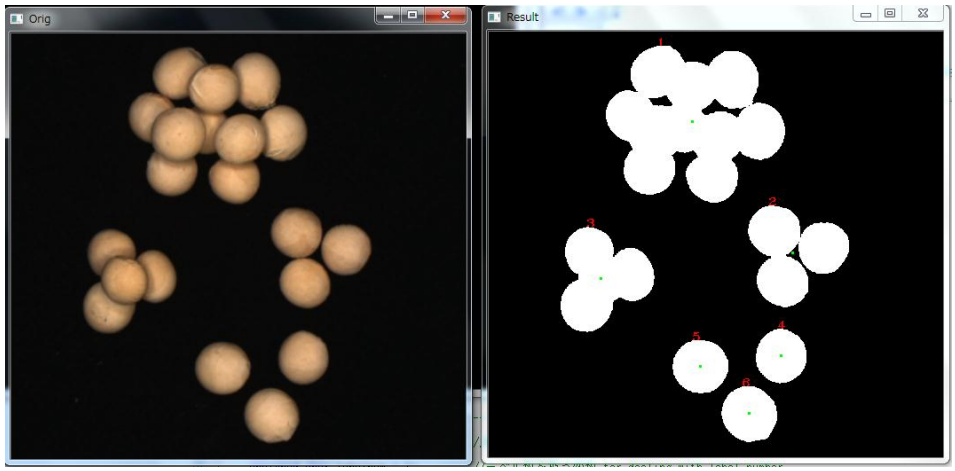
サンプルプログラムの概要

1. アルゴリズムを構築するための 4 枚のサンプル画像を次々と読み込む
2. RGB 分離を行い，R 画像を用いて閾値 40 で 2 値化
3. ラベリングを行う（ここで対象物の数を数えることになる）
4. ラベル付された対象の重心を計算
5. ラベル値と重心位置を 2 値画像に表示
   （赤い数字がラベル値，緑色の点が重心位置を表している）



6. テキストファイルに結果を書き出し
7. 画像とテキストファイルの保存
   保存先は C:¥Shiigi（実行前に作成したフォルダ）

もう少し詳しい説明

1. アルゴリズムを構築するための 4 枚のサンプル画像を次々と読み込む

ここで重要なことは画像を順番に読み込むための文字列操作。

for 文の番号 i を画像の番号として使用している。strcpy は文字列のコピー，sprinf は整数を文字列に変換，strcat は文字列を繋げる処理を行う。文字列操作の流れは，

```
//------画像の前名前と拡張子を変数に書き込む------------------------------//
//------write pre-name and extension of input image to variables ---------//
    strcpy(name1,"Sample");
    strcpy(name2,".bmp");

//------ファイルオープン処理------------------------------//
//------proccess to open file ---------//

    if((fp=fopen("C:¥¥Shiigi¥¥Result.txt","w")) == NULL){
        printf("file open error!!¥n");
        exit(1);
    }

//------画像を順次読み込むためのforループ----------------------------------//
//------"for roop" for sequential reading image---------------------------//
    for(i=0;i<4;i++){

//------画像を順次読み込むための名前変更----------------------------------//
//------change name for sequential reading image---------------------------//
        strcpy(in_name, "");
        sprintf(num,"%d",i);
        strcat(in_name, name1);
        strcat(in_name, num);
        strcat(in_name, name2);

//------画像の読み込み------------------------------------------------------//
//------read image----------------------------------------------------------//

        IplImage *Orig_image = cvLoadImage( in_name, CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_ANYCOLOR );
```

1) name1 に「Sample」をコピー
2) name2 に「.bmp」をコピー
3) in_name を初期化
4) num に i の整数値を文字列として代入
5) in_name に name1 を繋げる → in_name の中身：Sample
6) in_name に num を繋げる → in_name の中身：Sample0 （i が 0 のとき）
7) in_name に name2 を繋げる → in_name の中身：Sample0.bmp

