HR-Trainer による歩行制御と画像認識

氏名　石岡　匠也、　鶴峯　義久

１．研究概要

写真１に示す、HR-Trainer というヒューマノイドロボットを用いる。運動学と逆運動学の研究をもとに２足歩行のプログラムを作成し、色の表現、色認識、エッジ検出処理の研究をもとにして、物体を回避する動作のプログラムの開発を目指した。

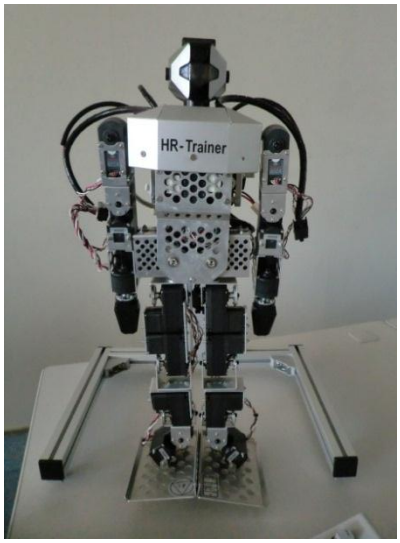


写真 1　HR-Trainer

２．研究の具体的内容

（１）全体の流れ

研究開始当初の全体の流れは図１に示す工程を予定していた。しかし、研究を進めて行く中で図１の工程では一人で行う作業が多く、効率が悪いことが判明したため、作業工程の見直しをした結果、図２に示す作業工程に変更した。

新しい工程表では、二足歩行プログラムと、オブジェクト検出プログラムはお互いに依存関係にないため、並列作業で研究を進めていくことにして、作業の効率化を期待した。

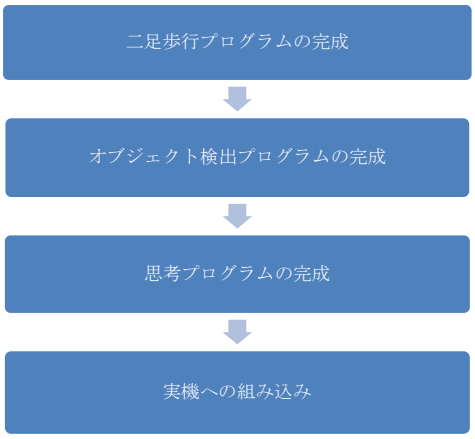


図 1　研究開始当初の工程表

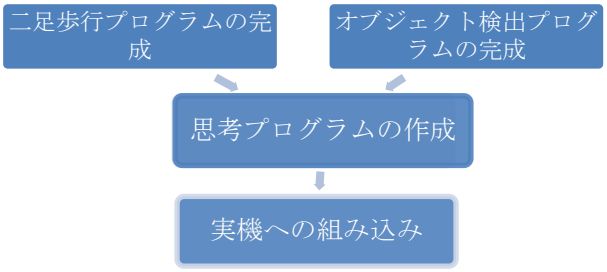


図 2　見直し後の工程表

３．２足歩行ロボット

（1）シミュレータの選定

歩行の動きは、ロボットの関節になるサーボモータの寿命を考え、先にコンピュータでシミュレートすることにした。シミュレータはオープンソースの物理エンジンである ODE(Open Dynamics Engine)を使用した。ODE は、初歩レベルの C 言語と高校生レベルの数学の知識だけで使用することができ、ホームページや本の資料、サンプルプログラムの多いことから採用した。

（2）シミュレータのカスタマイズ

ODE のサンプルプログラムの写真 2 と写真 3 をカスタマイズし、HR-Trainer 専用のシミュレータを作った。サンプルプログラムの写真 2 は、立ち幅跳びをするロボットであり、歩行はしない。

写真 2　ヒューマノイドロボット

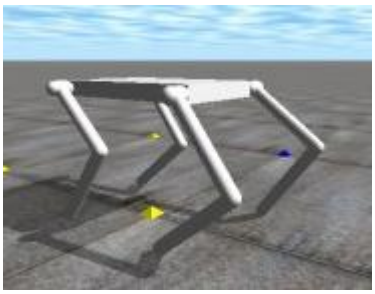


写真 3　4 足歩行ロボット

そこで、サンプルプログラムの写真 3 の歩行プログラムを参考にし、写真 4 の歩行プログラムを作成した。ここで作成したシミュレータの歩行プログラム使い、ソースプログラムの調整してから本体である HR-Trainer を歩行させた。このソースプログラムの調整は、モータの初期位置が、シミュレータが 0 度に対し、HR-Trainer は各関節ごとに違うので、それに合わせる必要があった。

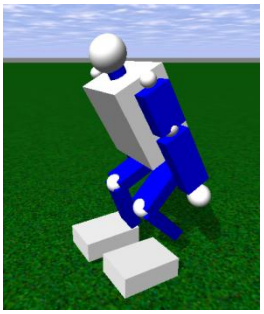


写真 2　HR-Trainer のシミュレータ

（a）ヒューマノイドロボットのカスタマイズ

ヒューマノイドロボットは、全長 120[cm]、体重 20[kg] であり、HR-Trainer より大きい。そのため、HR-Trainer の各パーツのサイズと質量を計測し、その値をヘッダーファイルの各パーツのパラメータに変更した。

また、HR-Trainer は足関節と股関節の x 軸回転と y 軸回転の座標が違う位置にある。股関節は x 軸回転より y 軸回転が下の位置にあるので、足関節の y 軸回転の座標を下に下げ、足関節は x 軸回転より y 軸回転が上の位置にあるので、足関節の y 軸回転の座標を上に上げた。

（3）運動学と逆運動学

ロボットの腕や足の関節の計算で使用した運動学と逆運動学の説明をする。

（a）運動学

運動学は各関節の角度から、先端位置を求めるものである。

図 1 は、θ 1 と θ 2 をロボットの関節、l1,l2 をロボットの腕や足に見立てた図形である。

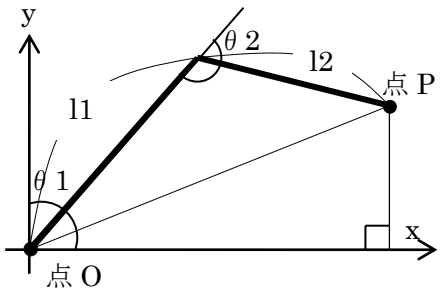


図 3　運動学の図形

運動学は、θ 1 と θ 2 の角度と、l1,l2 の長さで、点 O から点 P の長さを求めることが出来る。注意する点は、各関節の回転角度が同じでも回転させる順番が変わると先端位置が変わってしまう。だから計算方法は、関節を動かす順番に回転行列を掛けることにより求めるかことができる。

