境のどこにいるかを知るシステムである．我々に身近なものでは，GPS(Global Positioning System) があり地球上の現在の位置を調べるための衛星測位システムである．移動ロボット群向けのグローバルな位置計測システムとして，可動物体位置検出システムなどがある．この原理は，ロボットの移動領域を写すことができるようにテレビカメラを天井に固定する．ロボットには位置検出用の赤外 LED を搭載し，赤外線透過フィルタ付きのカメラで検出することで，照明条件の変化する環境下でも安定した位置同定が可能である．このシステムを利用すると，ロボットの位置と角度をカメラで検知することができ，ロボットに固有のマークをつけると個体識別を行うこともできる．[1] で提案したフォーメーション制御を実験するのもこのシステムを応用している．お互いに共通した位置情報を持てるというメリットがあるが，インフラを整備する必要があり移動ロボット群の使用環境や範囲が限られてしまう．

ローカルなシステムとは，自分と他の物体との位置関係を知るシステムである．このシステムの代表的なものとして超音波式センサ，光学式センサや CCD カメラを 2 台使用したステレオカメラ方式がある．超音波式センサと光学式センサは，多数商品化されており移動ロボット群に簡単にのせることができるが，一軸方向の距離しか測定できないので，ロボットの周囲に隙間ができないようにセンサを配置する必要がある．複数台の移動ロボットが複数台のセンサを使うと干渉によるエラーの問題もある．また，自らスキャンニング光や波を発し対象物の距離を測るため，どんな対象物でも測ることはできるが障害物とロボットの識別などはできない．ステレオカメラ方式は，障害物とロボットの識別はできるが一度に複数台のロボットを測ることはできない．

カメラの観測視野がレンズの画角により制限されるという欠点を解決し，周囲 360 度の情報を一度に観測できるセンサとして全方位カメラ [2] がある．近年，全方位カメラを搭載した移動ロボットも少なくない．これらの場合，全方位カメラは周囲の大まかな環境，障害物の有無まどを認識するために搭載されていることが多い．そのため，全方位カメラ以外に対象物までの距離を測るために光学式センサ [3] やステレオカメラ [4] を搭載している移動ロボットシステムがある．全方位カメラで対象物の角度を計測し，その角度にフォーカスして光学式やステレオカメラで対象物までの距離を計測する．全方位カメラと光学やステレオセンサを搭載することでコストアップとなり，一度に複数台のロボットを測ることができないなどのデメリットがある．[5] は全方位カメラのみで対象物までの距離を計測している．移動ロボット群の相対的な位置計測している．ロボットは容易に背景から識別できる視覚的特長をもつとしているためロボット間の角度は計測することができるが，ロボットの識別はできず，他のロボットまでの距離も画像中のロボット本体の多きさよりおおまかな距離計測しかできない．

そこで，本研究ではインフラに依存せず複数台のロボットを同時に計測することができるロボット間位置計測システムを提案し構築する．移動ロボット群において他のロボット

までの距離と角度とロボットの個体識別を行うロボット間位置計測する．

1.3 本論文の構成

本論文は 6 章で構成される．第 2 章では Background として本システムを実装する上で必要となる知識についてと本システムで想定している群ロボットシステムと使用環境について述べる．第 3 章では提案するシステムの手法と実装について述べる．第 4 章ではシミュレーションで用いる画像の構築について述べる．第 5 章では提案手法のシミュレーション結果と考察，第 6 章で結論を述べる．

第2章　予備知識とシステムの定義

本章では，まず本システムを実現するうえで必要となる画像処理の基本知識とシミュレーション画像を作成するときに使用するソフトである POV-Ray について説明する．次に，本システムで想定するロボットと環境についての定義を行う．

2.1　予備知識

2.1.1　画像処理の基本

ディジタル画像

画像とは，２次元平面上の強度濃度分布である．コンピュータで画像を扱うには，画像を数値として表現しなければならない．濃度値の２次元配列として，コンピュータで扱いやすくしたのがディジタル画像である．横方向に M 画素，縦方向に N 画素からなるディジタル画像において，画像中の画素位置を示す添字は，図 2.1 に示すように慣用的に f_{ij} と表記されることが多い．以後，この表記法を用いる．

２値化

濃淡がなく，白と黒しかない図形・画像は，画像の値が 0，1 の 2 つの値しかとらないので 2 値画像と呼ばれる．画像の特徴を解析するためには，画像から対象物を切り出し，対象物と背景を分離するために２値化処理を行うことが多い．画像の２値化は，次式の閾値処理によって行われる．

$$f_t(i,j) = \begin{cases} 1; f(i,j) \geq t \\ 0; f(i,j) < t \end{cases} \tag{2.1}$$

通常，結果の 2 値画像 $f_t(i,j)$ の中の値 1 の部分は対象図形を，値 0 の部分は背景を表す（図 2.2，図 2.3）．ここで問題となるのは閾値 t の決め方で t を決める方法は閾値選択といわれ，代表的な手法として，単一手動閾値方式や p-タイル法などがある．単一手動閾値

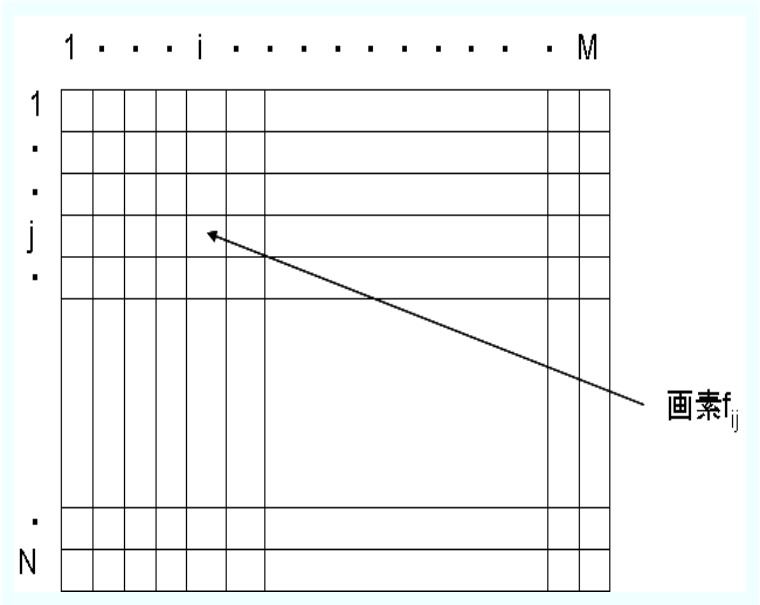

図 2.1: ディジタル画像中のがぞの表記のしかた

方式とは，指定された色深度を基準として，その値より入力画素の色深度値が明るければ白，暗ければ黒色として 2 値化する．このとき，出力画像は初期状態で黒色となるので，入力画像の画素値が閾値以上の大きさのときのみ出力画像へ画素値を書き込むことで，多少の高速化を図っている．p-タイル法とは，画像全体のうち背景色がどの程度の割合かを指定することでその閾値を決定し，その閾値で単一手動閾値方式による 2 値化を行う．このとき、閾値を決定するために入力画像全体の濃度ヒストグラムを取得し，得られたヒストグラムから指定された黒画素の割合と符合するような閾値を決定する．

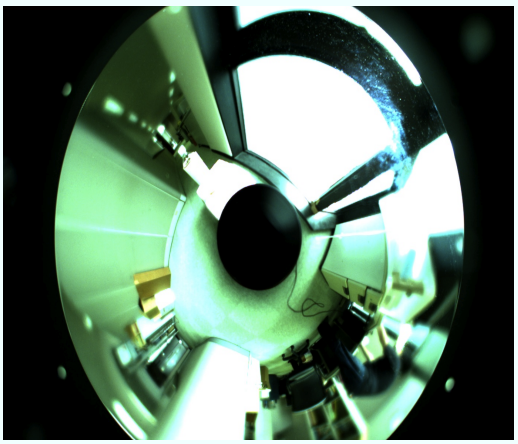

図 2.2: 入力画像

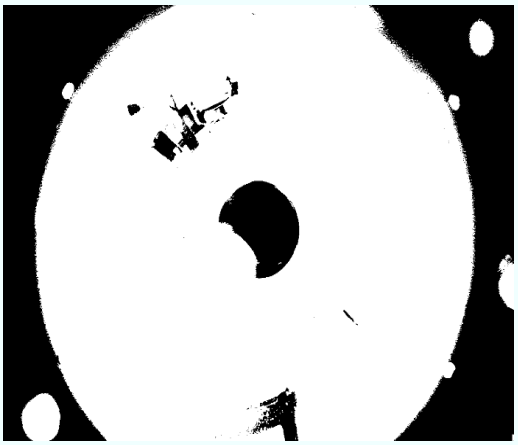

図 2.3: ２値化画像

2.1.2　全方位カメラ

通常のカメラは一度に一方向しか観察できない．全方位カメラとは，魚眼レンズ，円錐，多角錐，球面，双曲面ミラーなどのような光学系を用いて３６０°全方位がみえるようにしたカメラである．本システムでは，双曲面ミラーを用いた全方位カメラを用いるため，以下双曲面ミラーを用いた全方位カメラについて述べる．全方位カメラは鉛直上向きに設置したカメラと，その上に設置した双曲面状の鏡から構成される．この鏡は取り外しが可能であり，通常のカメラの上部に取り付けることで全方位の撮影が可能となる．本手法で用いた全方位カメラの構成を図 2.5 に示す．

図 2.4: 全方位カメラ外観

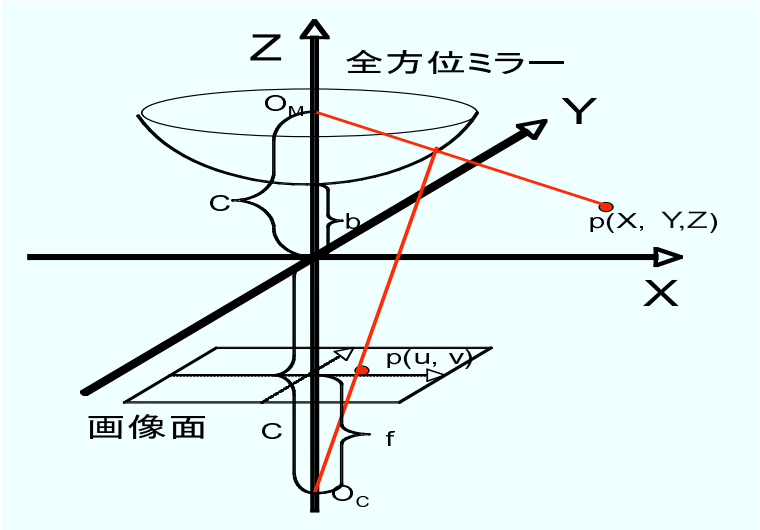

図 2.5: 光学系の構成

鏡の焦点 O_M とカメラのレンズ中心 O_C は，二葉双曲面が持つ 2 つの焦点 $(0, 0, +c)$, $(0, 0, -c)$ に位置し，画像面となる uv 平面は XY 平面に平行で，カメラのレンズ中心 OC

からカメラの焦点距離 f だけ離れた平面とする．鏡の双曲面および, OM, OC は次式で示される．

$$\frac{X^2+Y^2}{a^2}-\frac{Z^2}{b^2}=-1(Z>0) \tag{2.2}$$

ここで a, b は双曲面の形状を定義する定数である．鏡の焦点 O_M に集まる像は，双曲面の鏡を介してカメラのレンズ中心 O_C に集まる．したがって，O_C にレンズの中心をおいたカメラで O_M への全方位画像を撮影することができる．

また，図 2.6 の様に点 P と Z 軸を含む鉛直断面を想定すると，点 P と写像点 p の間には次の関係が成り立つ．

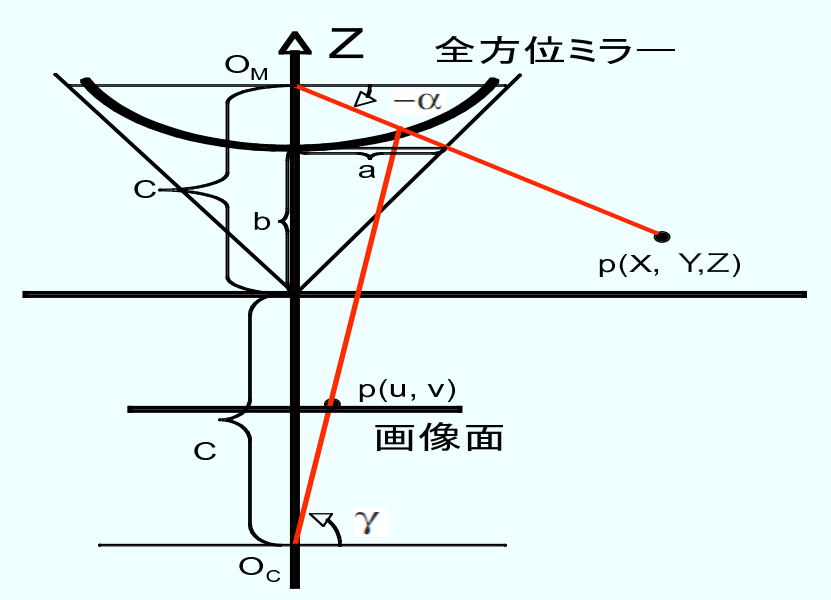

図 2.6: 点の射影鉛直方向

$$\begin{aligned} Z &= \sqrt{X^2+Y^2}\tan\alpha+c \\ \alpha &= \tan^{-1}\frac{(b^2+c^2)\sin\gamma-2bc}{(b^2-c^2)\cos\gamma} \\ \gamma &= \tan^{-1}\frac{f}{\sqrt{u^2+v^2}} \end{aligned} \tag{2.3}$$