こうすることで，画像を順番に読み込めるようになる。

for(i=0;i<4;i++)の 4 を 20 に変えれば，Sample0.bmp～Sample19.bmp の画像を順番に読み込める。

画像の読み込みは，cvLoadImage 関数（OpenCV）を利用している。

第 1 引数は画像の名前，第 2 引数はサンプルのように記入しておけば大抵の画像は扱ってくれる。IplImage 構造体の変数のポインタを返り値とする。

2．RGB 分離を行い，R 画像を用いて閾値 40 で 2 値化

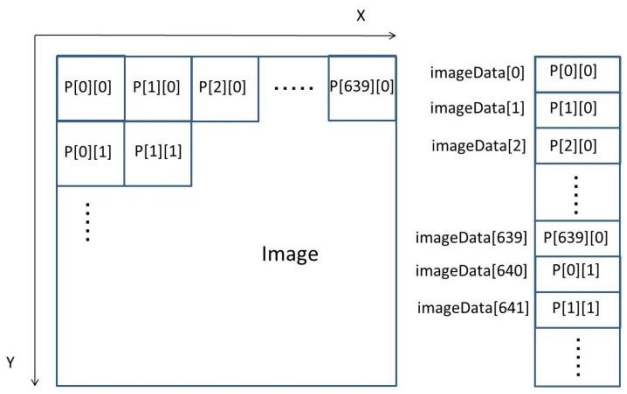
```
//------画像の生成-------------------------------------------------------------//
//------create image----------------------------------------------------------//

        IplImage *R_image = cvCreateImage ( cvGetSize(Orig_image), IPL_DEPTH_8U, 1 );           //Red
        IplImage *G_image = cvCreateImage ( cvGetSize(Orig_image), IPL_DEPTH_8U, 1 );           //Green
        IplImage *B_image = cvCreateImage ( cvGetSize(Orig_image), IPL_DEPTH_8U, 1 );           //Blue
        IplImage *Binary_image = cvCreateImage ( cvGetSize(Orig_image), IPL_DEPTH_8U, 1 );      //Binary
        IplImage *FinalLabel_image = cvCreateImage ( cvGetSize(Orig_image), IPL_DEPTH_8U, 1 );  //Label
        IplImage *Center_Cal_image = cvCreateImage ( cvGetSize(Orig_image), IPL_DEPTH_8U, 1 );  //Center
        IplImage *Result_image = cvCreateImage ( cvGetSize(Orig_image), IPL_DEPTH_8U, 3 );      //Result

//------画像の分離-------------------------------------------------------------//
//------split color image to RGB ----------------------------------------------//

        cvSplit( Orig_image, B_image, G_image, R_image, NULL );

//------2値化-----------------------------------------------------------------//
//------binarization ----------------------------------------------------------//

        for( y=0; y < Orig_image->height; y++){
            for ( x=0; x < Orig_image->width; x++){

                p_image = R_image->imageData[y * R_image->width + x];

                //2値化の閾値を40以上に設定
                //set more than 40 as threshhold value for binarization
                if(p_image>=40){
                    Binary_image->imageData[y * Binary_image->width + x] = (unsigned char) 255;
                }else{
                    Binary_image->imageData[y * Binary_image->width + x] = (unsigned char) 0;
                }
            }
        }
```

cvCreateImage 関数は画像の生成を行う。
第 1 引数：画像サイズ（cvGetSize により引数で渡された画像のサイズを返す），
第 2 引数：画像の bit 深度（IPL_DEPTH_8U は符号なし 8 bit），
第 3 引数：チャンネル数（グレー画像なら 1，カラー画像なら 3）。
IplImage 構造体のポインタを返り値とする。

cvSplit 関数で RGB 分離を行う。
第 1 引数：原画像の IplImage 構造体，第 2 引数：B 画像の IplImage 構造体，
第 3 引数：G 画像の IplImage 構造体，第 4 引数：R 画像の IplImage 構造体

IplImage 構造体のメンバ heigh は画像の高さ，width は画像の横幅，imageData は画像のデータ配列を示す。imageData の配列は，画像の左上が先頭，右下が最後になる。
2 値化の手順

1. R 画像のデータを変数 p_image 代入。
2. p_image の値が 40 以上であれば
   Binary_image の値を 255 に
   p_image の値が 40 未満であれば
   Binary_image の値を 0 に



3. ラベリングを行う

```
この先は最終的に2値化した結果をラベリング，重心位置計算，結果画像の保存＆表示，結果のテキスト出力に用いると便利です----//

最後に2値化した画像をラベリング用の画像にコピー-----------------------//
cvCopy(Binary_image, FinalLabel_image, NULL);
cvZero(Center_Cal_image);

ラベリング------------------------------------------------------------//
labeling ------------------------------------------------------------//

labelnum=1;
for(y=0;y<FinalLabel_image->height;y++){
    for(x=0;x<FinalLabel_image->width;x++){

        p_binari = FinalLabel_image->imageData[y * FinalLabel_image->width + x];
        if(p_binari == 255){
            //ラベリング関数
            //labeling function
            cvFloodFill( FinalLabel_image, cvPoint(x,y), cvScalarAll(labelnum), cvScalarAll(0), cvScalarAll(0), &comp[labelnum], 4, NULL );

            //面積50以下は除去
            //remove if are is less than 50
            if(comp[labelnum].area <= 50){
                cvFloodFill( FinalLabel_image, cvPoint(x,y), cvScalarAll(0), cvScalarAll(0), cvScalarAll(0), &comp[0], 4, NULL );
            }else{
                //もしラベル数が255以上になったとき画像上で表現できないためエラーとして扱う
                //deal with error when label number more than 255 because label number can not show on image
                if(labelnum == 255){
                    printf("limit¥n");
                    exit(1);
                }
                //show on command prompt
                //printf("labelnum = %d, area = %lf¥n",labelnum, comp[labelnum].area);
                labelnum ++;
            }
        }
    }
}
```