（b）逆運動学

逆運動学は、運動学の逆で先端位置から各関節の角度を求めるものだ。図 3 は逆運動学の図形である。

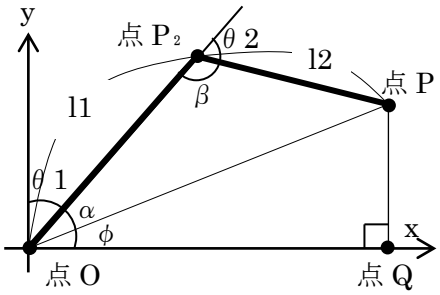


図 4　逆運動の図形

図 4 に示すようにリンク 1、リンク 2 の長さをそれぞれ l1,l2,先端位置をそれぞれ P_2,P とし、P から x 軸に垂直を降ろし、その交点を Q とする。図 4 の場合、点 P の座標と l1,l2 の長さから、$\theta 1$ と $\theta 2$ を求めることができる。

求め方をこれから説明する。

$\theta 1$ を求める。$\theta 1$ を簡単に求めると、

$$\theta 1 = \frac{\pi}{2} - (\phi + \alpha)$$

となる。ϕ はアーム先端位置 P(Px, Py)がわかっているので直角三角形 OPQ に着目すると

$$\phi = a\tan\frac{\overline{PQ}}{\overline{OQ}} = a\tan\frac{Py}{Px}$$

となる。

α は、余弦定理を使って求める。

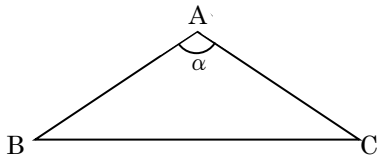


図 5　余弦定理

三角形 ABC に対して以下の式が成り立つ

$$\overline{BC}^2 = \overline{CA}^2 + \overline{AB}^2 - 2\overline{CA}\cdot\overline{AB}\cos\alpha$$

この余弦定理を使い、$\cos\alpha$ を求める。

$$-2\overline{CA}\cdot\overline{AB}\cos\alpha = \overline{BC}^2 - \overline{CA}^2 - \overline{AB}^2$$

$$\cos\alpha = \frac{\overline{CA}^2 + \overline{AB}^2 - \overline{BC}^2}{2\cdot\overline{CA}\cdot\overline{AB}}$$

三角形 OP_2P に着目し余弦定理を $\overline{OP_2}$ と $\overline{OP}$ の成す角 α について適用すると、

$$\cos\alpha = \frac{l1^2 + \overline{OP}^2 - l2^2}{2\cdot l1\cdot\overline{OP}}$$

$$\sin\alpha = \pm\sqrt{1-\cos^2\alpha}$$

ここで、$\overline{OP} = \sqrt{Px^2 + Py^2}$

$$\tan\alpha = \frac{\sin\alpha}{\cos\alpha}$$

$$\alpha = a\tan\left(\pm\frac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}\right)$$

$$= \pm a\tan\left(\frac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}\right)$$

ゆえに $\theta 1 = \frac{\pi}{2} - a\tan\frac{Py}{Px} - a\tan\left(\frac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}\right)$ (式 1)

または $\theta 1 = \frac{\pi}{2} - a\tan\frac{Py}{Px} + a\tan\left(\frac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}\right)$ (式 2)

$\theta 1$ は 2 つの解がある。

次に $\theta 2$ をもとめる。

図 4 より　$\theta 2 = \pi - \beta$

次に β を求める。$\overline{OP_2}$ と $\overline{P_2P}$ の成す角 β とする。β を求めるために三角形 OP_2P の β について余弦定理を適用する。

$$\cos\beta = \frac{l1^2 + l2^2 - \overline{OP}^2}{2\cdot l1\cdot l2}$$

$$\sin\beta = \pm\sqrt{1-\cos^2\beta}$$

ゆえに $\theta 2 = \pi - a\tan\left(\frac{\sqrt{1-\cos^2\beta}}{\cos\beta}\right)$ (式 3)

または $\theta 2 = \pi + a\tan\left(\frac{\sqrt{1-\cos^2\beta}}{\cos\beta}\right)$ (式 4)

$\theta 1, \theta 2$ はそれぞれ 2 個の解をもっているが、$\theta 1$ が決まると $\theta 2$ が決まるので 2 通りの姿勢をとることができる。

逆運動学は運動学と違い、点 P の座標の位置によって解がないときがある。つまり、点 P にアームの先端が届かないとき、解が出てこない。

この 2 つの関節の逆運動学を基にロボットの足関節の角度を求める。

(4) ロボットの足関節の計算

(a) ロボットを横から見たときの計算

ロボットの足の部分を図式化したものを図 6 に示す。逆運動学の図 4 と同じような図形になっている。O1,P1 は股関節部分、P4 は膝関節部分、P5 は足関節、P は足の裏の先端部分である。l4 は股関節と膝関節の距離、l5 は膝関節と足関節の距離、l6 は足関節と足の裏の先端部分の距離である。

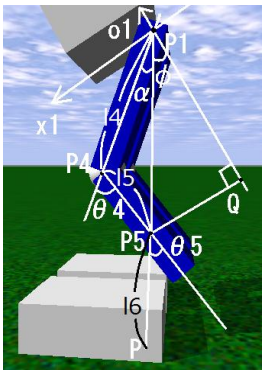


図 6　ロボットを横から見たときの図形

まず、逆運動学を求めるためには P 点の座標が必要になる。なお、P 点の座標は股関節の中心が基点となる点 O1 である。座標の説明は後の「(5) 歩行」でする。

今回、股関節の z 軸回転である θ 1 は使ってないので計算を省く。

股関節の y 軸回転 θ 3、膝間接の y 軸回転 θ 4 から求めていく。始めに、$\overline{P1P5}$ を求める。P5 の座標は、

$$P5 = P - l6$$

$$P5x = Px, \quad P5y = Py, \quad P5z = Pz - l6$$

になる。ここからは、逆運動学の計算と同じである。$\overline{P1P5}$ は

$$\overline{P1P5} = \sqrt{P5x^2 + P5y^2 + P5z^2} \quad (式\ 5)$$

になる。$\overline{P1P5}$ を使い、ほかも求める。

$$\cos\alpha = \frac{l3^2 + \overline{P1P5}^2 - l4^2}{2 \cdot l3 \cdot \overline{P1P5}}, \cos\beta = \frac{l3^2 + l4^2 - \overline{P1P5}^2}{2 \cdot l3 \cdot l4}$$

$$\alpha = a\tan\left(\frac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}\right), \beta = a\tan\left(\frac{\sqrt{1-\cos^2\beta}}{\cos\beta}\right) \quad (式\ 6)$$

$$\phi = a\tan\frac{\overline{QP5}}{P1Q} = a\tan\left(\frac{P5x}{\sqrt{P5x^2 + P5z^2}}\right) \quad (式\ 7)$$

$$\theta 3 = \frac{\pi}{2} - \phi - \alpha$$

$$= \frac{\pi}{2} - a\tan\left(\frac{P5x}{\sqrt{P5x^2 + P5z^2}}\right) - a\tan\left(\frac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}\right) \quad (式\ 8)$$

$$\theta 4 = \pi - \beta$$

$$= \pi - a\tan\left(\frac{\sqrt{1-\cos^2\beta}}{\cos\beta}\right) \quad (式\ 9)$$