即ちミラーの焦点 O_M からの点 P の方位角 θ および伏角 α は，カメラのレンズ中心 O_C を双曲面の焦点位置にすることで，写像点 $p(u,v)$ より，一意にもとまる．このとき，ミラーの焦点 O_M は固定なため，入力画像をミラーの焦点 O_M からみたカメラを鉛直軸周りに回転して得られる画像や一般のカメラの画像に変換できる．

また，式 2.3 を x，y を求める形にしたのが次式である．

$$u = \frac{Xf(b^2-c^2)}{(b^2+c^2)(Z-c)-2bc\sqrt{X^2+Y^2+(Z-c)^2}} \tag{2.4}$$

$$v = \frac{Yf(b^2-c^2)}{(b^2+c^2)(Z-c)-2bc\sqrt{X^2+Y^2+(Z-c)^2}} \tag{2.5}$$

また，$p(u,v)$ が与えられれば，式 2.6，2.7 より X と Z，Y と Z の関係がわかり，点 $P(X,Y,Z)$ と鏡の焦点 O_M を通る直線を求めることができる．

$$Z = \frac{-f(b^2-c^2)+2bc\sqrt{X^2+Y^2+f^2}}{(c^2-b^2)u}X + c \tag{2.6}$$

$$Z = \frac{-f(b^2-c^2)+2bc\sqrt{X^2+Y^2+f^2}}{(c^2-b^2)v}Y + c \tag{2.7}$$

2.1.3　パノラマ変換

全方位カメラで撮影した全方位画像は双曲面ミラーを用いて周囲のシーンを平面上に投影するため，周囲のシーンがその方位によって円周上に配置された同心円状の画像として得られる．その一例を図 2.7 に示す．

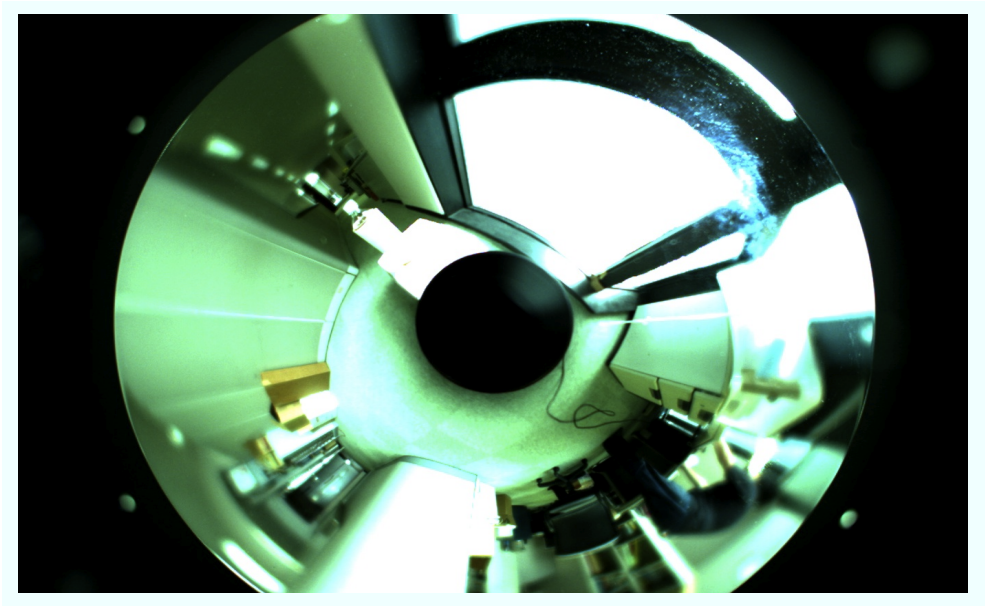

図 2.7: 全方位画像

従来のパターン認識や画像処理手法の多くは，画素が方眼状に並んでいることを前提としており，そのままでは図 2.7 のような全方位画像には適用しづらい．そのため，このような画像を画素が方眼状になるように表現形式を変換して用いることが多い．この変換がパノラマ変換である．

これまでの研究ではこの変換は極座標と直交座標の間の単純な座標変換として表現されてきた．具体的には，パノラマ画像中の座標を直交座標表現で $(x_{i,j}, y_{ij})'$，全方位画像中の座標を極座標表現で $(r_i, \theta_j)'$ として，以下のように表される．

$$\begin{cases} x_{i,j} = r_i cos\theta_j \\ y_{i,j} = r_i sin\theta_j \end{cases} \tag{2.8}$$

変換した画像は，全方位画像の極座標上で半径方向，円周方向のそれぞれについて等間隔にリサンプリングしたときの画像変換になっており，その画像例を図 2.8 に示す．

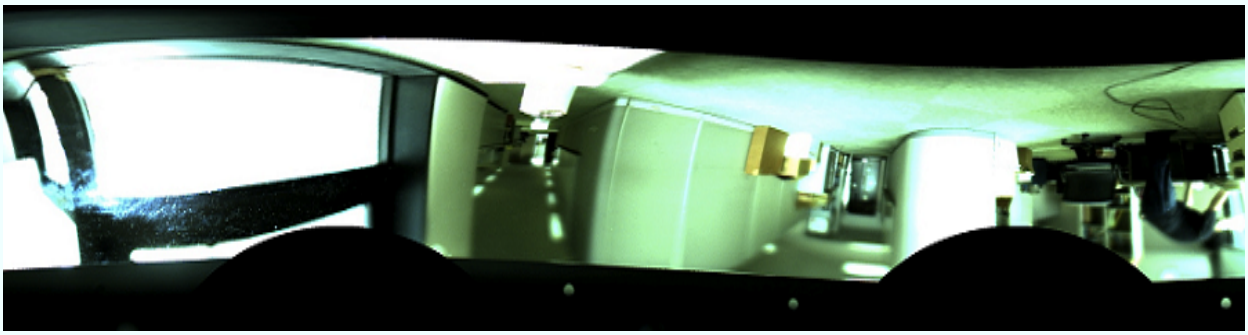

図 2.8: パノラマ変換画像

この画像は等間隔リサンプリングによるため，カメラから物体までの距離によって縦横の比率が異なっており，見た目上違和感のある画像であり，またパターン認識などを適用した場合も認識率の面で高い結果が得られにくいなどの欠点がある．

2.1.4 POV-Ray

シミュレーションで使用する画像をつくるためのソフトである POV-Ray について述べる．

POV-Ray(Persistaence of Vision Ray-Tracer) とは多くのコンピュータプラットフォームで利用できるレイ・トレーシングソフトウェアである．さらに，同じソースコードであればどのプラットホームでも同じ画像を得ることが可能である．プログラムのソースコードが一般に公開されているオープンソースの３ D レンダリングエンジンの一つで, 独自の C 言語風の構造化ドキュメントによりデータを入力し，マクロ関数による半自動配置ができるので，シミュレータとしても利用も可能である．簡単に言うと，どこにどんな形のものがあって、光源はどこで、それをどこから見る、ということを指定することにより、光源から出た光がどのようなルートを通ってカメラまで到達するかを計算することにより画像を作る．

POV-Ray のシーンファイルを書くときに最低限必要なのは，カメラと光源と物体の三つである．POV-Ray で画像を作るのは，実際のカメラを使って撮影するのと似ています．カメラを置く場所やカメラが向く方向，光源の光の強さや色，物体の形や大きさ，色などを指定して画像を作る．シンプルな例として以下のようなコードで図 2.9 のような画像を作ることができる．

```
//カメラの設定
  camera{
    location <0,0,-3>     //カメラを置く場所
    look_at <0,0,0>     //カメラの向く方向
  }
//ライトの設定
  light_source {
    <100,100,-100>       // ライトを置く場所
    color rgb <1,1,1>     // ライトの色
  }
//物体の設定
  object {
    sphere {<0 , 0 , 0> , 1}     // 物体の種類
    texture {
      pigment { rgb <0 , 1 , 0> }    // 物体の色
    }
  }
```

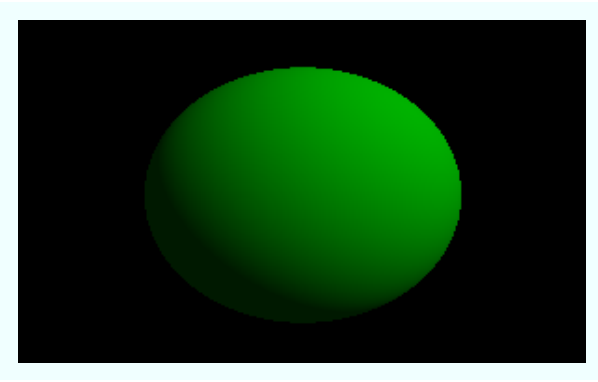

図 2.9: POV-Ray の画像例

2.2　システムの定義

本研究では，複数の移動ロボットが自分に搭載した全方位カメラだけを用いて，他のロボットを識別し位置情報を計測することを目標としている．この際，ロボットと環境について以下の仮定を置く．

- ロボットは，地面と水平に各自全方位カメラと固有のカラーコードを持つ．
- ロボットは，水平状態を保ったまま移動または静止でき，床面は水平である．
- 全方位カメラの特性，ロボットの高さとカラーコードの大きさは既知である．
- カラーコードまでの高さとカラーコードの大きさは全ロボット同じものとする．
- カラーコードは全方位カメラの下に全方向から同一に見えるようにつける．

以上の条件の下で，ロボット間位置計測計測手法を提案する．

第3章　実装

本章では，全方位カメラとロボット固有のカラーコードを利用したロボット間位置計測の提案と実装について述べる．提案する手法についての概要と全体の流れについて説明し，各フローについて詳しく説明する．

3.1　システムの概要

図 3.1 に本システムの概要を示す．各ロボットに全方位カメラと固有のカラーコードを搭載する．各ロボットは，自分に搭載された全方位カメラで全方位の静止画を撮り込み，カラーコードの特徴を抽出し易くするために前処理の画像処理を行い，取り込んだ静止画からカラーコードの情報以外を排除した画像をつくる．この画像でカラーコードを解析して他の全ロボットについての，個体識別，自分を基準にした角度と距離の 3 点を計測する．他のロボットに依存することなく，各ロボットが各自で距離を計測することができるため，他のロボットの故障には影響を受けない．

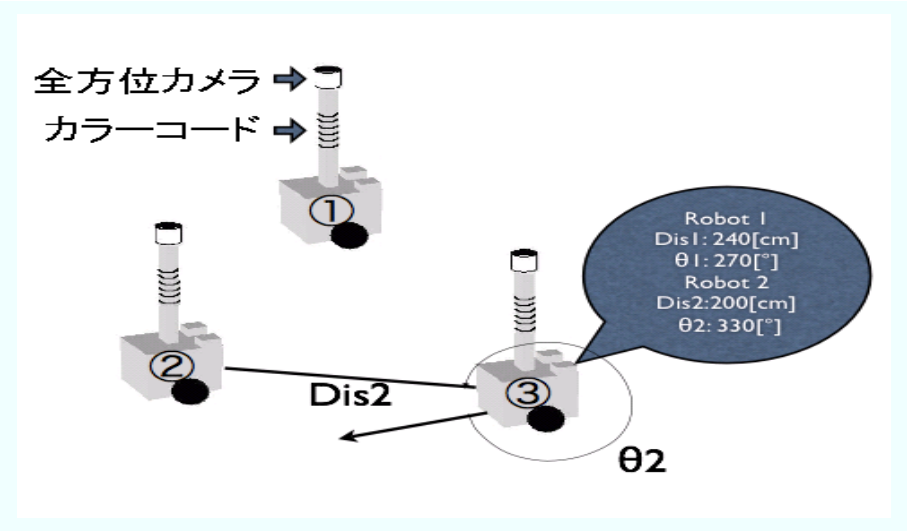

図 3.1: ロボットの構成

3.2　カラーコード

本システムで使用するカラーコードについて述べる．

本システムでロボット間位置を計測する場合，ロボット固有のカラーコードと全方位カメラの性能によって位置の精度が左右される．また，カラーコードの読み取り精度はCCD カメラの画素数によって決まる．画素数が多ければ分解能が上がるが処理時間が長くなる．カラーコードを大きくすれば読み取り精度と読み取り距離が上がるがロボットに搭載するため大きくするのにも限度がある．本システムで用いるコードを設計するにあたって，以下の点を考慮する必要がある．

