計測自動制御学会東北支部 第231回研究集会 (2006.10.31)
資料番号 231-4

視覚センサを用いた自律移動ロボットの行動決定

Decision making for an autonomous mobile robot using a vision sensor

○山口智充*，及川一美*, 大久保重範*, 高橋達也*

Tomomitsu Yamaguchi*, Kazumi Oikawa*, Sigenori Okubo*, Tatsuya Takahashi*

*山形大学

*Yamagata University

キーワード： 自律移動(autnomous mobile robot)，画像処理(image processing), 視覚センサ(vision sensor)

連絡先： 〒992-8510　山形県米沢市城南4-3-16 山形大学 工学部 機械システム工学科 大久保研究室
大久保重範，Tel.: (0238)26-3245, E-mail: sokubo@yz.yamagata-u.ac.jp

1. 緒言

本研究ではカメラから画像を取り込み,ホストコンピュータで画像処理を行う事で,黄色いボールを抜き出し, そのデータをロボットに転送することでロボットにボールを追従,把持させる研究を行っている。しかし,コストダウンが重視されている現在では,PCのスペックは安価なことが重要である。そのため本研究では,比較的性能の低いPCでも動作する自律移動ロボットの開発を目的としている。PCの性能が低い場合,画像処理の処理速度が早いことはロボットを円滑に動かすために必要である。そこで今回は4近傍ラベリング処理,ラインブロックを用いたラベリング処理,ヒストグラムを用いた推定位置算出法の3つの処理速度の検定を行い,今後用いていく手法の選定を行う。

2. 仕様

使用したPC,およびカメラの概略を,Table1,Table2に示す。

Table 1 : Host computer

CPU	Duron 750MHz
OS	Linux kernel 2.4.31-Ovl1.8
コンパイラ	gcc version 3.32

Table 2 : Camera

社名	クリエイティブメディア株式会社
型番	Video BLASTER WEBCAM Plus
解像度	320×240
焦点距離	15cm〜∞

ホストコンピュータとカメラはUSBにより接続し,カメラより得られた画像はホストコンピュータを用いて画像処理を行い,目標物の重心位置計算などを行うことができる。今回の目標物は黄色いボールを対象としている。

3. 画像処理の流れ

画像処理はFig.1に示すように,画像取得,2値化,ラベリング,重心計算の処理を行う。

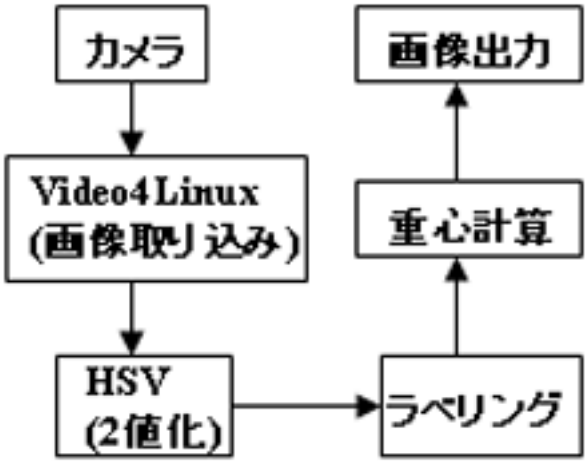

Fig. 1　Flow chart of image processing

4. ラベリング

ラベリング処理とは,2値化された画像内の連結している画素に同じラベル(札)を付与することで複数の領域をグループとして分類する処理のことである。各グループに貼られた「0,1,2…」というラベルにより目的とするオブジェクトを識別し,抽出することができる。ラベリング処理にはいくつかの方法があり,今回は4近傍ラベリング手法,ラインブロックを用いたラベリング手法, ヒストグラムを利用した連結成分検出法の3つをあげ,処理速度の比較を行っていく。