OpenCV の cvFloodFill 関数を用いてラベリングを行っている。
この関数は、指定した位置から始まる連結成分を指定した色で塗る。
第 1 引数：入力画像（IplImage 構造体），第 2 引数：連結成分の開始点
第 3 引数：塗りつぶす値（上記処理はラベル値で塗りつぶしている）
第 4 引数：連結するための条件（色の差の許容下限値：上記処理は下限値 0）
第 5 引数：連結するための条件（色の差の許容上限値：上記処理は上限値 0）
第 6 引数：CvConnectedComp 構造体（塗りつぶされた領域の情報：面積など）
第 7 引数：連結性（上記処理は 4 連結性を使用），第 8 引数：マスク

処理の流れ

1. FinalLavel_image 画像に 2 値画像（Binary_Image）をコピー
2. ラベル値（labelnum）を初期化（値は 1）
3. for ループへ
4. FinalLavel_image 画像の左上から値が 255 になっている位置を検索
5. 255 になっている位置を見つけた場合
    5－1. cvFloodFill 関数を用いてラベリング
        連結成分の開始点：255 を検索した位置
        塗りつぶす値：現在のラベル値（labelnum）
    5－2. ノイズ除去
        ラベリングされた領域の面積が 50 以下は、削除する

comp[labelnum].area : 現在のラベル値（labelnum）の面積を示す CvConnectedComp 構造体のメンバ

この値が 50 以下の場合 cvFloodFill 関数を用いて 0 の値でラベリング

5－3. ラベル値（labelnum）の更新

ラベリングされた領域の面積が 50 より大きければ、ラベル値（labelnum）に 1 を加える。

もし、ラベル値が 255 以上になった場合エラー処理を行う。8 bit の画像上で処理を行うことを前提としているため、この処理では 255 以上の物体の識別は不可能。

4. ラベル付された対象の重心を計算

cvMoments 関数および cvGetSpatialMoment 関数を用いてモーメントの計算を行い、重心位置を求める。

cvMoments

3 次までの空間モーメントあるいは中心モーメントを計算し，結果を第 2 引数に保存する.

第 1 引数：画像（IplImage 構造体）

第 2 引数：画像モーメントを表す構造体へのポインタ

第 3 引数：フラグ

cvGetSpatialMoment

以下のように画像モーメントから空間モーメントを計算する。

$M_{x\_order,\ y\_order} = Sum_{x,\ y}(I(x,y) \cdot x^{x\_order} \cdot y^{y\_order})$

ここで $I(x,y)$はピクセル (x,y)の強度（輝度）

処理の流れ

1. ラベル値ごとに 2 値画像（Center_Cal_image）を作成する
2. cvMoments 関数を用いて画像モーメント（CvMoments moments）を求める
3. 0 次モーメント（面積：m_00），x 方向 1 次・y 方向 0 次モーメント（m_10），x 方向 0 次・y 方向 1 次モーメント（m_01）を求める。
4. 重心（Gravity[ラベル値][x or y (x：0, y：1)]）の計算
5. cvZero 関数で Center_Cal_image を初期化（値を全て 0 にする）

```
//----------重心位置計算------------------------------------------------------//
//----------Calculate position of gravity-------------------------------------//
        fprintf(fp, "Lable, X, Y¥n");
        for(j=1;j<labelnum;j++){
            for(y=0;y<FinalLabel_image->height;y++){
                for(x=0;x<FinalLabel_image->width;x++){
                    p_binari = FinalLabel_image->imageData[y * FinalLabel_image->width + x];
                    if(p_binari==j){
                        Center_Cal_image->imageData[y * FinalLabel_image->width + x] = (unsigned char) 255;
                    }
                }
            }

            cvMoments(Center_Cal_image,&moments,0);
            m_00 = cvGetSpatialMoment(&moments,0,0);
            m_10 = cvGetSpatialMoment(&moments,1,0);
            m_01 = cvGetSpatialMoment(&moments,0,1);
            Gravity[j][0] = (int)(m_10/m_00);
            Gravity[j][1] = (int)(m_01/m_00);

            cvZero(Center_Cal_image);

            fprintf(fp, "%d, %d, %d¥n", j, Gravity[j][0], Gravity[j][1]);
        }

        fprintf(fp, "¥n");
```