(i) 全方位どこからみても，コードの見え方が一定であること．

(ii) 障害物など周囲の環境がら識別し易いこと．

(iii) ロボットの台数．

(iv) 計測可能距離．

まず (i) についてだが，全方位カメラの利点は全方位の画像を一度に撮れることである．よって全方位から見え方が同じコードを用いないとそのメリットが失われる．このことを考えて，円柱コードを書く．カメラを使って読み取るコードの代表例として，QR コードとバーコードがある．

図 3.2: バーコードと QR コード

QR コードとは、２次元コードの一種であり、「リーダにとって読み取り易いコード」を主眼にデンソーウェーブが開発した．QR コードは縦と横の２方向に情報を持つことで情報量を多く持つことができるものである．このコードは，情報量の多さ読み取り易さにおいて本システムへの適用に魅力を感じたが，２次元なので全方位どこから見ても同じ見え方にならず，(i) の条件を満たさないので本システムには適さない．また，バーコードは一次元で円柱に書いても全方位から同じに見ることができるので本システムに適していると考えられる．次に，(ii) と (iii) についてだがこれらの条件を満たすには，コードの情報量を増やすことが有効でる．情報量を増やせば自然界にはあまりない特徴をもつことが

でき，ロボット ID の情報も増える．しかし，これは (iv) とのトレードオフを考えなければならない．距離が遠くなるとそれだけ画像に写るコードの pixel 数は小さくなる．多い情報量を載せると計測可能距離が短くなる．バーコードは，白と黒の 1 ビットの組み合わせで，多くの情報を載せたバーコードを読み取るには多くの pixel 数を必要とする．そこで，本研究ではカラーコードする．例を図 3.3 に示す．

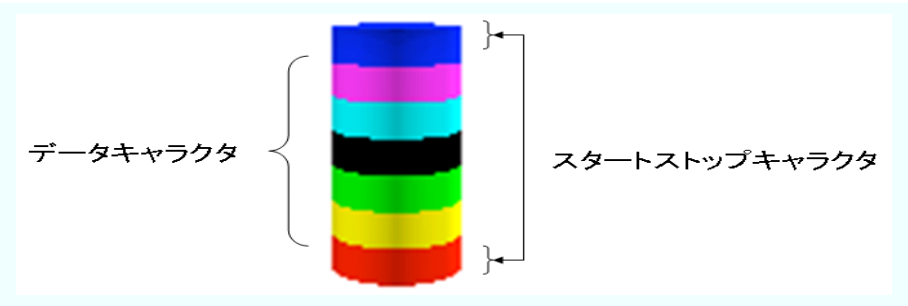

図 3.3: ロボット固有のカラーコードの構成

データキャラクタを何個にするのかと何色使用するのかを考えないといけない．これは，ロボットに搭載可能なカラーコードの大きさ，計測可能距離，必要なロボット ID 数の 3 つを考える必要がある．すなわち，タスクやロボットの種類によって適切なコードを決めるので，各自のシステムで違うものとなる．本研究では，タスクについては考えていないので，ロボットとカメラの性能を考慮して 5 個 6 色のカラーコードとする．

スタートストップキャラクタ コードの両端に，識別しやすいようにカラーコードチェッカーとして青色と赤色とする．これらは，カラーコードシンボルの始めと終わりを示すもので個体識別情報は持っていない．

データキャラクタ ここに計 6 色の色を使いロボット ID 情報を書く．使用する色は，黒，赤，緑，黄，シアン，マゼンタとする．同じ色が連続で続くことを許可すると，読み取りの距離と精度のパフォーマンスが悪くなるので同じ色が連続で続かないものとする．そうする合計 $6 \times 5 \times 5 \times 4(3000)$ 台までのロボット ID をもつ．

3.3　システムの流れ

ここではシステム全体の処理についての流れを説明し，各処理について概要を説明する．各処理の詳細については次節以降で述べる．

提案手法全体の流れを図 3.4 に示す．

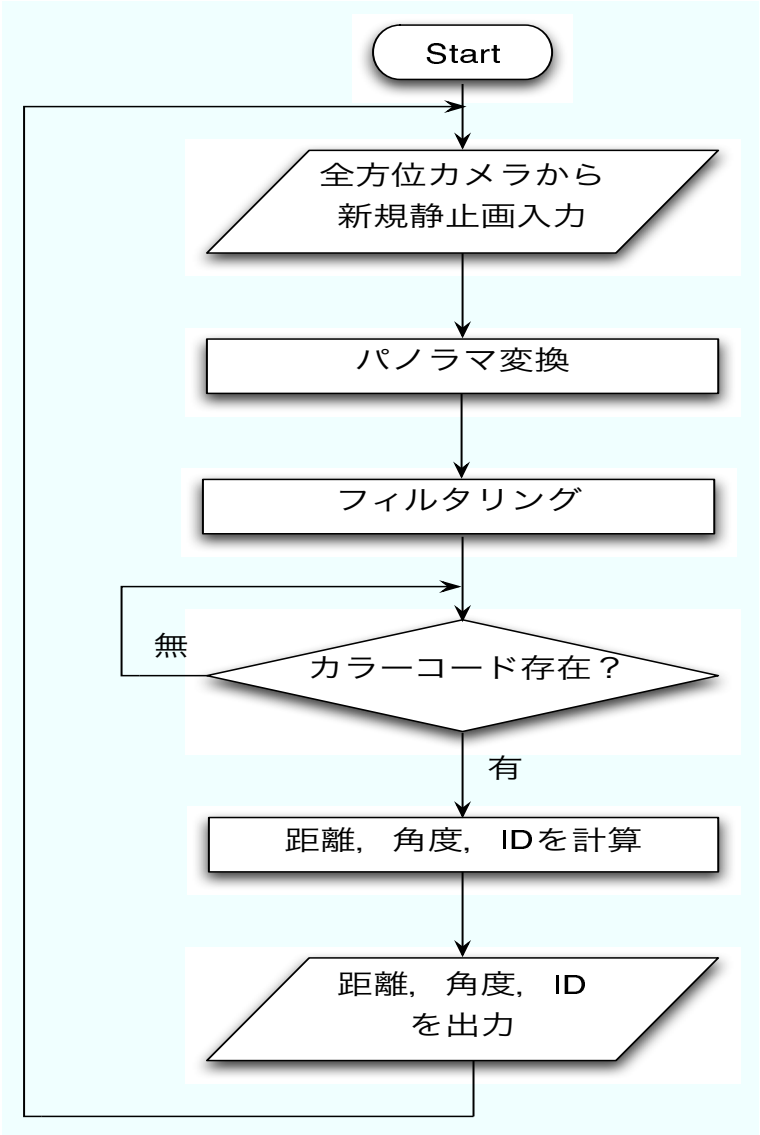

図 3.4: システム全体の流れ

パノラマ変換

全方位カメラから得られる画像は，ミラーを用いて平面に投影されるため，周囲の対象シーンが同心円状に撮影された画像となりそのままでは一般的な画像処理を適用することが困難である．そこで，パノラマ変換を行い，以降の処理を行い易くする．詳細は 3.3.1 節で述べる．

フィルタリング

パノラマ画像に対してカラーコードで使用する色の RGB データを閾値として各色で 2 値化し，カラーコードだけの画像を作りカラーコードを探し易くする．詳細は 3.3.5 節で述べる．

カラーコードの解析

カラーコードの特徴の特徴をもとにコードを探す．カラーコードの色情報から，ロボットの ID を計算する．詳細は 3.3.6 節で述べる．

角度の計算

全方位カメラの特性で，撮影した全方位画像にある対象物体の方位角 θ が，その物体の画像面上の写像の方位として直接現れる．すなわち，見つけたカラーコードが画像のどこにあるかで角度を計算することができる．詳細は 3.3.2 節で述べる．

距離の計算

個体識別と角度は画像の情報から直接計算することができるが，距離は求めることができない．そこで本研究では，実際の距離と画像に写るカラーコードの位置データをパターンわけしパターン認識を用いて距離を計算する．詳細は 5.1 節で述べる．

3.3.1 パノラマ変換

2.1.3 で述べたように，パノラマ変換は極座標から直交座標変換をする式 (2.8) を用いる．画像の半径を r[pix] とすると円の外周は $2\pi r$[pix] となる．外周を 1[pix] ずつ $2\pi r$[pix] まで直交変換する．以下にコードと画像を示す．

```
//panoramic extension

for (int q=0; q<pi2r; q++){
   a = a+(1/radius);
   cos = (double) Math.cos(a);
   sin = (double) Math.sin(a);
   for (int p=0; p<(int)(radius-2); p++ ) {
     if (Math.abs(p * sin) <= (h_center-1)) {

       panoramic[p][q] =
       image[h_center-Math.round(p*sin)][w_center+Math.round(p*cos)];

     }  else {
       panoramic[p][q] = 0;
     }
   }
}
```

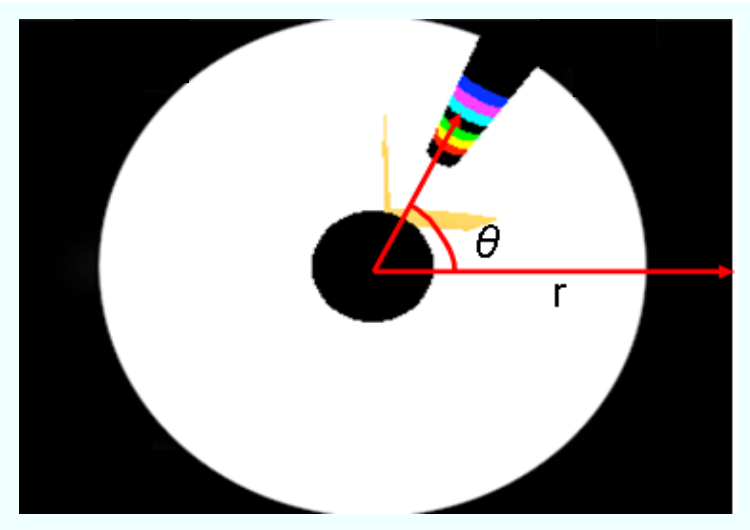

図 3.5: 全方位画像

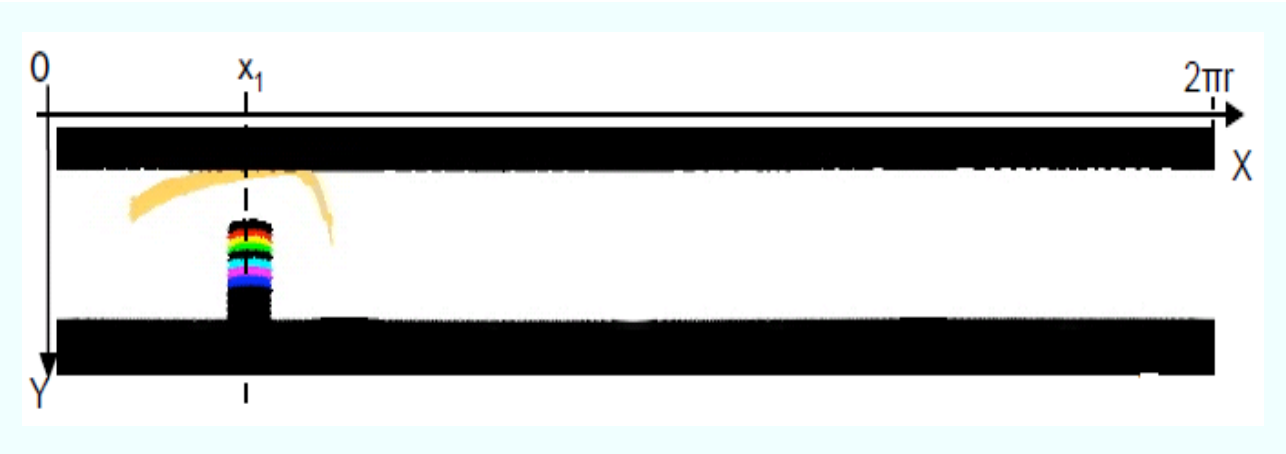

図 3.6: パノラマ画像

3.3.2 角度計測

3.3.3 角度計測の原理

図 2.5 に示しすように，空間内の任意の点 $P(X, Y, Z)$ に対する画像上での写像点を $p(u, v)$ としたとき，点 P の方位角 θ は次式で表される．

$$tan\theta = Y/X = y/x \tag{3.1}$$

すなわち Y/X で定まる点 P の方位角 θ は，y/x で定まる写像点 P の方位角 θ を算出することで得られる．このように, 撮影した全方位画像にある対象物体の方位角 θ が，その物体の画像面上の写像の方位として直接現れる．