4.1 4近傍ラベリング

4近傍ラベリングは左上から右下に向けて画像をラスター走査(左上のマス目から次々とその右隣のマス目を見ていく走査)を行って,注目画素の隣接画素を調べることで,逐次的に画素の塊を調べていく手法である。2値画像をf,横方向画素数をm,縦方向画素数をnとし,一番左上の画素をf(1,1)とする。Fig.2のように注目画素が値1(目標物の色)を持つとき, 注目画素の4つの隣接画素f(m-1,n),f(m-1,n-1),f(m,n-1),f(m+1,n-1)を検索し,ラベルの貼られた画素が存在しないときは,新しいラベルをつける。ラベルの貼られた画素が見つかった時は,連結している画素と見なし,同じラベルをつける。Fig.3のように隣接関係にある画素が異なるラベルを付与された場合(ラベルの衝突),同じ塊としてラベルを書き換える。またこの時,ラベルの付け替えは補助テーブルを用いて処理速度を向上している。最終的に,最大面積を持つラベルの貼られた塊を抜き出し,重心を求める事でボールの重心を求めることができる。

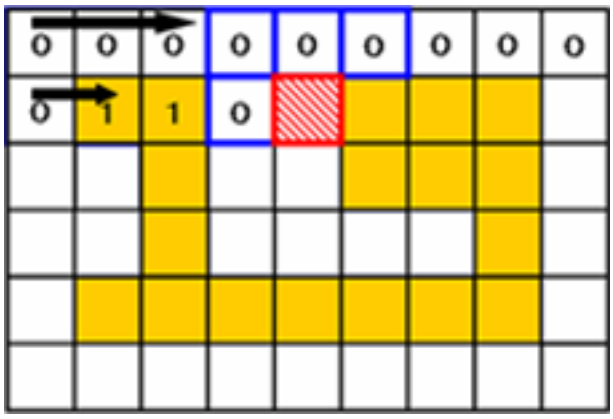

Fig. 2　Labeling 1

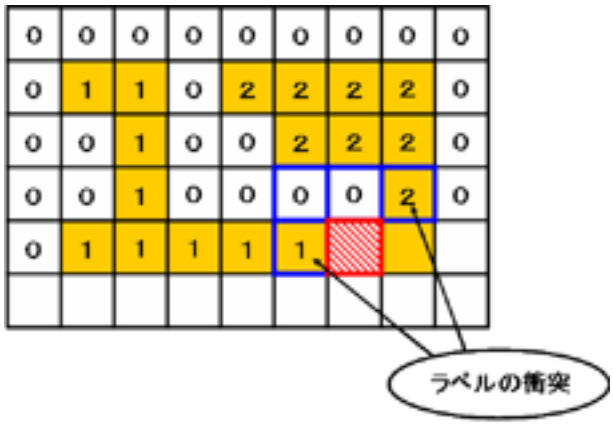

Fig. 3　Collision of label

4.2 ラインブロックを用いたラベリング手法

この手法は横の画素の塊をブロックとして捉え,上下のブロックの最大値,最小値の位置関係で繋がりを調べる手法である。

まず,ラインブロックの作成方法について述べる。2値画像をf,横方向画素数をm,縦方向画素数をnとし,4近傍と同様に左上の画素f(1,1)からラスタ走査によって1画素を探す。Fig.4のように注目画素f(m,n)に1を発見した時,f(m-1,n)を調べ0ならf(m,n)=minとする。また注目画素が0のときはf(m-1,n)を調べ1ならf(m-1,n)=maxとしてラインブロックを作成していく。

次に上下のブロックの繋がりの判定法を述べる。注目ブロックの上にブロックが存在し,Fig.5のようにmin1 < max2かつmax1 > min2の時,この二つのブロックは連結画素であるとしてラベルをつけていく。注目ブロックの上に連結していると判定されるブロックが二つある場合(ラベルの衝突)4近傍と同様に補助テーブルを用いてラベルを付け替えている。