次に、足関節の y 軸回転である θ 5 を求める。θ 5 は、

$$\theta 5 = \theta 4 - \phi - \alpha$$

である。

(b) ロボットを前から見たときの計算

図 7 はロボットの進行方向から見た図形である。図 6 を正面から見たもので、同じものである。f はリンク l4 を伸ばした直線と地面の交点である。

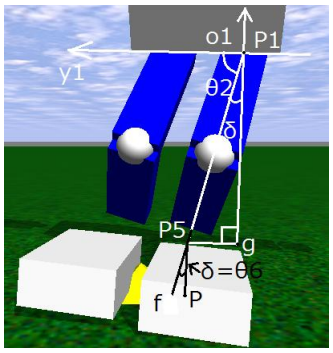


図 7　ロボットを進行方向から見た図形

θ 2 と θ 6 を求める。θ 2 と θ 6 は、

$$\theta 6 = \angle fP5P = \angle P5P1g = a\tan\frac{\overline{gP5}}{P1g} = a\tan\left(\frac{P5y}{\sqrt{P5x^2 + P5z^2}}\right) \quad (式\ 10)$$

$$\theta 2 = \frac{\pi}{2} - \angle P5P1g = \frac{\pi}{2} - a\tan\left(\frac{P5y}{\sqrt{P5x^2 + P5z^2}}\right) \quad (式\ 11)$$

となる。

(c) inverseKinematics 関数

逆運動学を C++でプログラミングしたものが、inverseKinematics 関数である。2 足歩行の逆運動学の関数は、4 足歩行の逆運動学の関数をカスタマイズし、作成した。

```
/*** 逆運動学の計算 (calculate inverse kinematics ***/
void  inverseKinematics(dReal x, dReal y, dReal z,
                        dReal *ang1, dReal *ang2, dReal *ang3,int posture)
{
  dReal l1a = 0, l3a = l3 + r3/2;

  double c3 = (x*x + z*z + (y-l1a)*(y-l1a) - (l2*l2+l3a*l3a))/(2*l2*l3a);
  double s2 = (y-l1a) / (l2 + l3a*c3);
  double c2 = sqrt(1 - s2 * s2);
  double c1 = (l2 + l3a*c3)*c2/sqrt(x*x+z*z);
    // printf("c3=%f s2=%f c2=%f c1=%f ¥n", c3,s2,c2,c1);
    if (sqrt(x*x+y*y+z*z) > l2 + l3) {
    printf(" Target point is out of range ¥n");
    }

  switch (posture) {
  case 1: // 姿勢１ (posture 1)
    *ang1 =   atan2(x,-z) - atan2(sqrt(1 - c1*c1),c1);
    *ang2 = - atan2(s2,c2);
    *ang3 =   atan2(sqrt(1-c3*c3),c3); break;
  case 2: // 姿勢２ (posture 2)
    *ang1=   atan2(x,-z) + atan2(sqrt(1 - c1*c1),c1);
    *ang2= - atan2(s2,c2);
    *ang3= - atan2(sqrt(1-c3*c3),c3); break;
  case 3:  // 姿勢３ (posture 3)
    *ang1 =   M_PI + (atan2(x,-z) - atan2(sqrt(1 - c1*c1),c1));
    *ang2 = - M_PI +  atan2(s2,c2);
    *ang3 = - atan2(sqrt(1-c3*c3),c3); break;
  case 4:  // 姿勢４ (posture 4)
    *ang1 =  M_PI + atan2(x,-z) + atan2(sqrt(1 - c1*c1),c1);
    *ang2 = -M_PI + atan2(s2,c2);
    *ang3 =  atan2(sqrt(1-c3*c3),c3); break;
  }
}
```

上のプログラムが 4 足歩行の逆運動学のプログラムである。4 足歩行ロボットは、１つの足に関節が３つしかない。だから、計算が 2 足歩行ロボットより簡単になっている。2 足歩行ロボットは１本の足の関節数が増えたため、ang が 1 から 6 に増えている。4 足歩行ロボットは、4 つの姿勢で歩行するパターンがあるので姿勢 1 から 4 まで書かれている。2 歩行ロボットは、１つの姿勢しかない。

次のプログラムが作成した 2 足歩行ロボットのソースプログラムである。

```
static void inverseKinematics(dReal x, dReal y, dReal z,
                        dReal *ang1, dReal *ang2, dReal *ang3,
                        dReal *ang4, dReal *ang5, dReal *ang6)
{
    dReal l3 = 0.088, l4 = 0.074, l6 = 0.0780;

    dReal p5x = x, p5y = y, p5z = z-l6;
    dReal p1p5 = sqrt(p5x*p5x + p5y*p5y + p5z*p5z);
    dReal ca = (( l3*l3 + p1p5*p1p5 - l4*l4 ) / ( 2 * l3 * p1p5 ));
    dReal cb = (( l3*l3 + l4*l4 - p1p5*p1p5 ) / ( 2 * l3 * l4 ));
    dReal a = atan2( sqrt(1 - ca*ca), ca);
    dReal b = atan2( sqrt(1 - cb*cb), cb);
    dReal iA = atan2( p5x, sqrt(p5y*p5y + p5z*p5z));

            *ang1 = 0;
            *ang2 = - atan2(p5y, sqrt( p5x*p5x + p5z*p5z ));
            *ang3 = - iA - a;
            *ang4 = M_PI - b;
            *ang5 = *ang4 - iA - a;
            *ang6 = atan2(p5y,sqrt(p5x*p5x+p5z*p5z))
                    + atan2(y-p5y,sqrt((x-p5x)*(x-p5x)+(z-p5z)*(z-p5z)));
}
```

ソースプログラムの dReal は ODE の演算子であり、数値を表している。iA は、φの代わりに使っている。l3,l4,l6 の長さをあらかじめ決めておき、点 P の xyz 座標を入力してやると、各関節の角度を計算してくれる関数である。点 P が先端のとどかない位置のとき、解がないのでエラーが帰ってくる。

（5）歩行

（a）歩行手順

逆運動学を使い、足を動かすので足の座標を、歩行手順に従って右左の足それぞれのステップごとに決める。歩行手順をまとめた表が表 1 である。

「進行方向から見たとき」と「横から見たとき」の列は、ロボットの足を図式化したものである。右足と左足の列は、それぞれの足の点 P の座標である。z0 は、点 O から点 P までの距離で、z1 は点 O から足をあげたときの距離であり、z0-z1 で足を上げる距離が求められる。y1 は左右の重心の移動の距離である。fs は、足を前に出す距離である。

