J-09 画像処理による自動走行車の製作

発表者氏名：伊藤　祥吾

指導教員：小笠原　祐治

1.　はじめに

「目で見て、楽しめる物を作りたい」と云う思いから、私は自動走行車を製作しようと考えた。

2.　研究概要

本研究では、走行進路上の障害物を認識し、それを回避するラジコンカーを作成する。

ラジコンカーには、カメラが搭載されたマイコンを取り付け、障害物の認識は、そのカメラで撮影した画像を基に行う事とする。

3.　処理手順

障害物の検出には、物体の色情報を利用している。障害物の検出までの処理の手順は大きく分けて以下の通り 3 つある。

・前処理
「画像の取り込み」、「縮小･形式変換」
・画像処理
「色数の削減」、「色番号への置き換え」、「領域分割」
・障害物の位置計算

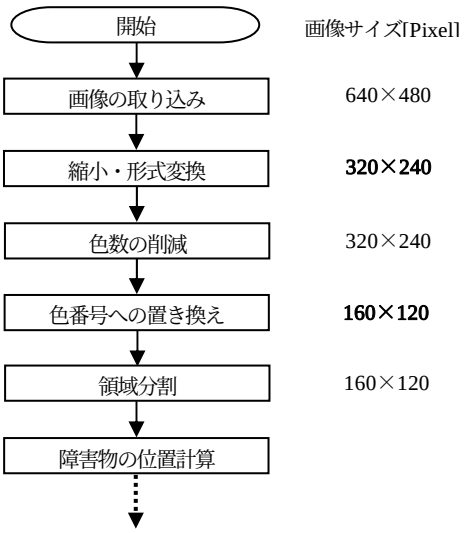

図 1.　処理の流れ

4.　各処理について

4.1　前処理

ここでは、「取り込んだ画像サイズを縮小」、「色を表現する形式の変換」を行っている。

4.1.1　サイズの縮小

取り扱うデータ数が多いと、処理に時間がかかってしまう。また、マイコンのメモリ容量も限られている為、画像サイズを縮小させる事によって、データ数を減らしている。

4.1.2　形式変換

取り込まれた画像は、YUV 形式である。これは輝度情報を基に色を表している。コンピュータで色を表現する際には RGB 形式を用いる為、変換処理を行っている。

4.2　画像処理

ここでは、「画像を表現している色数を減らす」、「RGB 値を色番号に置き換える」、「領域分割」の 3 つの処理を行っている。

4.2.1　色数の削減

物体は形状や光源からの距離によって、光の当たり方が変わり、同じ色でも異なった色の様に見えてしまう。障害物は色情報を用いて認識させる為、同じ物体でも RGB 値が異なっていると障害物として認識できない。その為、類似した色を同一の色と見なし、画像で表現されている色の数を減らす必要がある。

以下にその処理の手順を示す。

① コントラストの強調
② 色の階調数を各 16 階調に置き換える
③ 利用頻度の高い上位 10 色（代表色）を見付ける

4.2.2　色番号への置き換え

次に、画像の色情報を 4.2.1 で求めた代表色に置き換える。しかし、このままでは画像サイズが大きく､R,G,B と 3 つの値を保持させると、扱うデータ量が多くなってしまう。

その為、ここでは画像サイズの縮小と範囲内で最も用いられている代表色に対応する色番号を与えている。

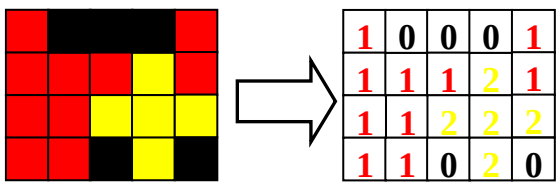

図 2.　色番号の割り当て

4.2.3　領域分割

領域分割では、障害物は同一色の集まりであると仮定した上で、類似した色の集まりを 1 つの領域と見なす。各画素に領域番号を付与し、同じ領域番号が付与された画素の集まりを 1 つの領域としている。