3.3.4 全方位画像とパノラマ変換画像の角度の関係

前章で述べたように，見つけたカラーコードが画像のどこにあるかで角度を計算することができる．全方位画像のままで処理する場合は，画像に対するカラーコードの位置がわかれば角度 θ は式で簡単にもとまる．しかし，2.2 章のシステム全体の流れで述べたように，角度を計算する過程は全方位画像をパノラマ変換してバーコードを探してからである．パノラマ変換後の画像での角度は，カラーコードの中心の X 座標を x_1 とすると次式により求まる．

$$\theta = x_1 \times \frac{180}{\pi \times r}[\ ^\circ] \tag{3.2}$$

3.3.5 フィルタリング

本研究では，各色の式 (3.3) に示す RGB データの閾値できれいにフィルタリングできるのもとする．

$$f_t(i,j) = \begin{cases} 1; R(i,j) > 200, G(i,j) < 50, B(i,j) < 50\ (\text{赤}) \\ 2; R(i,j) < 50, G(i,j) < 50, B(i,j) > 200\ (\text{青}) \\ 3; R(i,j) < 50, G(i,j) < 50, B(i,j) < 50\ (\text{黒}) \\ 4; R(i,j) < 200, G(i,j) > 200, B(i,j) < 50\ (\text{黄}) \\ 5; R(i,j) < 50, G(i,j) > 200, B(i,j) < 50\ (\text{緑}) \\ 6; R(i,j) < 50, G(i,j) > 200, B(i,j) < 50\ (\text{シアン}) \\ 7; R(i,j) > 200, G(i,j) < 50, B(i,j) > 200\ (\text{マゼンタ}) \\ 0; (\text{その他}) \end{cases} \tag{3.3}$$

赤 (1), 青 (2)，黒 (3)，黄 (4)，緑 (5)，シアン (6)，マゼンタ (7) としその他の色を 0 として配列に入れる．これにより，カラーコードの情報のみを持った配列をつくることができ，色情報を抽出する．

3.3.6 カラーコードの解析

カラーコードの解析の流れを以下に示す．

(i) フィルタリングした画像上の配列を左上から X 軸方向に走査して赤を探す．赤があれば Y 軸の方向に青を探し青があれば，データキャラクタを走査してロボットの ID を計算する．

(ii) カラーコードの情報として図 3.7 に示す A-D のピクセル数をカウントする．

(iii) A-C よりコードの端点座標をだし，複数台に対応する．

図 3.7: カラーコードの情報

ただし，0[°] 付近では，パノラマ変換をするとカラーコードが切れてしまうことにも注意しなければならない．付録に，これらカラーコードの情報を抽出するコードとそれらの情報を距離，ID，角度情報として出力するところまでのコードを示す．

第4章　シミュレーション画像の構築

本章では，pov-ray を用いてシミュレーション画像の構築を行う．実際に実験で使用する機器のパラメータを用いて画像を作る．実験するには，全方位カメラとロボットと部屋が必要であるので，それぞれパラメータとコード，画像例を紹介する．

4.1　全方位カメラ

全方位カメラはカメラと全方位ミラーから成る．カメラは pov-ray ではライブラリを使えば簡単に実現でき，画素数も指定できる．全方位ミラーは，ライブラリにはないので作る必要がある．2.1.2 節で述べたように全方位ミラーは式 2.2 の双曲面の特性をもつ．式 2.2 と双曲面の形状を定義する a と b により作ることができる．カメラを鉛直上向きに，全方位ミラーから焦点距離だけ離して置くことにより全方位カメラを実現した．カメラ，レンズと全方位ミラーのパラメータを表 4.1〜表 4.3 に示す．全方位ミラーのパラメータ a,b,c は，ミラーの販売元である株式会社映像様の希望により非公開とする．

表 4.1: カメラのパラメータ

Marlin F-145C2	
センサ	SONY IT CCD
有効ピクセル数	1392 × 1038 [pixel]
オプティカルフォーマット	1/2

表 4.2: レンズのパラメータ

FUJINON HF9HA-1B	
焦点距離	0.09 [m]
絞り範囲	F1.4-16
最短撮影距離	0.1 [m]
画角	39.09 [°]×29.52 [°](水平 × 垂直)

表 4.3: 全方位ミラーのパラメータ

Eizoh Hyper70	
俯角	50 [°]
仰角	30 [°]
カメラ最短撮影距離	0.1 [m] 以下
a	販売元の希望により非公開
b	
c	

```
//カメラの設定
camera {
  location <0, 0.538, 0>      //カメラを置く場所
  look_at  <0, 0.61, 0>       //カメラの向く方向
}

//全方位ミラーの設定
#declare Mirror =
union {
   intersection {
     quadric {
        <1/(a*a),-1/(b*b),1/(a*a)>, <0,0,0>, <0,0,0>, 1    //メタル色の双曲面
        texture { F_MetalE }
     }
       cylinder {
        <0, 0., 0>,<0, 0.092, 0>, 0.045     //双局面の終点を指定
        pigment { color Black }
     }
}

//ミラーを支える支柱
       cylinder {
        <0, 0.045, 0>,<0, 0.0451, 0>, 0.0018
        pigment { color Black }
     }
}

//ミラーを置く位置
object {
    Mirror
    translate  0.509*y
}
```

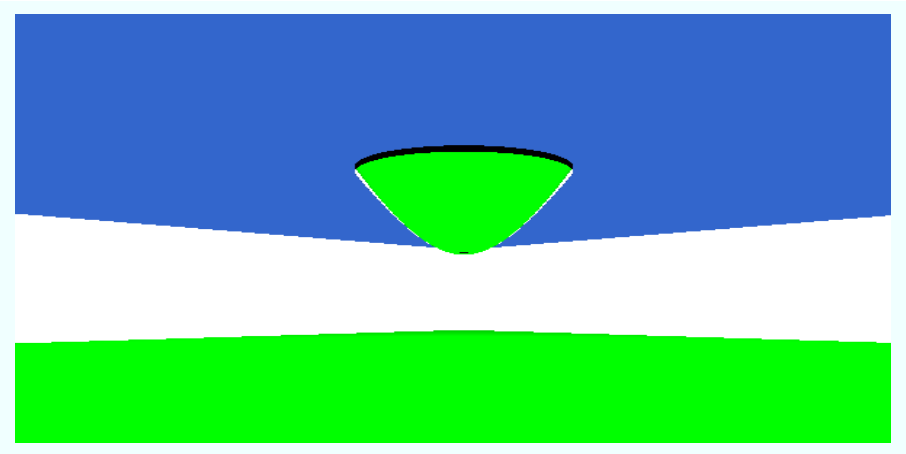

図 4.1: ミラーの画像

4.2 ロボット

ロボットのパラメータについて説明する．移動ロボット群のロボットには，特に決まりはなく各研究で異なったロボットを使っている．本研究では実際にロボットを使用しないので，一般的なロボットの大きさで適当な値に決めた．ロボット固有のカラーコードの大きさは，簡単に作れることより，1.5 リッターのペットボトルに A4 の大きさでプリントアウトしたカラーコードを巻きつけた大きさとする．以下に詳細なパラメータを示す．

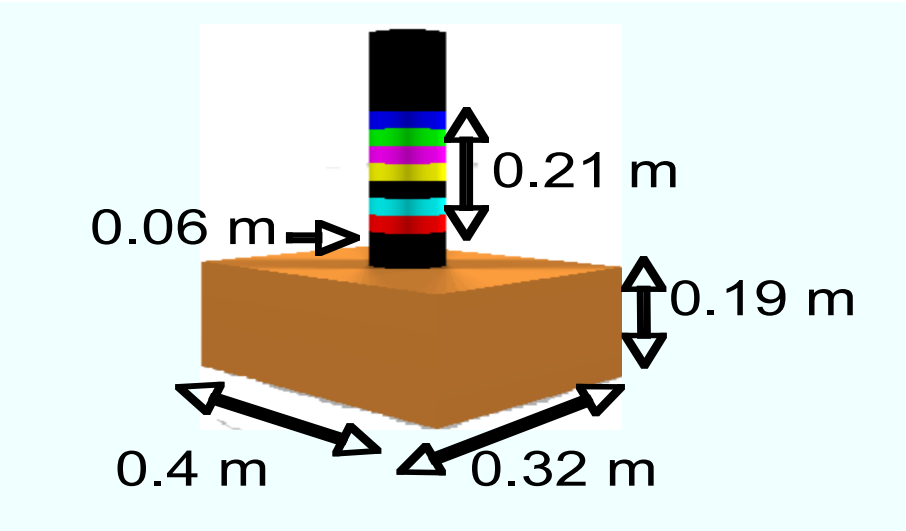

図 4.2: ロボットのパラメータ

ロボットは，POV-Ray のライブラリにある長方形と円柱を組み合わせ，色を指定することにより簡単に作ることができる．しかし，このままだと図 4.3 に示すようにカラーコー

ドに影ができてしまい，計測に影響がでる．そこで，ロボットにカラーコードを照らすライトを載せて影ができないようにした．

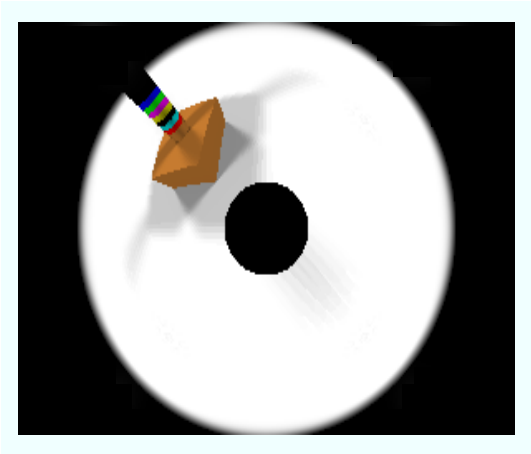

図 4.3: カラーコード影の処理をしてない画像

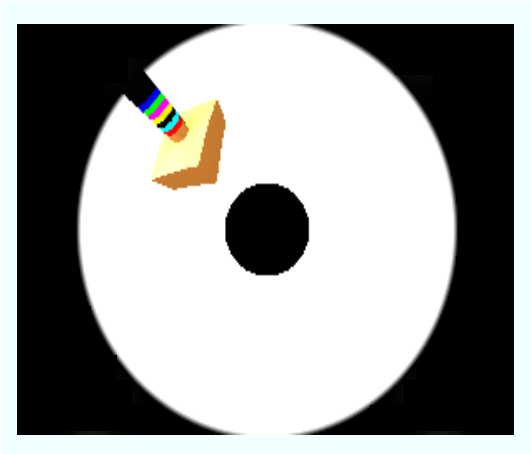

図 4.4: カラーコード影の処理後の画像

4.3　部屋

部屋は，理想的な環境を想定しロボットのカラーコードの識別に影響のない背景色とする．今回は，床の色と壁の色をともに白色とした．また，形は立方体とし大きさは縦 20[m] 横 20[m] 高さ 2[m] とする．

(2,0.5,2) に置いたカメラから (0.0.6,0) を写した画像例を図 4.5 に示す．ミラーが写るようにこの画像のみ壁の色を白色とした．画像の中央に写っているのが全方位ミラーである．このと同様の環境を全方位カメラで撮ると図 4.6 のようになる．

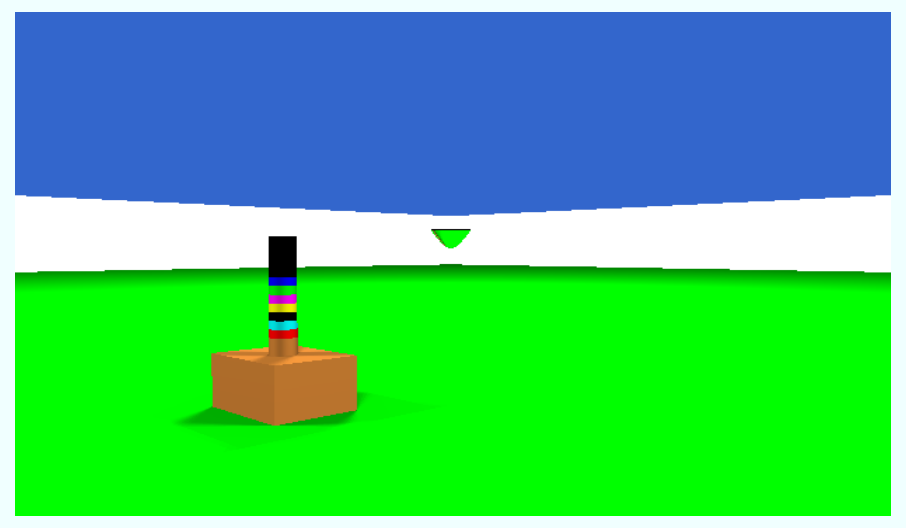

図 4.5: 部屋の画像

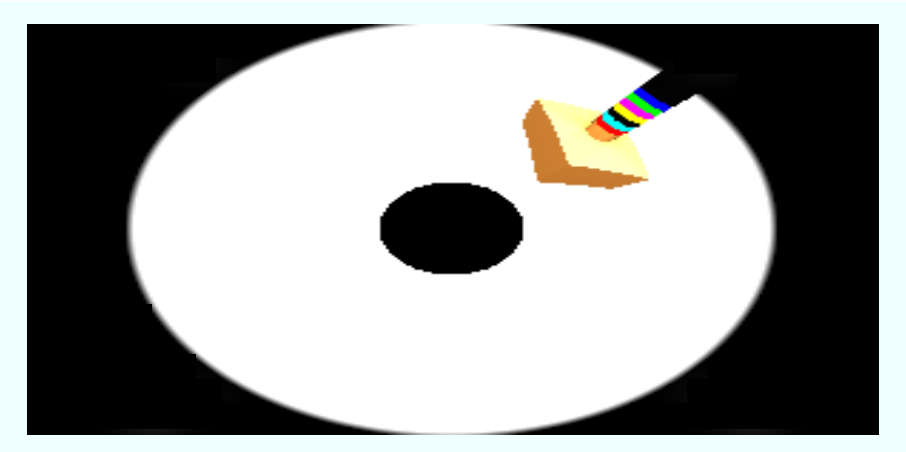

図 4.6: 全方位カメラで撮った画像

第5章　シミュレーション

距離のマッチングデータの取得とシステムの性能評価を目的とする．

シミュレーションは，写真のようなリアルな３次元グラフィックスを作ることのできるPOV-Rayで画像を作成し，javaで構築したシステムを用いておこなった．POV-Rayは`Version3.6.1(g++ 4.0.1 @ powerpc-apple-drawin8.8.0)`，javaは`Version1.5.0_06`を使用した．まず，予備シミュレーションとして，プロトタイプシステムを用いた実際の距離と写真に写るカラードの情報とのマッチングデータもモデル化を行う．