表 1　歩行手順の表

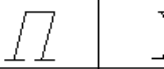
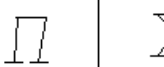
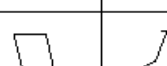
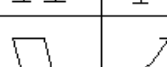
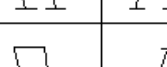
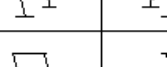
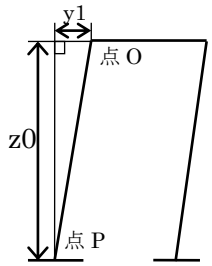
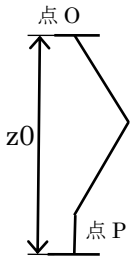
STEP	進行方向から見たとき	横から見たとき	左足			右足		
			x	y	z	x	y	z
1			0	y1	z0	0	y1	z0
2			f1	y1	z1	0	y1	z0
3			fs	y1	z0	0	y1	z0
4			f1	y1	z0	-f1	y1	z0
5			f1	-y1	z0	-f1	-y1	z0
6			0	-y1	z0	-f1	-y1	z0
7			0	-y1	z0	-f1	-y1	z1
8			0	-y1	z0	0	-y1	z0

表 1 を各ステップごとに、説明する。

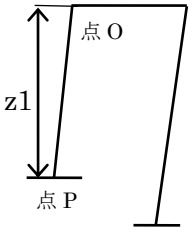
STEP1　重心を左に移動

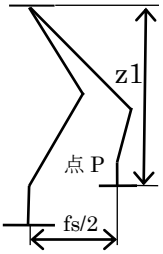
足の図形の説明である。右が進行方向で、左が横から見た図である。



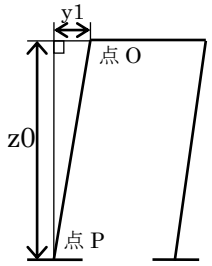


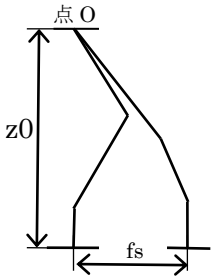
STEP2　右足離地



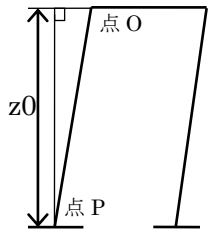


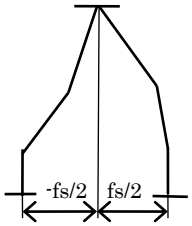
STEP3　右足着地



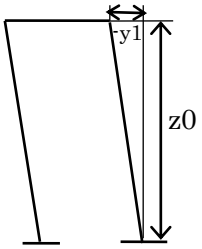


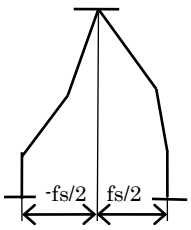
STEP4　前後の重心の移動



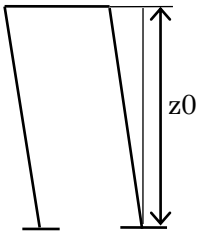


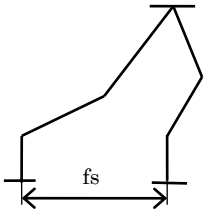
STEP5　重心を右に移動



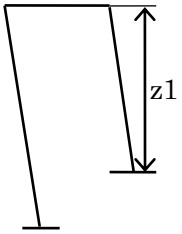


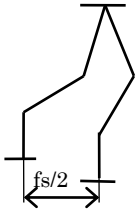
STEP6　前に重心の移動



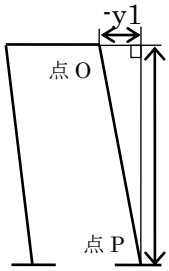


STEP7　左足離地





STEP8　左足着地



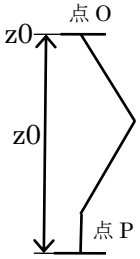


図 8　足の図形

（b）calcAngle 関数

```
void calcAngle()        /*** 目標角度の計算  ***/
{
  dReal z0 = -0.4,z1 = -0.37;
  dReal y1 = 0.05, fs = 0.2;
  dReal f1 = fs/4, f2 = fs/2, f3 = 3 * fs/4, f4 = fs;
  dReal  traj[12][LEG_NUM][3] = { // 目標軌道点

    // 左前leg0遊脚 左後leg1    右後 leg2    右前leg3
    {{ 0, y1,z0},{  0, y1,z0},{  0, y1,z0},{  0, y1,z0}},// 0 重心移動
    {{f2, y1,z1},{  0, y1,z0},{  0, y1,z0},{  0, y1,z0}},// 1 離地
    {{f4, y1,z0},{  0, y1,z0},{  0, y1,z0},{  0, y1,z0}},// 2 着地
    {{f3,-y1,z0},{-f1,-y1,z0},{-f1,-y1,z0},{-f1,-y1,z0}},// 3 重心移動
    //    leg0        leg1     leg2 遊脚(swing)  leg3
    {{f3,-y1,z0},{-f1,-y1,z0},{ f1,-y1,z1},{-f1,-y1,z0}},// 4 離地
    {{f3,-y1,z0},{-f1,-y1,z0},{ f3,-y1,z0},{-f1,-y1,z0}},// 5 着地
    {{f2,-y1,z0},{-f2,-y1,z0},{ f2,-y1,z0},{-f2,-y1,z0}},// 6 重心移動
    //    leg0        leg1      leg2        leg3 遊脚(swing)
    {{f2,-y1,z0},{-f2,-y1,z0},{ f2,-y1,z0},{  0,-y1,z1}},// 7 離地
    {{f2,-y1,z0},{-f2,-y1,z0},{ f2,-y1,z0},{ f2,-y1,z0}},// 8 着地
    {{f1, y1,z0},{-f3, y1,z0},{ f1, y1,z0},{ f1, y1,z0}},// 9 重心移動
    //   leg0    leg1 遊脚(swing)  leg2         leg3
    {{f1,y1, z0},{-f1, y1,z1},{ f1, y1,z0},{ f1, y1,z0}},// 10 離地
    {{f1,y1, z0},{ f1, y1,z0},{ f1, y1,z0},{ f1, y1,z0}} // 11 着地
  };

  dReal angle1, angle2, angle3;
  int posture = 4;                                      // 姿勢
  for (int i = 0; i < 12; i++) {
    for (int k = 0; k < LEG_NUM; k++) {
      inverseKinematics(traj[i][k][0],traj[i][k][1],     // 逆運動学
          traj[i][k][2],&angle1, &angle2, &angle3,posture);
      gait[i][k][0] = angle1;
      gait[i][k][1] = angle2;
      gait[i][k][2] = angle3;
    }
  }
}
```

calcAngle 関数も inverseKinematics 関数と同様に 4 足歩行ロボットのソースプログラムから作成した。上のプログラムは、4 足歩行の calcAngle 関数である。2 足歩行ロボットの足は、2 本だから 4 足歩行より 1 ステップの目標点は少ない。しかし､1本の足の関節数は増えるのでgaitとangleは 6 個に増える。