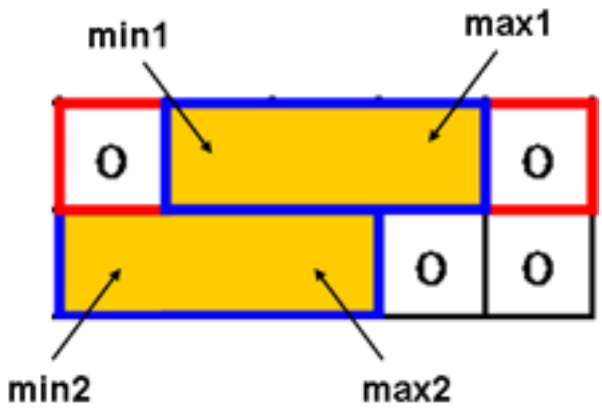

Fig. 4 Lineblock Labeling 1

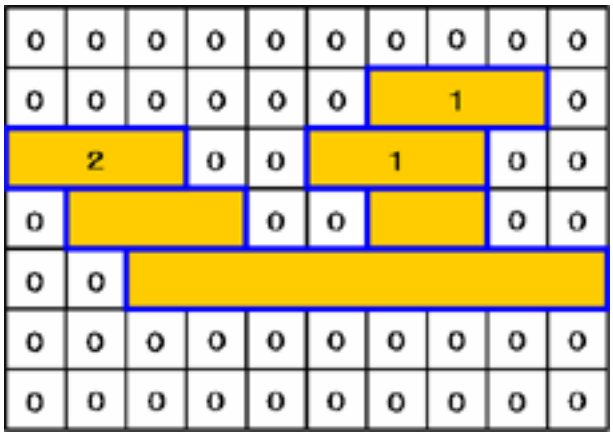

Fig. 5 Lineblock Labeling 2

4.3 ヒストグラムを用いた連結成分検出法

この手法は2値化後の,黄色の部分だけが抜き出された画像に対して,横方向に黄色の画素がどのくらい含まれているかを調べる。2値画像をf,横方向画素数をm,縦方向画素数をnとする。まず横方向に1,2,3 … nと1行ごとに含まれる黄色の画素がいくつ含まれているかを走査し,Fig.6のようなヒストグラムを作る。縦方向にも同様の走査を行い, 最後にそれぞれの最大画素数の座標を目標物の推定位置と見なす手法である。しかし,この手法は最大画素数を読み取っているだけなので,Fig.7に示すようにノイズに影響されやすいという欠点がある。