次にシステムの性能評価シとして角度，距離，ロボットIDのエラー測定とロボット台数の増加と処理速度の関係について評価する．

5.1　予備シミュレーション

5.1.1　距離測定用のマッチングデータ

個体識別と角度は画像の情報から直接計算することができるが，距離は求めることができない．そこで本研究では，実際の距離と画像に写るカラーコードの位置データをモデル化したデータを用いて距離を計算する．POV-Rayでシミュレーション環境の画像をつくりカラーコードの写る大きさ，位置と実際の距離についてのデータをとりモデル化した．

シミュレーションモデル

部屋，全方位カメラ，ロボットの大きさと形は，4.1節で述べたパラメータと同じとする．カラーコードの大きさも同じとするが，今シミュレーションでは個体識別する必要がないため簡易化し単色の赤とする．

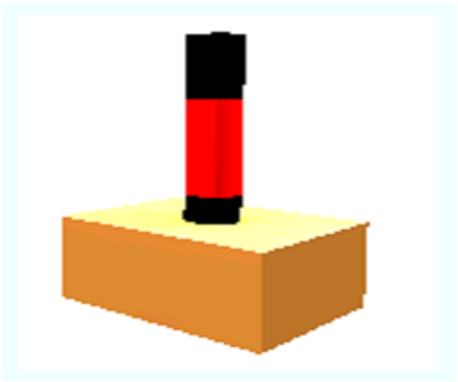

図 5.1: プロトタイプのロボット

シミュレーション内容

パノラマ画像に写るカラーコードの位置データ情報として，カラーコードの上辺のｙ座標 (A), カラーコードの幅 (B)，カラーコードの高さ (C)，カラーコードの下辺のｙ座標 (D) 以上４点の情報についてのデータをとる．

角度の間隔を 0 [°] から 360 [°] まで 15 [°] 間隔とし，各角度について 0.2[m] から 5[m] まで 0.1[m] 間隔で，5[m] から 8[m] まで 0.3[m] 間隔で，8[m] から 10[m] まで 0.5[m] 間隔で測定した．これらのデータについて各角度での平均値の結果を図 5.2 に示す．

横軸が距離，縦軸に写真に写るコードのデータのピクセル数を示す．どれも 2[m] 付近までは大きく変化するがそれ以降の変化は少なく，同じような特性となる．

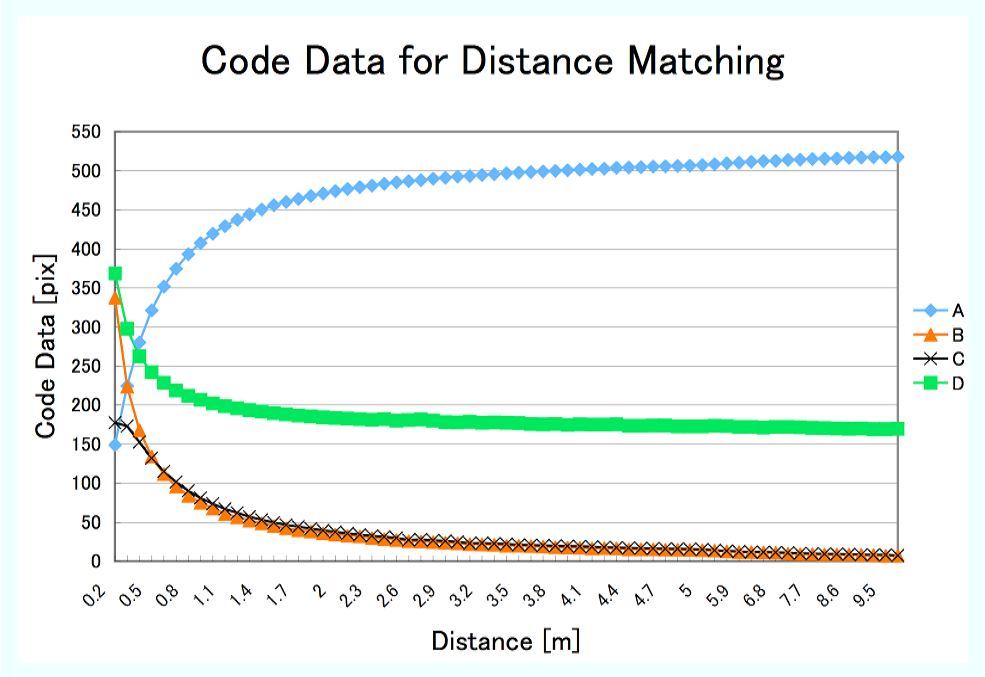

図 5.2: カラーコード情報の結果

5.1.2 マッチングデータのモデル化

一つのデータだけでモデル化するのではなく，複数の特性を組み合わせることによりより良いモデリングが期待できるが，どの曲線の特性はどれもほぼ同じで２つ以上のデータを組み合わせ意味はない．そこで，0.2[m] から 10[m] まですべての区間で一番変化量の多い A の曲線をマッチングデータに使う．

この曲線をモデル化するのに，すべてを一つの近似式であらわすことも可能であり参照する場合一番楽なのであるがエラーの量は多くなる．すべての距離とピクセルのデータを一対一でもてばエラーなくシステムを実現できるが参照量が多すぎて賢い方法とはいえない．近似式の細かさとエラーとの関係はトレードオフの関係がある．本システムでは，各プロット間の直線の式を参照して距離を出すこととする．

5.2 システムの性能評価

5.2.1 角度の性能評価

距離を 1[m] の固定とし，5[°] 間隔で角度を測定した結果を図 5.2 に示す．理想的な環境でのシミュレーションにもかかわらず約 $-0.3sin(2\theta)$ でエラーが発生する．角度はパノラマ変換画像の X 座標に依存される．パノラマ変換で極座標から直交座標に変換するする

ときに，画像の配列には整数しかないので sin の値を四捨五入して変換する．このエラーは，このとき生じるエラーだと考えられる．パノラマ変換をせずにシステムを実装すればエラーをなくすことができると考えられるが，全方位のステレオカメラの処理などはパノラマ変換することが前提となっていたり，システムを拡張する上でパノラマ変化を用いたほうが良いと考え容認誤差とする．

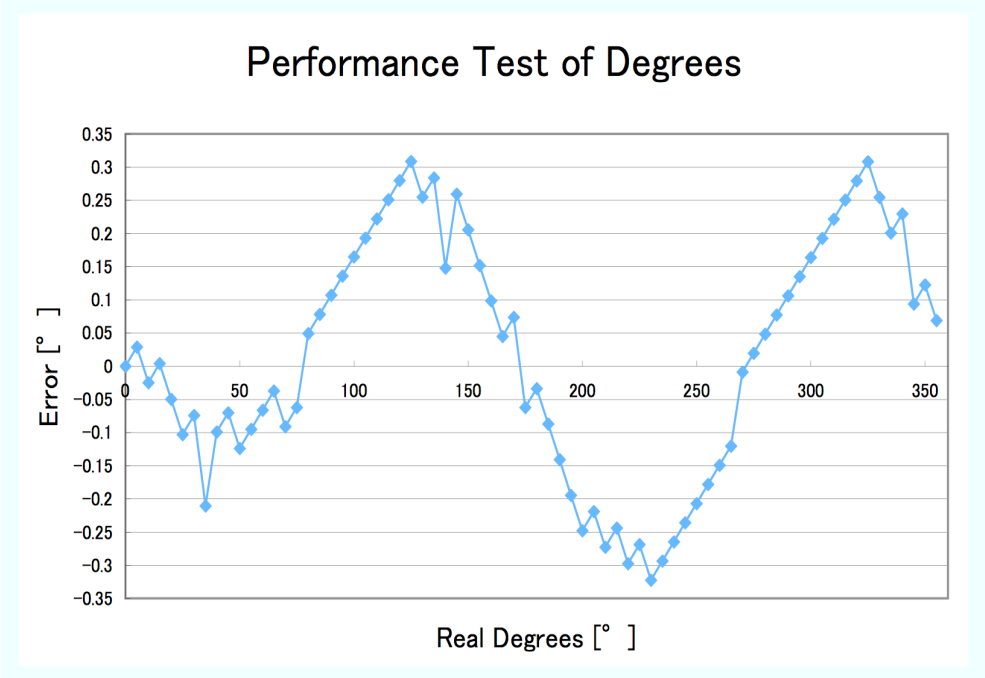

図 5.3: 角度の測定結果

5.2.2 距離の性能評価

0.2[m] から 5[m] まで 0.1[m] 刻み，5[m] から 10[m] まで 0.3[m] 刻みで距離のエラーを 45[°] 間隔で測定した結果を図 5.4 に示す．横軸が距離 [m] で縦軸が測定エラー [m] で，エラーの最大値と平均値のグラフである．

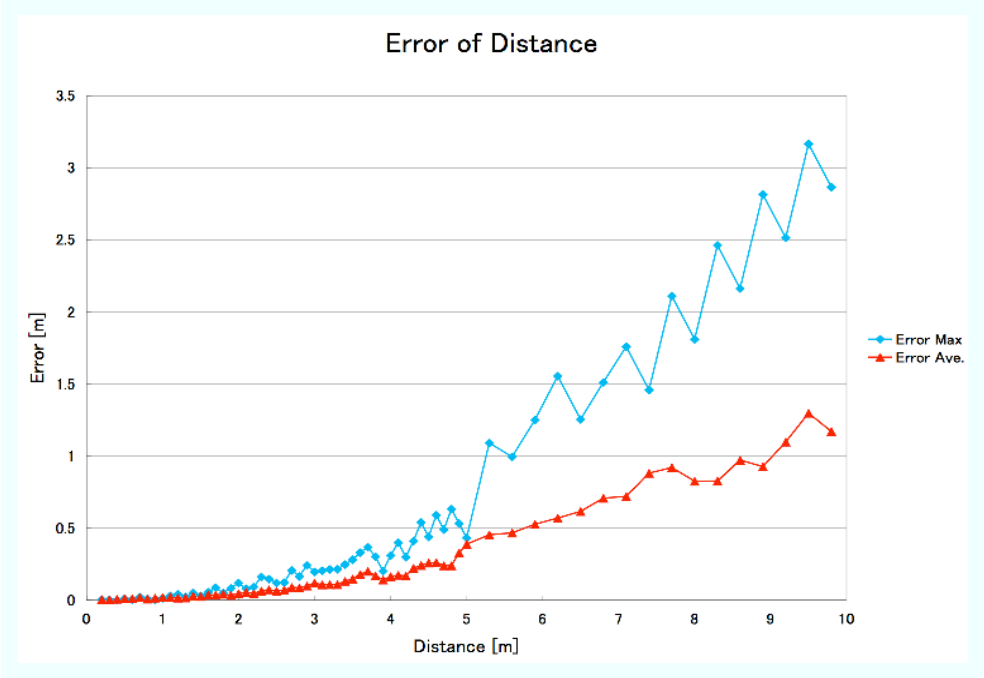

図 5.4: 距離の測定結果 [m]

平均値は，4.8[m] では 0.23[m] の誤差でそれ以降急激に増え 9.5[m] のときに最大で 1.29[m] の誤差がある．5[m] 以降の誤差は 5[m] までのそれと 4 倍の差がある．最大値も同じような特性となり 5[m] 以降急激に誤差が増えそれまでの 5 倍の差がある．また，平均値と最大値の差も 5[m] 以降で多きくなる．これは，距離のマッチングデータで使用したカラーコード情報 A の曲線の特性と A の測定ピクセルとマッチングデータとの誤差が原因であると考えられる．A は 1[m] 間隔ごとで表??のようなピクセルの差がある．0.2[m]-1.0[m] では，A の測定結果がマッチングデータと 1[pix] ずれても 2.96×10^{-3}[m] のエラーとなるが，8[m] 以降では 1[pix] ずれると 1[m] のエラーとなってしまう．