下のプログラムが 2 足歩行ロボットの calcAngle 関数である。2 足歩行ロボットの各 STEP ごとに逆運動学を求めている。

```
static void calcAngle()
{

    dReal z0 = -0.04, z1 = -0.03;//z0:地面までの高さ  z1:最高到達点
    dReal y1 = 0.015, fs = 0.1;// y1:左右の変位  fs:歩幅
    dReal f1 = fs/2;
    dReal traj[STEP][LEG_NUM][3] = {
        {{ 0, y1,z0},{  0, y1,z0}},
        {{f1, y1,z1},{  0, y1,z0}},
        {{fs, y1,z0},{  0, y1,z0+0.017}},
        {{f1,y1,z0-0.025},{-f1,y1,z0}},
        {{f1,-y1,z0-0.025},{-f1,-y1,z0}},
        {{ 0,-y1,z0},{-fs,-y1,z0+0.02}},
        {{ 0,-y1,z0-0.02},{-f1,-y1,z1+0.02}},
        {{ 0,-y1,z0},{-0,-y1,z0}}

    };

    dReal angle1, angle2, angle3, angle4, angle5, angle6;

    for( int v = 0; v < STEP; v++){
         for( int k = 0; k < 2 ; k++){
        //逆運動学の計算
            inverseKinematics(traj[v][k][0], traj[v][k][1],traj[v][k][2],
                &angle1, &angle2, &angle3, &angle4, &angle5, &angle6);

                    gait[v][k][0] = angle1;
                    gait[v][k][1] = angle2/M_PI*180;
                    gait[v][k][2] = angle3/M_PI*180;
                    gait[v][k][3] = angle4/M_PI*180;
                    gait[v][k][4] = angle5/M_PI*180;
                    gait[v][k][5] = angle6/M_PI*180;
         }
    }
}
```

traj[STEP][LEG_NUM][3]の中に点Pのxyz座標を入れている。z0 に数値を足し引きして、誤差を修正している。逆運動学の inverseKinematics 関数を使い､各 STEP の関節の角度を求める。M_PI はπのことである。求めた角度は、ラジアンになっているので度に変換してやるために、angle/M_PI*180 の計算をしている。そして、この値を gait[STEP][LEG_NUM][3]の中に代入しておく。gait の値をステップごとに各関節に入れることにより、歩行が可能となる。

４．オブジェクト検出プログラムの開発

（１）ブジェクト検出プログラムの機能

HR-Trainer に搭載されている CCD(Charge Coupling Device)カメラからロボット前方の画像データを取得し、その画像内に目的とするオブジェクト（物）があれば検出し検出した物の座標を出力する。

（２）画像認識のライブラリの選定

画像認識を最初から勉強しさらにプログラムを開発していては課題研究の期間中の開発完了は難しいと判断したため、Intel 社の開発したオープンソースライブラリである OpenCV を使用して研究を進めていくことにした。

（３）OpenCV とは

OpenCV は Intel 社によって開発された、画像認識に関連する機能のライブラリで、コンピュータビジョンと呼ばれる、画像認識・解析が主な用途である。C 言語、C++言語、Physon と多くの言語で記述が可能であり、さらにプラットフォームも Windows,Linux,Unix,Android など複数対応している。オープンソースソフトウェア（OSS）として提供されているために誰でも無料で利用することが可能である。

（４）CCD カメラとは

CCD カメラは受光部に CCD を用いているカメラで、ディジタルカメラやディジタルビデオカメラなど多くの製品に使用されている。なお、HR-Trainer には頭部に搭載されている。

（５）RGB 方式とは

色の表現方法の一つに図 9 に示す R（red:赤）G（Green:緑）B（Blue:青）の三原色で表現する RGB 方式がある。

RGB 方式で図 10 の黄緑のサンプルカラーを各色最小値 0.0 最大値 1.0 とする 0.0 から 1.0 の範囲で表現すると式 12 のようになる。

$$\begin{cases} R = 0.478 \\ G = 0.882 \\ B = 0.176 \end{cases} \quad \text{（式 12）}$$

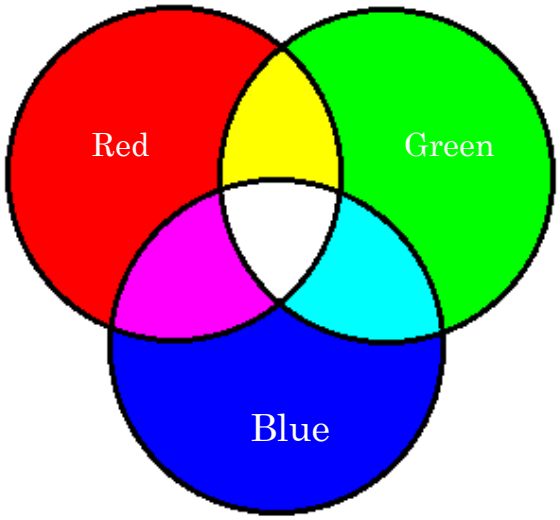


図 9　ＲＧＢ方式：加法混色

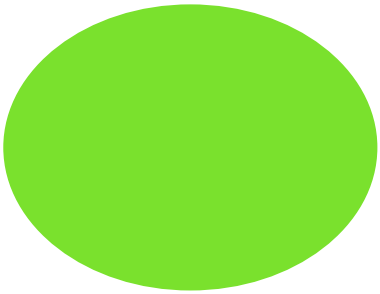


図 10　サンプルカラー

（６）HSV 方式とは

色を表現する方法に図 11 に示す H（Hue:色相）S（Saturation:彩度）V（Value:明度）で表現される HSV 方式がある。H,S,V の表す要素は以下のとおりである。

・色相（Hue）

色合いを表す。

赤、緑および青を 0~360°（0~2n）の角度を用いて表す。

・彩度（Saturation）

色の鮮やかさを表すもので、どれだけ純色に近いかを表す。

・明度(Value)

色の明るさを表す。

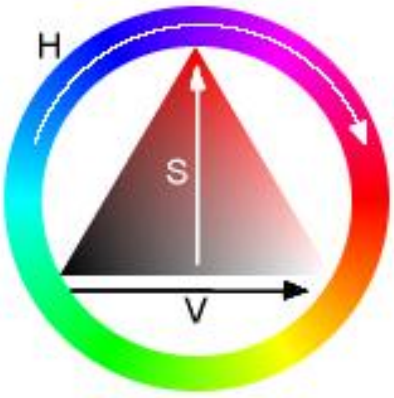


図 11　HSV 方式

HSV 方式で図 10 のサンプル画像を H が 0°から 360°の範囲内で、S,V ともに最小値 0.0 最大値 1.0 とする 0.0 から 1.0 の範囲にあるとして表すと式 13 のようになる。

$$\begin{cases} H = 94.6° \\ S = 0.79 \\ V = 0.52 \end{cases} \quad \text{（式 13）}$$