以下に領域分割の手順を示す。

画素の左上から順に次の処理を行う

① 領域番号が設定されているか確認し、設定されていない場合、その画素を処理の起点とする。

② 画素に新たな領域番号を付与し、隣接画素に対し、次の処理を行う。

③－① 領域番号が設定されていなく、起点とする画素と同じ色番号を保持する場合、②の処理を行う。

③－② それ以外の場合は、起点とする画素の次の画素に対し、①の処理を行う。

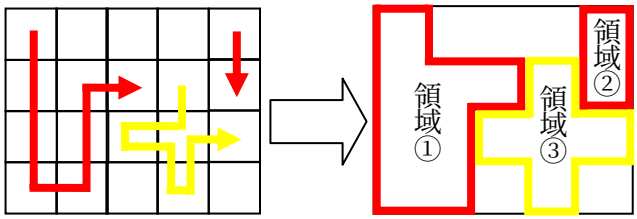

図 3. 領域分割

領域番号の付与では、同じ処理を何度も繰り返し行っている為、再起呼び出しを利用する事によって、処理の効率化を図った。

しかし、このまま再起呼び出しを行うと、同じ領域が続けば続く程、再起呼び出しの回数が多くなる為、スタックオーバフロー（スタックサイズが不足する事）となってしまう。

これを解決する為に、再起呼び出しの回数が一定回数を超えると、一旦再起呼び出しを終了し、その後、再度その続きから領域分割を行えるようにした。

4.3 障害物の位置計算

領域分割した結果から、画像における領域の高さ、幅、面積、中心座標が分かる。それらの情報を用いて、実際の障害物の位置を求めることが出来る。

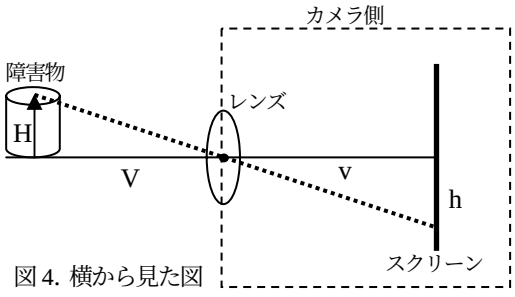

図 4. 横から見た図

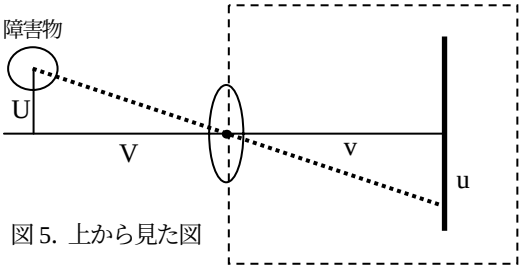

図 5. 上から見た図

H ：障害物の高さ[mm]
V ：カメラから障害物までの距離（奥行き）[mm]
U ：カメラから障害物までの距離（横方向）[mm]
h ：障害物の高さ[Pixel]
v ：レンズの焦点距離（定数）[Pixel]
u ：障害物までの距離（横方向）[Pixel]

上記の図は、画面上の障害物と実際の障害物の関係を示した図である。

この図から、下記の様な式が成り立つ。

$$V = Hv / h \quad , \quad U = Vu / v$$

この式を用いて、障害物の位置を特定する事が出来る。しかし、画像サイズを縮小している為、ある程度誤差が発生してしまう。1 画素当たりの誤差を求める式は下記の通りである。

$$\Delta V = Hv / (h \pm \text{縮小率}) - V$$

$$\Delta U = V * (u \pm \text{縮小率}) / v - U$$

なお、定数 v はカメラからの距離 30cm の時の高さ、幅共に 1cm の物体の画素数から求めた結果、v=700 である事が判明した。

5. 実行速度

昨年度と今年度のプログラムの実行速度を比較した結果、以下の通りとなった。

表 1. 実行速度の比較

処理	昨年度[ms]	今年度[ms]
前処理	126.37	125.82
画像処理	590.02	771.76
障害物の位置計算	0.40	1.47

昨年度のプログラムが障害物の色情報を指定している。しかし、それでは照明光の違いによる色の変化や指定していない色の障害物は認識できなくなってしまう。

今年度はその点を改善した為、処理に時間がかかる様になったが、他の箇所で実行速度を向上させている為、全体的には少し遅くなる程度で済んだ。

現状では処理に 1 秒近くかかってしまう。画像の解像度を下げる事によって、処理の高速化を図る事ができる。しかし、その分誤差が大きくなってしまう。

6. おわりに

現在は、障害物の位置計算までの処理を終えた。今後、残された作業は、走行方法を考える事だけである。障害物を認識する際の条件の検討や実行速度の向上、検出誤差等の課題は残されてはいるが、後少しで完成できるところまできて、間に合わなかったのは、非常に残念である。是非とも来年度の後輩にはこの研究を引き継いでもらい、完成まで導いて欲しいと切に願う。