表 5.1: 1[m] 間隔の A の差

距離 [m]	A の差 [pix]
0.2 - 1.0	270
2.0 - 3.0	54
3.0 - 4.0	18
4.0 - 5.0	9
5.0 - 6.0	5
6.0 - 7.0	4
7.0 - 8.0	2
8.0 - 9.0	1
9.0 - 10.0	1

測定誤差を距離との割合の結果は下図のようになる．5[m] までは 5[%] 以内で距離データの値として有効といえるが 5[m] 以降最大で 34[%] の誤差となりロボットの有無の識別には有効であるが距離計測の値としては有効なものとはいえない．

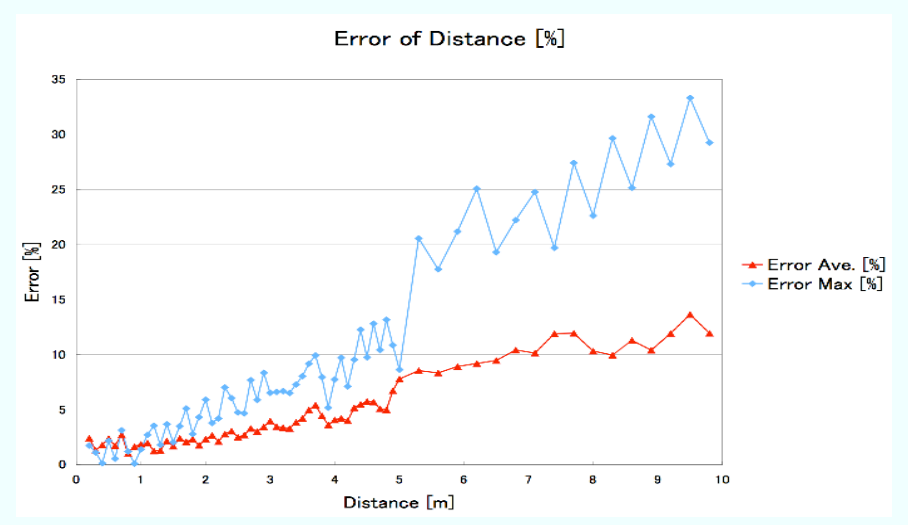

図 5.5: 距離の測定結果

5.2.3 ロボット ID の性能評価

画像からカラーコードを探すとき，ロボット ID がないとコードとみなさないように実装していることと，フィルタリングが問題なくできているので認識可能な距離にあるときには問題なく識別できた．

5.2.4 ロボット台数と処理速度の関係

ロボット台数と処理速度の関係についての結果を図 5.6 に示す．本システムを実装する過程で，オプティマイズのことは考えていないので一回の処理速度としては約 7 秒かかった．まだ，リアルタイムで計測するにはほど遠いがオプティマイズして実装すると数倍のスピードアップは可能であると考えられる．ロボット台数が増加しても，処理速度に大きな変化はなく複数台のロボットを一度に計測するのに適しているシステムであることがわかった．

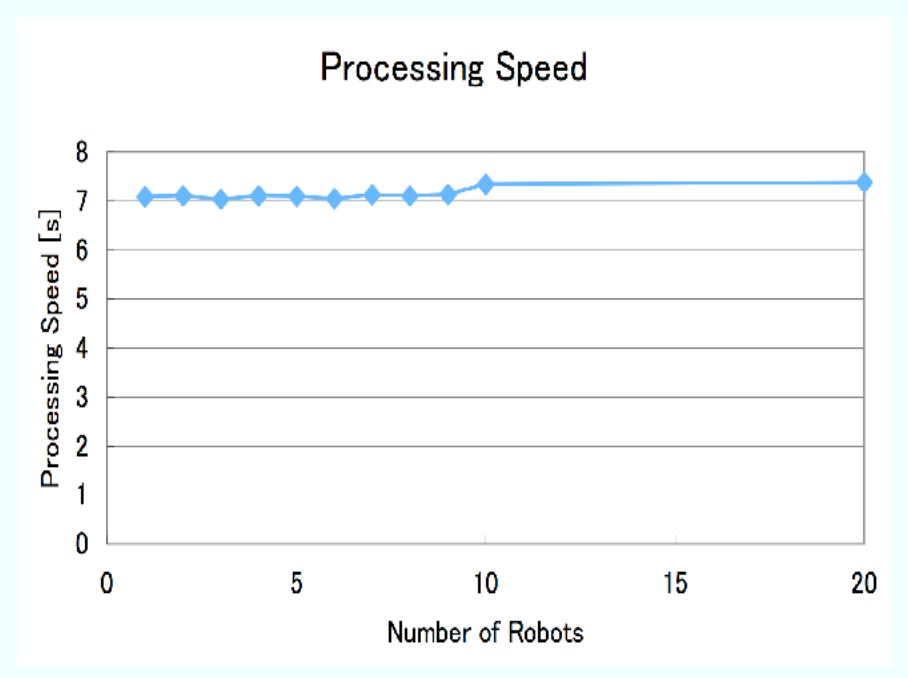

図 5.6: ロボット台数と処理速度

第6章　まとめ

本研究では，移動ロボット群向けの個体識別可能なロボット間位置計測システムを提案，実装しシミュレーションにより性能評価について考察を行った．具体的には，自分自身を基準として他のロボットの位置 (角度と距離) とロボットの ID の 3 点を計測するシステムである．各ロボットに全方位カメラと固有のカラーコードを搭載する．各ロボットは，自分に搭載された全方位カメラで全方位の静止画を撮り込み，カラーコードの特徴を抽出し易くするために前処理の画像処理を行い，取り込んだ静止画からカラーコードの情報以外を排除した画像をつくる．この画像でカラーコードを解析して計測する．本システムに適切なロボット固有のコードとして円柱に一次元の情報を載せるコードを提案した．全方位カメラのパラメータを用いて POV-Ray で理想的なシミュレーション画像の構築を行い，この画像を用いてシステムの性能評価を行った．角度に関しては，パノラマ変換時の sin 関数の四捨五入により ± 0.3 のエラーがでる．これは，画像のピクセルを参照してシステムを実装する上では容認誤差だといえる．距離は，5[m] まではエラーが 5[%] 以内で距離データの値として有効といえるが 5[m] 以降最大で 34[%] の誤差となりロボットの有無の識別には有効であるが距離計測の値としては有効なものとはいえない．ロボット ID に関しては，カラーコードの色情報がすべて読み取れる距離までは確実に識別できた．個体識別と角度がメインで，距離測定はそのおまけとしてシステム考えたほうが良い．測定したい距離，搭載可能なコードの大きさ，カメラの精度を考えてシステムを構築すればロボット間位置情報システムとしては有効であることがわかった．同じ時間の複数台のロボットの位置関係と個体識別を測定するシステムとしては有効であることを示した．

6.1　謝辞

本研究を進めるにあたりご指導頂きました Xavier Defago 先生に深く感謝します．研究以外にも，自己管理や物の考え方について本当に多くのものを学ぶことができたことは，これからの自分の人生に非常にプラスになると思います．また，本研究に進めるにあたり色々と相談にのっていただいた東原大記さんに感謝します．成長でき有意義な 1 年半をすごさせてくれた Xavier 研のみなさんありがとございました．

参考文献

[1] 倉林大輔，長谷川研太 幾何条件による自律移動ロボット群の編隊構造遷移，日本ロボット学会誌, Vol. 23, No. 3, 376/382, 2005.

[2] 山澤一誠, 全方位視覚センサ HyperOmni Vision に関する研究 -移動ロボットのナビゲーションのために-, 大阪大学 博士論文，1997.

[3] 宜保洋平、大矢晃久, 全方位視覚センサとレーザ距離センサによる人間の位置トラッキング, 平成１４年電気学会電子・情報・システム部門大会，pp.578-579，2002.

[4] 鈴木敬弘、大矢晃久、油田信一，投射光を用いた移動ロボットへの動作指示システムの開発 -コンセプトの提案と基礎実験-，日本ロボット学会 第２１回学術講演会，3F22，2003.

[5] 加藤浩二，石黒浩，マシュー バース，全方位視覚センサをもつ複数ロボットの同定と位置決め, 電子情報通信学会論文誌，Vol.J84-D-Ⅱ No.7, pp.1270-1278 2001.

[6] 前田賢一郎，天井光源の幾何学的特長を利用したロバストな自己位置同定手法，奈良先端科学技術大学院大学 修士論文，2004.

[7] 中里 祐介, 神原 誠之, 横矢 直和，不可視マーカを用いたウェアラブル AR システム，画像の認識・理解シンポジウム (MIRU2005) 講演論文集，pp. 1614-1615, July 2005.

[8] 田村秀行，“コンピュータ画像処理”オーム社，2002.