（７）RGB → HSV変換とは

一般的な jpg や png、bmp などの画像データファイルは画像のピクセルデータを RGB 方式で表現されており。また、HR-Trainer に搭載しているカメラから送られてくる画像も RGB 方式のデータで送られてくる。

RGB で表現されている画像データを HSV 方式に基づく色認識に使用するためには、RGB 方式から HSV 方式への変換が必要になる。

変換の式は、以下の式 14、式 15、式 16 である。

I_{max}を R, G, B の中の最大値

I_{min}を R, G, B の中の最小値

とすると

$$\begin{cases} H = 60° \times \frac{G-B}{I_{max}-I_{,min}} & [deg]\ (I_{max} = R) \\ H = 60° \times \frac{B-R}{I_{max}-I_{min}} + 120°[deg] & (I_{max} = G) \\ H = 60° \times \frac{R-G}{I_{max}-I_{min}} + 240°[deg] & (I_{max} = B) \end{cases}$$

（式 14）

$$S = \frac{I_{max}-I_{min}}{I_{max}} \quad \text{（式 15）}$$

$V = I_{max}$（式 16）

（８）HSV → RGB変換とは

HSV 方式で処理した画像をファイルに保存したりするときにはHSV方式の画像をRGB方式の画像に変換する必要がある。HSV 方式から RGB 方式への変換は以下の式 17 によって行われる。

$$h = \left\lfloor \frac{H}{60} \right\rfloor$$

$$p = V \times (1 - S)$$

$$Q = V \times \left\{1 - S \times \left(\frac{H}{60} - h\right)\right\}$$

$$T = V \times \left\{1 - S \times \left(1 - \frac{H}{60} - h\right)\right\}$$

とすると、

$$\begin{cases} R = V \\ G = T \ (h = 0) \\ B = P \end{cases}$$
$$\begin{cases} R = Q \\ G = V \ (h = 1) \\ B = P \end{cases}$$
$$\begin{cases} R = P \\ G = Q \ (h = 2) \\ B = V \end{cases}$$
$$\begin{cases} R = P \\ G = Q \ (h = 3) \\ B = V \end{cases}$$
$$\begin{cases} R = T \\ G = P \ (h = 4) \\ B = V \end{cases}$$
$$\begin{cases} R = V \\ G = P \ (h = 5) \\ B = Q \end{cases}$$

(式 17)

（９）カーネルとは

画像処理を行うとき、

表 2　カーネルの例のように中心を注目輝度、その周辺を周辺輝度とする正方形のレート表をかけ合わせて注目座標の輝度を求める。この表のことをカーネルと呼ぶ。

表 2　カーネルの例

	周辺輝度	
	中心輝度	
	周辺輝度	

（10）ガウシアン平滑処理とは

一般的な画像では注目画素に近い画素の輝度値は注目がその輝度値と近く、遠くなればなるほど注目画素との輝度値の差が大きくなる場合がある。

このことを考慮し、注目画素に近いほど、平均値を計算するときの重みを大きくし、遠くなるほど重みを小さくなるよう式 18 のガウス分布の関数を用いてレートを計算しているものがガウシアンフィルタである。

$$f(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$(式 18)

式 18 でσの値が小さいほど平滑化の効果は小さくなり。大きいほど効果が大きくなるが、よく表 4 や表 5 のカーネルが用いられる。

ガウシアン平滑化処理を行うと、画像がぼやかすことができる。

表４　３×３のカーネル

$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$
$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$
$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$

表５　５×５のカーネル

$\frac{1}{255}$	$\frac{4}{255}$	$\frac{6}{255}$	$\frac{4}{255}$	$\frac{1}{255}$
$\frac{4}{255}$	$\frac{16}{255}$	$\frac{1}{255}$	$\frac{16}{255}$	$\frac{4}{255}$
$\frac{6}{255}$	$\frac{4}{255}$	$\frac{36}{255}$	$\frac{24}{255}$	$\frac{6}{255}$
$\frac{4}{255}$	$\frac{16}{255}$	$\frac{4}{255}$	$\frac{16}{255}$	$\frac{4}{255}$
$\frac{1}{255}$	$\frac{4}{255}$	$\frac{6}{255}$	$\frac{4}{255}$	$\frac{1}{255}$

写真 5　エッジ検出処理前の画像

（11）微分フィルタとは

画像の輝度値に対して、隣り合う画素の輝度の差が大きいほど、画像のエッジだと考えることができ、画像のピクセルの列番号をi,行番号をjとすると、画像データ$f(i,j)$に対して表２のようになる。

縦方向の偏微分を行うときは画像データの並びがボトムアップかボトムダウンかでどちらかを使い分ける。

表２　微分フィルタのカーネル

●横方向の差分(偏微分)

$\Delta_x f(i,j)$は以下のカーネルで求めることができる。

0	0	0
0	-1	1
0	0	0

●縦方向の差分(偏微分)

$\Delta_y f(i,j)$は以下のカーネルで求めることができる。

0	0	0
0	-1	0
0	1	0

または、

0	1	0
0	-1	0
0	0	0

この時のエッジの強さは、式 19 で求められる。

$$\sqrt{\{\Delta_x f(i,j)\}^2 + \{\Delta_y f(i,j)\}^2} \quad (式\ 19)$$

エッジの傾きは式 20 で求められる。

$$\tan^{-1}\frac{\Delta_x f(i,j)}{\Delta_y f(i,j)} \quad (式\ 20)$$

この方法で写真 5 をエッジ検出すると写真 7 のようになる。

写真 7　微分フィルタによるエッジ検出

（12）ソーベルフィルタ(Sobel filter)とは

表３　ソーベルフィルタのカーネル

●横方向の差分

-1	0	1
-2	0	2
-1	0	1

●縦方向の差分

1	2	1
0	0	0
-1	-2	-1

または

-1	-2	-1
0	0	0
1	2	1

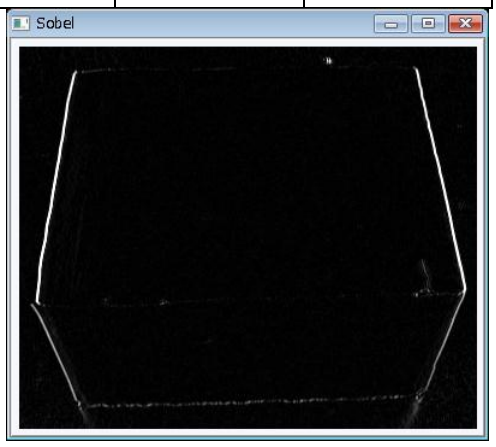


写真 8　ソーベルフィルタによるエッジ検出

微分フィルタではノイズに弱いため、横方向、または縦方向のエッジを計算してから縦方向、または横方向にガウシアン平滑化処理を行ったのがソーベルフィルタである。

写真 5 をソーベルフィルタを用いてエッジ検出をすると、写真 8 のようになる。

（13）キャニーフィルタ（Canny filter）とは

キャニーフィルタはガウシアンフィルタとソーベルフィルタとを組み合わせることで細線化されたエッジを検出するフィルタである。また、OpenCV でのキャニーフィルタはエッジの有無で二値化されている。