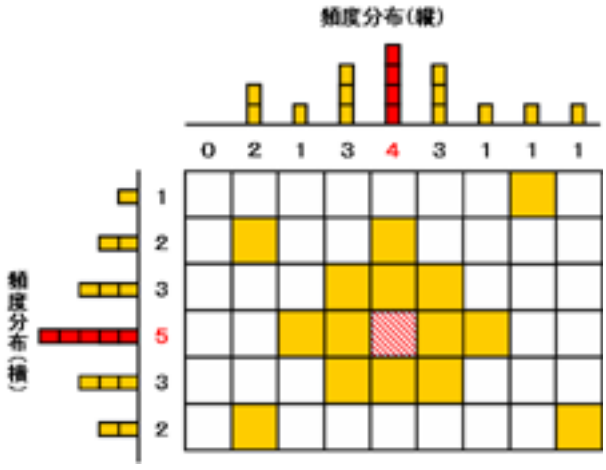

Fig. 6 Histogram 1

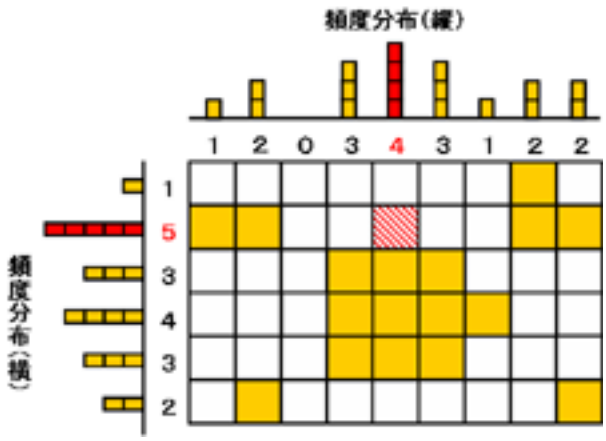

Fig. 7　Histogram 2

5.　処理時間の検定

5.1　サンプル

今回はPCにかかる負荷も考慮し,AM11時,PM1時,PM3時,PM5時,PM7時の5回毎に処理速度を計測しデータを取得した。これはホストコンピュータに用いているPCは本研究室内でサーバーとして用いられており,サーバーの利用が集中する(負荷のかかる)11時～の間で処理速度が変わるのではないかということ。また,日光の量によって取得した画面内の,閾値内に入る画素の量が変化することで,ラベリングを行う量に変化が起こり, 処理速度に変化が現れるのではないかということ,以上二つの仮説から,日没後7時まで2時間ごとにデータを取得することにした。また,サンプルは50個ずつ取り,同時間帯のデータ同士で検定を行うことで有意差を求める。計測結果をグラフにしたものを裏面に示す。また,グラフは横軸に回数,縦軸に処理速度を取っている。

5.2　検定結果

4近傍ラベリング手法をA,ラインブロックラベリング手法をB,ヒストグラムを用いた連結成分検出法をCとし, それぞれの検定結果をTable 3に示す。

Table 3 : Results

1回目	平均[msec]	標準偏差
A	18.6	0.756
B	9.76	0.476
C	8.86	0.351

2回目	平均[msec]	標準偏差
A	18.6	0.884
B	9.66	0.479
C	9.10	0.365

3回目	平均[msec]	標準偏差
A	19.2	1.81
B	9.88	0.746
C	8.86	0.640

4回目	平均[msec]	標準偏差
A	19.1	1.11
B	10.0	0.808
C	9.14	0.405

5回目	平均[msec]	標準偏差
A	18.1	0.544
B	9.56	1.59
C	9.06	0.314

5.3　考察

t-検定の結果,ヒストグラムを用いた手法とラインブロックラベリング手法は4近傍ラベリング手法に比べて平均値に有意な差があることが証明された。また,5回の計測に渡って,処理速度の早さの順位に変化はなく,4近傍ラベリング手法,ラインブロックラベリング手法, ヒストグラムを用いた手法の順に処理速度が早くなっている。このことから時間帯による処理速度の大きな変化は見られなかった。しかしヒストグラムを用いた手法は先に述べたように,ノイズに影響されやすい欠点があることから, ラインブロックが優れていると考えられるため,ラインブロックを用いたラベリング手法を今後採用していくことにする。

6. 結言

今回は3つの推定位置算出法の検定を行うことで効率の良い画像処理方法を求めることができた。今後はカメラの画像から,カメラから見たボールの位置を求め, そこにロボットを追従させるプログラムの改良を行いたいと思っている。現在ロボットはボールがカメラから斜めの方向にあるとき,ボールが画面の真ん中に来るように旋回し,画面の中央に来た時に前進を行い,中央からそれると,また中央に来るように旋回を行い,前進旋回を繰り返してボールに向かうプログラムになっているので,曲線を描く軌跡で一度にボールに向かえるようにしたい。またボールを見失った時の動作が,旋回によってボールを探すようになっており効率が悪いので,ロボットに乗せられたカメラ自体を回転させることも試みたい。

参考文献

1) 井上誠喜:C言語で学ぶ実践画像処理,オーム社雑誌局(1999)

2) 飯尾淳:Linuxによる画像処理プログラミング,オーム社雑誌局(2000)

3) 酒井幸市:デジタル画像処理入門,CQ出版社(2002)

4) 内田智史:C言語によるプログラミング,オーム社雑誌局(2001)

5) 藏前智映:視覚センサを用いた自律移動ロボットの開発,2006年度山形大学機械システム工学専攻修士論文(2006)

6) 若生大輔:視覚による自律移動ロボットのボール把持タスクの実現,2006年度山形大学工学部卒業論文(2006)