付録

以下に，カラーコードの情報を抽出しそれらの情報を距離，ID，角度情報として出力するところまでのコードを示す.

```
//////////////////////////analyze about color code ///////////////
robotid = new int[7];
robotData = new double[15];
ArrayList robotList = new ArrayList();
////////////////////// look for first red ////////////////////////
for (int i = 0; i < panoramic.length; i ++ ) {
for (int j = 0; j < panoramic[i].length; j++)
////////////////////// height ////////////////////////////////////
if (panoramic[i][j] == 1 && Counter == 0) {
y_0 = i;
x_0 = j;
Counter++;
}

if (panoramic[y_0][x_0] == 1 && Counter > 0) {
int mandeyanen = 0;
while ((temp_x+x_0) < (pi2r-1)) {//mandeyanen == 0

if ((panoramic[y_0][temp_x + x_0]  != 1) || (panoramic[y_0][temp_x + x_0] == 0)) {

mandeyanen++;
break;
}
if (panoramic[y_0][temp_x + x_0] == 1) {
xCounter++;
}//if
temp_x++;
}//while
}

if (panoramic[y_0][x_0 + xCounter/2] == 1 && Counter > 0) {
while ((temp_y+y_0) < radius) {
```

```
if (panoramic[temp_y + y_0][x_0 + (xCounter/2)] == 2 && FirstBlueCounter == 0) {

FirstBlueCounter++;

break;
}
if (panoramic[temp_y + y_0][x_0 + (xCounter/2)] != 2 && FirstBlueCounter == 0) {

yCounterUntillBlue++;
}//if
if (temp_y >= 175 ) {
temp_y = 0;
yCounterUntillBlue = 0;
aoganai++;
break;
}

temp_y++;
}//while
}

if (aoganai > 0) {
while (((temp_y+y_0) < radius) || (FirstBlueCounter < 0)) {

if (panoramic[temp_y + y_0][x_0 + (xCounter/2) + 1] == 2 && FirstBlueCounter == 0) {
FirstBlueCounter++;

break;
}//if
if (panoramic[temp_y + y_0][x_0 + (xCounter/2) + 1] != 2 && FirstBlueCounter == 0) {
yCounterUntillBlue++;
}//if
temp_y++;
}//while
}//if
```

```
if (FirstBlueCounter >0) {
while ((temp_yBlue + yCounterUntillBlue + y_0) < radius) {
if (panoramic[temp_yBlue + yCounterUntillBlue + y_0][x_0 + (xCounter/2)] ==3
                        && FirstyCounter == 0) {
FirstyCounter++;
break;
}
if (panoramic[temp_yBlue + yCounterUntillBlue + y_0][x_0 + (xCounter/2)] != 3
                       && FirstyCounter == 0) {

yCounterBlue++;
}//if
temp_yBlue++;
}//while
}


if (FirstyCounter > 0) {
while(yCounterStart == 0) {
yCounter = yCounterUntillBlue + yCounterBlue;
length_bottom = (int) (radius - (yCounter + y_0));
yCounterStart++;
break;
}//while
}//if

////////////////////////// width( choose max ver.) ////////////////////////////
if (yCounterStart > 0) {
while(widend == 0){

for (int high = y_0+1; high < y_0+yCounter-1; high++){
////////////////// xCounterRight  ///////////////////////////////////////
if (panoramic[high][x_0 + xCounter/2] != 0) {
while ((temp_x_right + x_0 + xCounter/2) < (pi2r-1)) {
if (panoramic[high][temp_x_right + x_0 + xCounter/2] == 0) {
break;
}
```

```
if (panoramic[high][temp_x_right + x_0 + xCounter/2] != 0) {
xCounterRight++;
}//if
temp_x_right++;

}//while
}

////////////// xCounterLeft NOT around 0 degrees ////////////////////////////
if (x_0 > 0) {
while ((x_0 + xCounter/2) - temp_x_left >= 0) {
if (panoramic[high][(x_0 + xCounter/2) - temp_x_left] == 0) {
break;
}
if (panoramic[high][(x_0 + xCounter/2) - temp_x_left] != 0) {
xCounterLeft++;
}//if

if ((x_0 + xCounter/2) - temp_x_left == 1) {
kireteru = x_0;
x_0 = 0;
temp_x_left = 0;
maxofXCL = 0;
break;
}

temp_x_left++;
}//while
}//if

////////////////////////////////xCounterLeft for around 0 degree  ////////////////
if (x_0 == 0) { //panoramic[high][x_0 + xCounter/2] == 1  &&
while (true) {
if (panoramic[high][pi2r - temp_x_left -1] == 0) {
break;
}
if (panoramic[high][pi2r - temp_x_left -1] != 0) {
xCounterLeft++;
```

```
}//if
temp_x_left++;
}//while
}//if

codeWidth = xCounterRight + xCounterLeft -1;
maxofXCR = (int) Math.max(xCounterRight, maxofXCR);
maxofXCL = (int) Math.max(xCounterLeft, maxofXCL);
maxofWidth = (int) Math.max(codeWidth, maxofWidth);
temp_x_right = 0;
temp_x_left = 0;
xCounterRight = 0;
xCounterLeft = 0;

}//for Width

x_3 = (x_0 + xCounter/2) - (maxofXCL -1);
x_4 = x_0 + xCounter/2 + (maxofXCR);

widend++;
break;
}//while
}//if

////////////// theta for NOT 0 degrees ////////////////////////////////////
if (widend > 0 && x_0 > 0) {
while(thetaend == 0) {
codeWidthforTheta = (maxofWidth)/2;
theta = ((double)(x_3 + codeWidthforTheta) * (180/(pi*radius)));
thetaend++;
break;
}//while
}//if

///////// theta for 0 degrees ////////////////////////////////////
if (widend > 0 && x_0 == 0) {
while(thetaend == 0) {
codeWidthforTheta = (maxofXCR + xCounter/2)-((maxofXCL + maxofXCR + xCounter/2)/2);
```

```
theta = (double)(codeWidthforTheta * (180/(pi*radius)));
thetaend++;
break;
}//while
}//if

////////////// Robot ID ////////////////////////
if(thetaend > 0) {
while(idend == 0) {
if (y_0 + ID1 == 653) {
break;
}
if(ID1 == 1) {
while(ID1end==0 && iro1==0) {
//black
                              //under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai] == 3){
robotid[0] = 2;
ID1end++;
iro1 = 3;
break;
}//if
//right under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai + 1] == 3){
robotid[0] = 2;
ID1end++;
iro1 = 3;
break;
}//if
//left under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai -1] == 3){
robotid[0] = 2;
ID1end++;
iro1 = 3;
break;
}//if

//yellow
```

```
                              //under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai] == 4){
robotid[0] = 3;
ID1end++;
iro1 = 4;
break;
}//if
//right under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai + 1] == 4){
robotid[0] = 3;
ID1end++;
iro1 = 4;
break;
}//if
//left under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai -1] == 4){
robotid[0] = 3;
ID1end++;
iro1 = 4;
break;
}//if

//green
                              //under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai] == 5){
robotid[0] = 4;
ID1end++;
iro1 = 5;
break;
}//if
//right under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai + 1] == 5){
robotid[0] = 4;
ID1end++;
iro1 = 5;
break;
}//if
//left under
```

```
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai -1] == 5){
robotid[0] = 4;
ID1end++;
iro1 = 5;
break;
}//if

//cyan
                              //under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai] == 6){
robotid[0] = 5;
ID1end++;
iro1 = 6;
break;
}//if
//right under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai + 1] == 6){
robotid[0] = 5;
ID1end++;
iro1 = 6;
break;
}//if
//left under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai -1] == 6){
robotid[0] = 5;
ID1end++;
iro1 = 6;
break;
}//if

//magenta
                              //under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai] == 7){
robotid[0] = 6;
ID1end++;
iro1 = 7;
break;
}//if
```

```
//right under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai + 1] == 7){
robotid[0] = 6;
ID1end++;
iro1 = 7;
break;
}//if
//left under
if (panoramic[y_0 + ID1][x_0 + (xCounter/2) + aoganai -1] == 7){
robotid[0] = 6;
ID1end++;
iro1 = 7;
break;
}//if

ID1++;
}//while
}//if


if(ID1end > 0) {
while(ID2end==0) {
if (y_0 + ID1 + ID2 == 653) {
break;
}

//red
                            //under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai] == 1 && (iro1!=1)){
robotid[1] = 1;
ID2end++;
iro2 = 1;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai + 1] == 1 && (iro1!=1)){
robotid[1] = 1;
ID2end++;
```

```
iro2 = 1;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai -1] == 1 && (iro1!=1)){
robotid[1] = 1;
ID2end++;
iro2 = 1;
break;
}//if

//black
                              //under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai] == 3  && (iro1!=3)){
robotid[1] = 2;
ID2end++;
iro2 = 3;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai + 1] == 3 && (iro1!=3)){
robotid[1] = 2;
ID2end++;
iro2 = 3;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai -1] == 3 && (iro1!=3)){
robotid[1] = 2;
ID2end++;
iro2 = 3;
break;
}//if

//yellow
                              //under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai] == 4 && (iro1!=4)){
robotid[1] = 3;
```

```
ID2end++;
iro2 = 4;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai + 1] == 4 && (iro1!=4)){
robotid[1] = 3;
ID2end++;
iro2 = 4;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai -1] == 4 && (iro1!=4)){
robotid[1] = 3;
ID2end++;
iro2 = 4;
break;
}//if

//green
                            //under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai] == 5 && (iro1!=5)){
robotid[1] = 4;
ID2end++;
iro2 = 5;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai + 1] == 5 && (iro1!=5)){
robotid[1] = 4;
ID2end++;
iro2 = 5;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai -1] == 5 && (iro1!=5)){
robotid[1] = 4;
ID2end++;
```

```
iro2 = 5;
break;
}//if

//cyan
                              //under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai] == 6 && (iro1!=6)){
robotid[1] = 5;
ID2end++;
iro2 = 6;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai + 1] == 6 && (iro1!=6)){
robotid[1] = 5;
ID2end++;
iro2 = 6;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai -1] == 6 && (iro1!=6)){
robotid[1] = 5;
ID2end++;
iro2 = 6;
break;
}//if

//magenta
                              //under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai] == 7 && (iro1!=7)){
robotid[1] = 6;
ID2end++;
iro2 = 7;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai + 1] == 7 && (iro1!=7)){
robotid[1] = 6;
```

```
ID2end++;
iro2 = 7;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2][x_0 + (xCounter/2) + aoganai -1] == 7 && (iro1!=7)){
robotid[1] = 6;
ID2end++;
iro2 = 7;
break;
}//if

ID2++;
}//while
}//if


if(ID2end > 0) {
while(ID3end==0) {
//System.out.println("ID3");
if (y_0 + ID1 + ID2 + ID3 == 653) {
break;
}

//red
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai] == 1
&&(iro2!=1)){
robotid[2] = 1;
ID3end++;
iro3 = 1;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai + 1] == 1
&&(iro2!=1)){
robotid[2] = 1;
ID3end++;
```

```
iro3 = 1;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai -1] == 1
&&(iro2!=1)){
robotid[2] = 1;
ID3end++;
iro3 = 1;
break;
}//if

//black
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai] == 3
&&(iro2!=3)){
robotid[2] = 2;
ID3end++;
iro3 = 3;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai + 1] == 3
&&(iro2!=3)){
robotid[2] = 2;
ID3end++;
iro3 = 3;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai -1] == 3
&&(iro2!=3)){
robotid[2] = 2;
ID3end++;
iro3 = 3;
break;
}//if
```

```
//yellow
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai] == 4
&&(iro2!=4)){
robotid[2] = 3;
ID3end++;
iro3 = 4;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai + 1] == 4
&&(iro2!=4)){
robotid[2] = 3;
ID3end++;
iro3 = 4;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai -1] == 4
&&(iro2!=4)){
robotid[2] = 3;
ID3end++;
iro3 = 4;
break;
}//if

//green
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai] == 5
&&(iro2!=5)){
robotid[2] = 4;
ID3end++;
iro3 = 5;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai + 1] == 5
&&(iro2!=5)){
```

```
robotid[2] = 4;
ID3end++;
iro3 = 5;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai -1] == 5
&&(iro2!=5)){
robotid[2] = 4;
ID3end++;
iro3 = 5;
break;
}//if

//cyan
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai] == 6
&&(iro2!=6)){
robotid[2] = 5;
ID3end++;
iro3 = 6;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai + 1] == 6
&& (iro2!=6)){
robotid[2] = 5;
ID3end++;
iro3 = 6;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai -1] == 6
&& (iro2!=6)){
robotid[2] = 5;
ID3end++;
iro3 = 6;
break;
```

```
}//if

//magenta
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai] == 7
&& (iro2!=7)){
robotid[2] = 6;
ID3end++;
iro3 = 7;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai + 1] == 7
&& (iro2!=7)){
robotid[2] = 6;
ID3end++;
iro3 = 7;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3][x_0 + (xCounter/2) + aoganai -1] == 7
&& (iro2!=7)){
robotid[2] = 6;
ID3end++;
iro3 = 7;
break;
}//if

ID3++;
}//while
}//if

if(ID3end > 0) {
while(ID4end ==0) {
if (y_0 + ID1 + ID2 + ID3 + ID4 == 653) {
break;
}
//red
```

```
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai] == 1
&& (iro3!=1)){
System.out.println("unko");
robotid[3] = 1;
ID4end++;
iro4 = 1;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai + 1] == 1
&& (iro3!=1)){
robotid[3] = 1;
ID4end++;
iro4 = 1;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai -1] == 1
&& (iro3!=1)){
robotid[3] = 1;
ID4end++;
iro4 = 1;
break;
}//if

//black
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai] == 3