写真 5 の画像をキャニーフィルタを用いてエッジ検出をすると写真 9 の様になる。

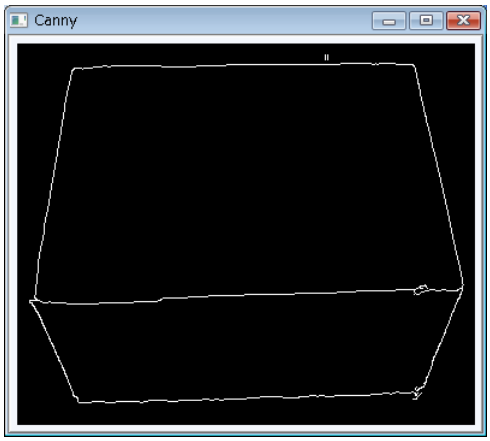


写真 9　キャニーフィルタによるエッジ検出

（14）オブジェクトの検出方法の決定

（a）オブジェクトと背景の切り離し方法

色の表現方法で HSV 方式は、オブジェクトへの光の当たり方が変わっても明度が変化するだけで、色相はほとんど変化しないため、HSV 方式を用いてオブジェクトの色を色相で判別することで、色の判別プログラムを簡略化できるため、HSV 方式を用いてオブジェクトを色認識し、背景を黒に塗りつぶし、オブジェクトのみを残すようにした。

（b）エッジの検出方法の決定

オブジェクトの検出方法は、微分フィルタ、ソーベルフィルタ、キャニーフィルタをすべて試し、その結果、写真 9 のようにキャニーフィルタを用いることで、最もエッジがきれいに検出できたため、キャニーフィルタによるエッジ検出を行うことにした。

（c）オブジェクト検出の効率化

オブジェクトと背景を切り離すときに写真 11 のように背景とオブジェクトの切り離しが不十分な状態でゴミが残ってしまっている画像でエッジ検出を行うと写真 12 のように不要なエッジまで検出されてしまいオブジェクトとロボットとの距離を検出するときに支障が出る。

これを解決するために、背景とオブジェクトの切り離しで不要な部分が残ってしまった写真 11 にガウシアン平滑化処理を施し、写真 13 のように画像をぼやかした。その後、エッジ検出を行うと、写真 14 のように目に見えるようなゴミの部分はなくなりオブジェクトの輪郭のみがきれいに検出されるようになった。

写真 10 オブジェクト

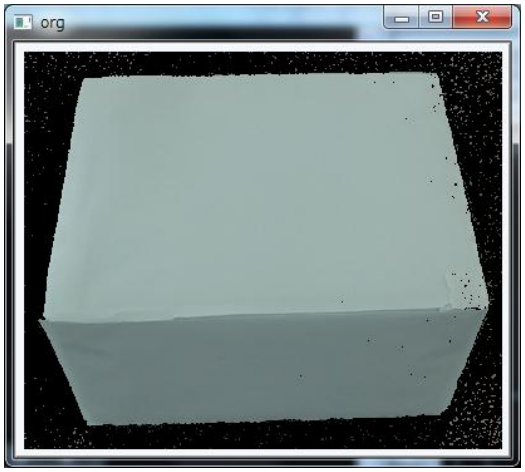


写真 11　画像から背景を消した画像

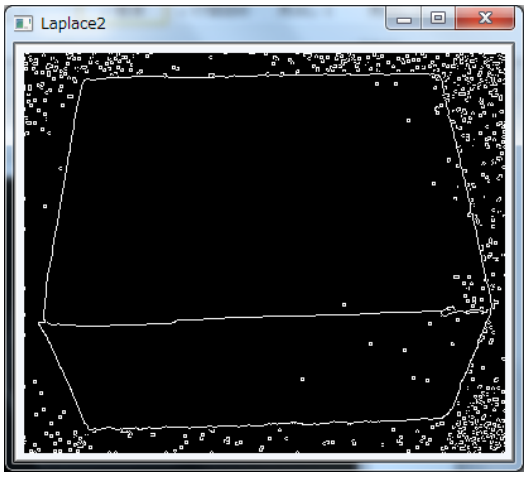


写真 12　写真 11 をエッジ検出した結果

写真 13 ガウシアンフィルタを施したもの

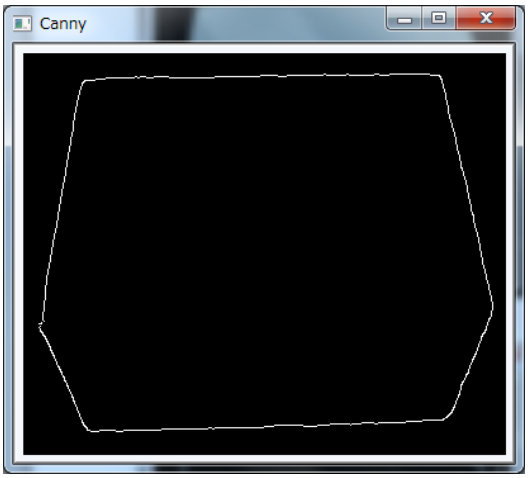


写真 14　エッジ検出されたオブジェクト

(15) オブジェクト検出プログラムの流れ

(a) 処理の前提条件

今回はオブジェクト検出プログラムをできるだけ簡素にするために以下に示すことを処理の条件とする。

・オブジェクトは単色で塗りつぶされている。

・ロボットの行動する範囲は屋内の狭い区域で、オブジェクト以外の単色になっており、ロボットに搭載されている CCD カメラで行動範囲外を撮影できないように高い壁で囲われている。

・オブジェクトの画像を写真 10 に示す。

(b) 処理の流れ

処理の簡単な流れを図 12 に示す。

オブジェクト検出プログラムは思考プログラムから呼び出される。

（ア）ロボットから CCD カメラの映像を取得する。

（イ）前提条件より、オブジェクトの色とロボットの周囲の色とは一致しないので、色でオブジェクトを判別しオブジェクトのみを残し、その他の部分は全て黒で塗りつぶしてしまう。

（ウ）オブジェクトのみを残した画像にエッジ処理を施すことによって、画像中にオブジェクトの各辺がどこにあるのかがわかるようになる。

（エ）写真 13 は白(255)もしくは黒(0)に二値化されており、エッジは白で描画されている。ここで、写真 14 を図 13 のように右下から x 方向に走査していき、初めてエッジが現れる点、すなわち白(255)を検出した点の座標がオブジェクトとロボットとのもっとも近い点の座標となる。もしも、白（255）を検出しなければ、オブジェクトがないということで座標(-1,-1)を出力する。