計測結果

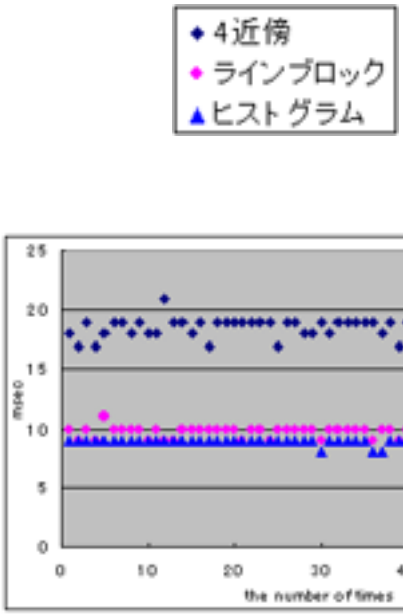

Fig. 8　First experiment

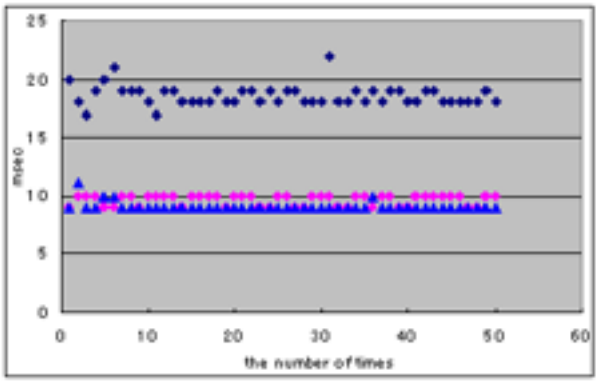

Fig. 9　Second experiment

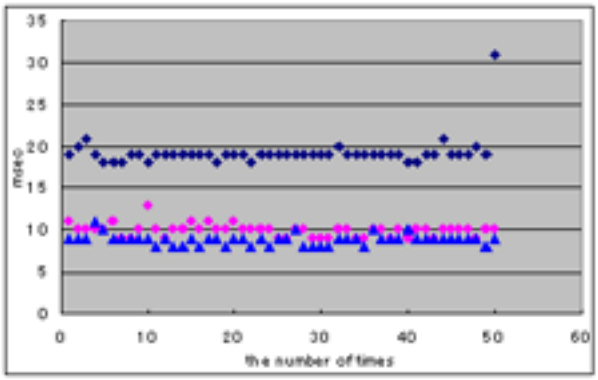

Fig. 10　Third experiment

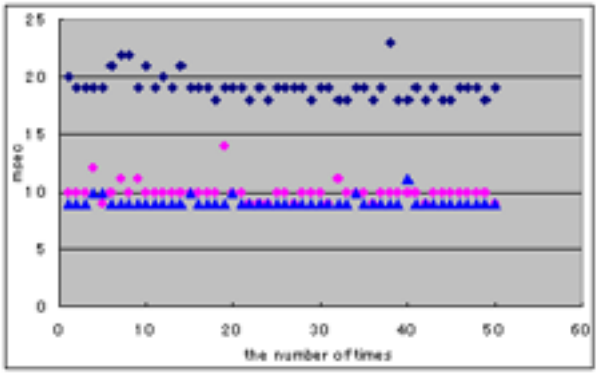

Fig. 11　Fourth experiment

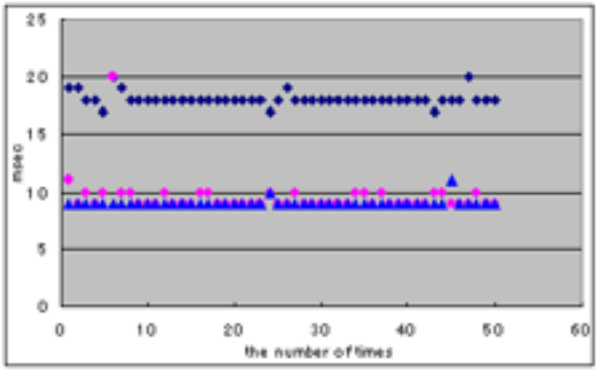

Fig. 12　Fifth experiment