&& (iro3!=3)){
robotid[3] = 2;
ID4end++;
iro4 = 3;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai + 1] == 3
&& (iro3!=3)){
```

```
robotid[3] = 2;
ID4end++;
iro4 = 3;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai -1] == 3
&& (iro3!=3)){
robotid[3] = 2;
ID4end++;
iro4 = 3;
break;
}//if

//yellow
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai] == 4
&& (iro3!=4)){
robotid[3] = 3;
ID4end++;
iro4 = 4;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai + 1] == 4
&& (iro3!=4)){
robotid[3] = 3;
ID4end++;
iro4 = 4;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai -1] == 4
&& (iro3!=4)){
robotid[3] = 3;
ID4end++;
iro4 = 4;
break;
```

```
}//if

//green
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai] == 5
&& (iro3!=5)){
robotid[3] = 4;
ID4end++;
iro4 = 5;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai + 1] == 5
&& (iro3!=5)){
robotid[3] = 4;
ID4end++;
iro4 = 5;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai -1] == 5
&& (iro3!=5)){
robotid[3] = 4;
ID4end++;
iro4 = 5;
break;
}//if

//cyan
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai] == 6
&& (iro3!=6)){
robotid[3] = 5;
ID4end++;
iro4 = 6;
break;
}//if
//right under
```

```
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai + 1] == 6
&& (iro3!=6)){
robotid[3] = 5;
ID4end++;
iro4 = 6;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai -1] == 6
&& (iro3!=6)){
robotid[3] = 5;
ID4end++;
iro4 = 6;
break;
}//if

//magenta
                            //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai] == 7
&& (iro3!=7)){
robotid[3] = 6;
ID4end++;
iro4 = 7;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai + 1] == 7
&& (iro3!=7)){
robotid[3] = 6;
ID4end++;
iro4 = 7;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4][x_0 + (xCounter/2) + aoganai -1] == 7
&& (iro3!=7)){
robotid[3] = 6;
ID4end++;
```

```
iro4 = 7;
break;
}//if

ID4++;
}//while
}//else if

if(ID4end > 0) {
while(ID5end == 0) {
if (y_0 + ID1 + ID2 + ID3 + ID4 + ID5== 653) {
break;
}
//red
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai] == 1
&& (iro4!=1)){
robotid[4] = 0;
ID5end++;
iro5 = 1;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai + 1]==1
&& (iro4!=1)){
robotid[4] = 0;
ID5end++;
iro5 = 1;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai -1]==1
&& (iro4!=1)){
robotid[4] = 0;
ID5end++;
iro5 = 1;
break;
}//if
```

```
//black
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai]==3
&& (iro4!=3)){
robotid[4] = 2;
ID5end++;
iro5 = 3;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai + 1]==3
&& (iro4!=3)){
robotid[4] = 2;
ID5end++;
iro5 = 3;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai -1]==3
&& (iro4!=3)){
robotid[4] = 2;
ID5end++;
iro5 = 3;
break;
}//if

//yellow
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai]==4
&& (iro4!=4)){
robotid[4] = 3;
ID5end++;
iro5 = 4;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai + 1]==4
```

```
&& (iro4!=4)){
robotid[4] = 3;
ID5end++;
iro5 = 4;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai -1]==4
&& (iro4!=4)){
robotid[4] = 3;
ID5end++;
iro5 = 4;
break;
}//if

//green
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai]==5
&& (iro4!=5)){
robotid[4] = 4;
ID5end++;
iro5 = 5;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai + 1]==5
&& (iro4!=5)){
robotid[4] = 4;
ID5end++;
iro5 = 5;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai -1]==5
&& (iro4!=5)){
robotid[4] = 4;
ID5end++;
iro5 = 5;
```

```
break;
}//if

//cyan
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai]==6
&& (iro4!=6)){
robotid[4] = 5;
ID5end++;
iro5 = 6;
break;
}//if
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai + 1]==6
&& (iro4!=6)){
robotid[4] = 5;
ID5end++;
iro5 = 6;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai -1]==6
&& (iro4!=6)){
robotid[4] = 5;
ID5end++;
iro5 = 6;
break;
}//if

//magenta
                              //under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai]==7
&& (iro4!=7)){
robotid[4] = 6;
ID5end++;
iro5 = 7;
break;
}//if
```

```
//right under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai + 1]==7
&& (iro4!=7)){
robotid[4] = 6;
ID5end++;
iro5 = 7;
break;
}//if
//left under
if (panoramic[y_0 + ID1 + ID2 + ID3 + ID4 + ID5][x_0 + (xCounter/2) + aoganai -1]==7
&& (iro4!=7)){
robotid[4] = 6;
ID5end++;
iro5 = 7;
break;
}//if

ID5++;
}//while
}//else if

robotID = robotid[0]*10000 + robotid[1]*1000 + robotid[2]*100 + robotid[3]*10
+ robotid[4];

idend++;
break;
}//while
}//if

////////////// corner point for NOT around 0 degrees ///////////////////
if (idend > 0 && x_0 > 0){
while(cornerpointend == 0) {
// point a
x_a = x_3; // - 10
y_a = y_0; //- 10
//point b
x_b = x_3 + maxofWidth; // + 10
y_b = y_a;
```

```
// point c
x_c = x_3; // - 10
y_c = y_0 + (3*yCounter)/2;
// point d
x_d = x_b;
y_d = y_c;
cornerpointend++;
break;
}//while
}//if

///////////////// corner point for around 0 degrees ////////
if (idend > 0 && x_0 == 0){
while(cornerpointend == 0) {
// point a
x_a = pi2r-1 - maxofXCL ;
y_a = y_0;
//point b
x_b = kireteru+maxofXCR+xCounter/2;
y_b = y_a;
// point c
x_c = x_a;
y_c = y_0 + (3*yCounter)/2;
// point d
x_d = x_b;
y_d = y_c;

cornerpointend++;
break;
}//while
}//if


////////////// Distance ////////////////////////////////////////
if (cornerpointend > 0) {
while (distanceend == 0) {
///////////////// Distance matching final version ///////////////////
```

```
// 0.2-0.3
if (y_0 < 224) {
dis_matching_last = 0.0013*y_0 + 0.0015;
}//if

// 0.3-0.4
if (y_0 >= 224 && y_0 < 280) {
dis_matching_last = 0.0018*y_0 - 0.1017;
}//if

// 0.4-0.5
if (y_0 >= 280 && y_0 < 321) {
dis_matching_last = 0.0024*y_0 - 0.2834;
}//if

// 0.5-0.6
if (y_0 >= 321 && y_0 < 351) {
dis_matching_last = 0.0033*y_0 - 0.5518;
}//if

// 0.6-0.7
if (y_0 >= 351 && y_0 < 374) {
dis_matching_last = 0.0043*y_0 - 0.9258;
}//if

// 0.7-0.8
if (y_0 >= 374 && y_0 < 393) {
dis_matching_last = 0.0055*y_0 - 1.3463;
}//if

// 0.8-0.9
if (y_0 >= 393 && y_0 < 408) {
dis_matching_last = 0.0068*y_0 - 1.8867;
}//if

// 0.9-1.0
if (y_0 >= 408 && y_0 < 420) {
dis_matching_last = 0.0084*y_0 - 2.5431;
```

```
}//if

// 1.0-1.1
if (y_0 >= 420 && y_0 < 429) {
dis_matching_last = 0.0103*y_0 - 3.3326;
}//if

// 1.1-1.2
if (y_0 >= 429 && y_0 < 437) {
dis_matching_last = 0.0122*y_0 - 4.1282;
}//if

// 1.2-1.3
if (y_0 >= 437 && y_0 < 444) {
dis_matching_last = 0.0143*y_0 - 5.0496;
}//if

// 1.3-1.4
if (y_0 >= 444 && y_0 < 450) {
dis_matching_last = 0.0162*y_0 - 5.9179;
}//if

// 1.4-1.5
if (y_0 >= 450 && y_0 < 455) {
dis_matching_last = 0.0194*y_0 - 7.3367;
}//if

// 1.5-1.6
if (y_0 >= 455 && y_0 < 460) {
dis_matching_last = 0.0218*y_0 - 8.454;
}//if

// 1.6-1.7
if (y_0 >= 460 && y_0 < 464) {
dis_matching_last = 0.026*y_0 - 10.382;
}//if

// 1.7-1.8
```

```
if (y_0 >= 464 && y_0 < 467) {
dis_matching_last = 0.0271*y_0 - 10.9;
}//if

// 1.8-1.9
if (y_0 >= 467 && y_0 < 471) {
dis_matching_last = 0.0317*y_0 - 13.017;
}//if

// 1.9-2
if (y_0 >= 471 && y_0 < 474) {
dis_matching_last = 0.0339*y_0 - 14.082;
}//if

// 2.0-2.1
if (y_0 >= 474 && y_0 < 476) {
dis_matching_last = 0.0365*y_0 - 15.319;
}//if

// 2.1-2.2
if (y_0 >= 476 && y_0 < 479) {
dis_matching_last = 0.0442*y_0 - 18.965;
}//if

// 2.2-2.3
if (y_0 >= 479 && y_0 < 481) {
dis_matching_last = 0.0463*y_0 - 19.998;
}//if

// 2.3-2.4
if (y_0 >= 481 && y_0 < 483) {
dis_matching_last = 0.0475*y_0 - 20.555;
}//if

// 2.4-2.5
if (y_0 >= 483 && y_0 < 485) {
dis_matching_last = 0.0514*y_0 - 22.416;
}//if
```

```
// 2.5-2.6
if (y_0 >= 485 && y_0 < 486) {
dis_matching_last = 0.0594*y_0 - 26.309;
}//if

// 2.6-2.7
if (y_0 >= 486 && y_0 < 488) {
dis_matching_last = 0.0731*y_0 - 32.981;
}//if

// 2.7-2.8
if (y_0 >= 488 && y_0 < 489) {
dis_matching_last = 0.0613*y_0 - 27.226;
}//if

// 2.8-2.9
if (y_0 >= 489 && y_0 < 491) {
dis_matching_last = 0.0826*y_0 - 37.67;
}//if

// 2.9-3.0
if (y_0 >= 491 && y_0 < 492) {
dis_matching_last = 0.0704*y_0 - 31.659;
}//if

// 3.0-3.3
if (y_0 >= 492 && y_0 < 495) {
dis_matching_last = 0.0888*y_0 - 40.725;
}//if

// 3.3-3.6
if (y_0 >= 495 && y_0 < 498) {
dis_matching_last = 0.1091*y_0 - 50.809;
}//if

// 3.6-4.0
if (y_0 >= 498 && y_0 < 501) {
```

```
dis_matching_last = 0.1328*y_0 - 62.619;
}//if

// 4.0-4.3
if (y_0 >= 501 && y_0 < 503) {
dis_matching_last = 0.1372*y_0 - 64.807;
}//if

// 4.3-4.6
if (y_0 >= 503 && y_0 < 505) {
dis_matching_last = 0.2074*y_0 - 100.22;
}//if

// 4.6-5
if (y_0 >= 505 && y_0 < 507) {
dis_matching_last = 0.2117*y_0 - 102.41;
}//if

// 5.0-5.6
if (y_0 >= 507 && y_0 < 509) {
dis_matching_last = 0.2499*y_0 - 121.76;
}//if

// 5.6-6.2
if (y_0 >= 509 && y_0 < 511) {
dis_matching_last = 0.2991*y_0 - 146.81;
}//if

// 6.2-7.1
if (y_0 >= 511 && y_0 < 513) {
dis_matching_last = 0.4025*y_0 - 199.69;
}//if

// 7.1-8
if (y_0 >= 513 && y_0 < 515) {
dis_matching_last = 0.5557*y_0 - 278.48;
}//if
```

```
// 8-8.9
if (y_0 >= 515 && y_0 < 517) {
dis_matching_last = 0.5563*y_0 - 278.74;
}//if

// 8.9-9.8
if (y_0 >= 517) {
dis_matching_last = 0.9517*y_0 - 483.17;
}//if

distanceend++;
break;
}
}


////////////////////////////// RobotData //////////////////////
if (distanceend > 0) {
while (robotdataend == 0) {
robotData [0] = robotID;
robotData [1] = theta;
robotData [2] = dis_matching_last;
robotData [3] = yCounter;
robotData [4] = maxofWidth;
robotData [5] = y_0;
robotData [6] = length_bottom;
robotData [7] = x_a;
robotData [8] = x_b;
robotData [9] = x_c;
robotData [10] = x_d;
robotData [11] = y_a;
robotData [12] = y_b;
robotData [13] = y_c;
robotData [14] = y_d;
robotdataend++;

break;
}
```

```
}


////////////////// RobotoList ////////////////////////////
if (robotdataend > 0) {
while (robotlistend == 0) {

robotList.add(robotData);
//System.out.println("robotData");
robotlistend++;
// exchangepanoramic++;
break;
}
}

////////////// exchange of panoramic[][] data for NOT around 0 degrees //////////////
if (robotlistend > 0 && x_0 > 0) {
while (exchangepanoramic == 0) {

for (int epi = y_a; epi < y_c; epi ++ ) {
       for (int epj = x_a; epj < x_b; epj++)
       panoramic[epi][epj] = 0;

       }

exchangepanoramic++;
break;
}
}


/////////////// exchange of panoramic[][] data for around 0 degrees ///////////////
if (robotlistend > 0 && x_0 == 0) {
while (exchangepanoramic == 0) {

for (int epir = y_a; epir < y_c; epir ++ ) {
for (int epjr = x_a; epjr <  pi2r; epjr++)
panoramic[epir][epjr] = 0;
```

```
}

for (int epil = y_b; epil < y_d; epil ++ ) {
for (int epjl = 0; epjl < x_b; epjl++)
panoramic[epil][epjl] = 0;
}

exchangepanoramic++;
break;
}
}

////////////// initialyze parameters ////////////////////////////////
if (exchangepanoramic > 0) {
while (initend == 0) {

dis_matching_last = 0;
y_0 = 0;
x_0 = 0;
x_a = 0;
x_b = 0;
x_c = 0;
x_d = 0;
y_a = 0;
y_b = 0;
y_c = 0;
y_d = 0;
Counter = 0;
xCounter = 0;
temp_x = 0;
FirstBlueCounter = 0;
yCounterUntillBlue = 0;
temp_y = 0;
FirstyCounter = 0;
yCounterBlue = 0;
temp_yBlue = 0;
yCounterStart = 0;
length_bottom = 0;
```

```
yCounter = 0;
widend = 0;
temp_x_right = 0;
xCounterRight = 0;
xCounterRight = 0;
xCounterLeft = 0;
temp_x_left = 0;
xCounterLeft = 0;
temp_x_left = 0;
codeWidth = 0;
maxofXCR = 0;
maxofXCL = 0;
maxofWidth = 0;
thetaend = 0;
codeWidthforTheta = 0;
theta = 0;
idend = 0;
ID1 = 1;
ID1end = 0;
ID2end = 0;
ID2 = 1;
ID3end = 0;
ID3 = 1;
ID4end = 0;
ID4 = 1;
ID5end = 0;
ID5 = 1;
ID6end = 0;
ID6 = 1;
ID7end = 0;
ID7 = 1;
robotID = 0;
cornerpointend = 0;
distanceend = 0;
robotdataend = 0;
robotlistend = 0;
exchangepanoramic = 0;
iro1 = 0;
```

```
iro2 = 0;
iro3 = 0;
iro4 = 0;
iro5 = 0;


////////////// print out result ////////////////////////////////
double test0[];
test0 = new double[15];
test0 = (double[]) robotList.get(robotList.size()-1);

System.out.println("//// " + "Robot " + robotList.size() + "//// ");
System.out.println("Color code and Robot data as follows,");
System.out.println("Length = " + (int)(test0[5]));
System.out.println("Height = " + (int)(test0[3]));
System.out.println("Width = " + (int)(test0[4]));
System.out.println("theta = " + test0[1] + " [degrees]");
System.out.println("Robot ID = " + (int)(test0[0]));
System.out.println("Distance = " + test0[2] + " [m]");
System.out.println("////////////////////////////////////// ");
////////////////////////////////////////////////////////////////////

initend++;
break;
}//while
initend = 0;
}//if
```