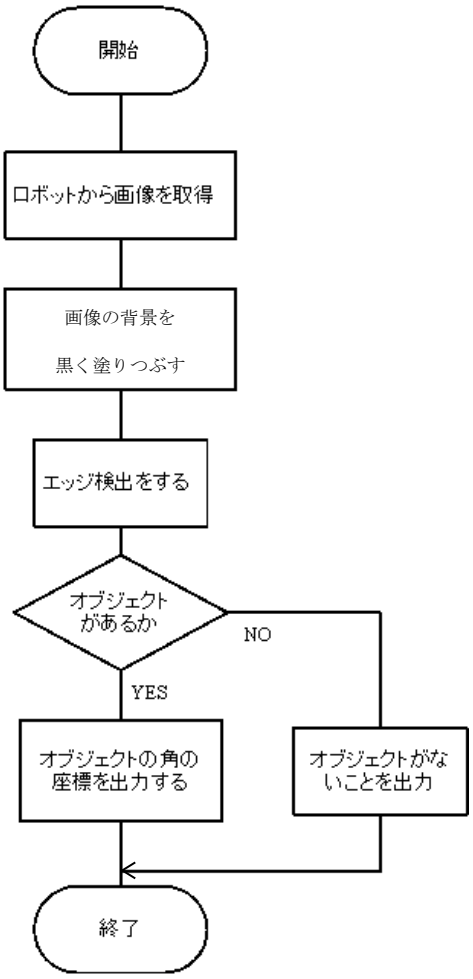


図 12　オブジェクト検出の簡単な流れ

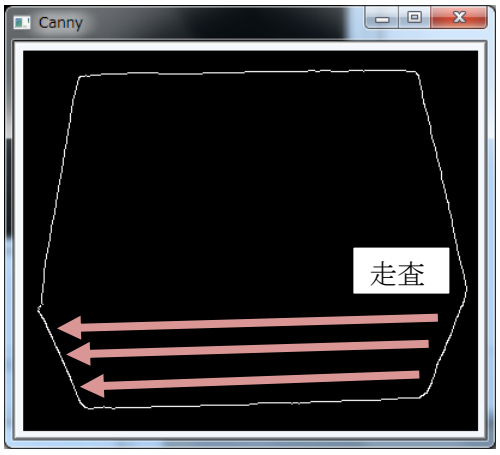


図 12　オブジェクトとロボットの最も近い座標を走査する。

5．歩行ルーチン

2 足歩行のプログラムとオブジェクト検出プログラムを使い、歩行ルーチンを考える。

図 13 は、歩行ルーチンのフローチャートである。HR-Trainer のカメラから画像認識を行い、オブジェクトの有無を確かめ、オブジェクトのロボットに、もっとも近い点の座標が画面の下から高さの 1/3 までの領域にないとき、ロボットが 1 歩直進するようになる。もし、その領域にロボットとのもっとも近い点があれば、ロボットは旋回を行う。

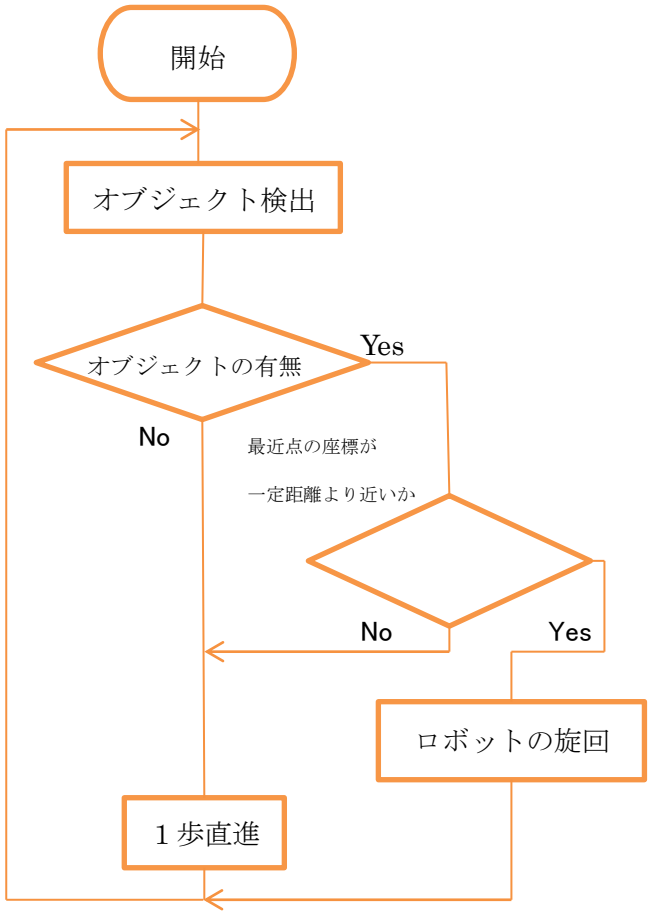


図 13　歩行ルーチンのフローチャート

4．研究のまとめ

（1）内容のまとめ

(a) ODE のサンプルプログラムを使い HR-Trainer のシミュレータと、各ステップの関節の角度を求める 2 足歩行プログラムを作成する。2 足歩行プログラムは、運動学と逆運動学を使い各関節の角度を求める。

作成した 2 足歩行プログラムを HR-Trainer 本体で使用できるように調整し、実装する。

（b）画像認識には多くの複雑なアルゴリズムを用いたプログラムが必要だが、OpenCV を用いることにより比較的簡単に画像認識を行うことができる。
（c）色の表現にもさまざまな方法があり、その方法ごとに、利点、欠点が存在する。したがって、その時々に応じて適切な色の表現方法を選択することが必要となる。

（2）感想

今回の課題研究では、普段無意識のうちにものを認識し、判断しているが、これをコンピュータで行うのはとても難しく、複雑であったので、生物の視覚機能のすごさを改めて実感した。

また、この研究を通して、今までに知らなかった色の表現方法や、物の輪郭とらえる方法、さらに、物を画像から抜き出す方法について多く学ぶことができた。今後の研究ではこれらの内容をさらに深めていきたいと思った。（石岡）

作業をしているとき、わからない数学や英語がよくあり、勉強の大切さを実感した。この課題研究をする前は、プログラムが苦手だったが、作業をしているうちに、プログラムに慣れることができた。（鶴峯）

（3）今後の課題

HR-Trainer 本体を前方に歩行させるプログラムの完成を目指す。そして、HR-Trainer のシミュレータを使い、旋回する歩行プログラムを作成し、本体に実装する。

今回は規定されたオブジェクトのみの検出であったがこれをどのようなオブジェクトでも検出を可能にするために、オブジェクトの識別器を用いたオブジェクトの検出も検討していきたいと思う。

参考文献

出村　公成
簡単！実践！ロボットシミュレーション
森北出版株式会社

出村　公成
demura.net:ロボットの開発と教育
http://demura.net/ode

Akira
画像処理ソリューション
http://imagingsolution.blog107.fc2.com/