5．ラベル値と重心位置を 2 値画像に表示

（赤い数字がラベル値，緑色の点が重心位置を表している）

ラベル値の表示

cvInitFont 関数

文字描画関数に渡されるフォント構造体を初期化する

第 1 引数：この関数で初期化されるフォント構造体（CvFont　font[0]）へのポインタ

第 2 引数：フォント名の識別子

第 3 引数：幅の比率、第 4 引数：高さの比率

cvPutText 関数

指定したフォントと色で文字列を画像中に描画する

第 1 引数：画像（IplImage 構造体）

第 2 引数：描画する文字（text）

第 3 引数：最初の文字の左上の座標

第 4 引数：フォント構造体へのポインタ（&font[0]）

第 5 引数：文字の色（CvScalar　color）

処理の流れ

1．cvInitFont 関数でフォントの設定

2．検索するラベル値の値（labelnumcounter）を初期化（1）

3．検索するラベル値を持つピクセルが見つかれば

　3－1．Sprintf 関数を用いて、ラベル値（labelnumcounter）を文字列（text）に変換

　3－2．　cvPutText 関数を用いて、Result_image 上に、text の文字を、x, y の位置に、font[0]および色 color で書き込む

　　色情報は、CvScalar 構造体 CvScalar color={0,0,255,0};で定義

　　CvScalar color={青,緑,赤,0}

　3－3．検索するラベル値の値（labelnumcounter）を更新（＋1）

4．ラベル画像から 2 値画像の作成

```
//------ラベル値の表示-----------------------------------------------------------//
//------show label number -------------------------------------------------------//

        //フォントの設定
        //set font
        cvInitFont(&font[0], CV_FONT_HERSHEY_COMPLEX_SMALL, 0.7,0.7);

        labelnumcounter=1;
        for(y=0;y<FinalLabel_image->height;y++){
            for(x=0;x<FinalLabel_image->width;x++){

                p_binari = FinalLabel_image->imageData[y * FinalLabel_image->width + x];
                if(p_binari > 0){
                    if(labelnumcounter==p_binari){
                        sprintf(text, "%d", labelnumcounter);
                        //ラベル数の書き込み
                        //write label number
                        cvPutText (Result_image, text, cvPoint (x,y), &font[0], color);
                        labelnumcounter++;
                    }
                    //2値化
                    //binarization
                    Result_image->imageData[(y * FinalLabel_image->width + x)*3] = (unsigned char) 255;
                    Result_image->imageData[(y * FinalLabel_image->width + x)*3 + 1] = (unsigned char) 255;
                    Result_image->imageData[(y * FinalLabel_image->width + x)*3 + 2] = (unsigned char) 255;
                }else{
                    Result_image->imageData[(y * FinalLabel_image->width + x)*3] = (unsigned char) 0;
                    Result_image->imageData[(y * FinalLabel_image->width + x)*3 + 1] = (unsigned char) 0;
                    Result_image->imageData[(y * FinalLabel_image->width + x)*3 + 2] = (unsigned char) 0;
                }
            }
        }
```

重心位置の表示

重心位置を中心に３ｘ３のエリアを緑色で表示

```
-----重心位置表示-------------------------------------------------------------//
-----show position of gravity-------------------------------------------------//
 for(j=1;j<labelnum;j++){
     for(y=-1;y<=1;y++){
         for(x=-1;x<=1;x++){
             Result_image->imageData[((Gravity[j][1]+y) * FinalLabel_image->width + (Gravity[j][0]+x))*3] = (unsigned char) 0;
             Result_image->imageData[((Gravity[j][1]+y) * FinalLabel_image->width + (Gravity[j][0]+x))*3 + 1] = (unsigned char) 255;
             Result_image->imageData[((Gravity[j][1]+y) * FinalLabel_image->width + (Gravity[j][0]+x))*3 + 2] = (unsigned char) 0;
         }
     }
 }
```

6. テキストファイルに結果を書き出し

重心位置計算前に、画像の名前、ラベル値（対象物の数）を書き込み

重心位置計算時に、ラベル値、重心ｘ座標、重心ｙ座標を書き込み

```
fprintf(fp, "Image file name, number of sample¥n");
fprintf(fp, "%s, %d¥n",in_name,labelnumcounter-1);

-----重心位置計算-------------------------------------------------------------//
-----Calculate position of gravity-------------------------------------------//
fprintf(fp, "Lable, X, Y¥n");
for(j=1;j<labelnum;j++){
    for(y=0;y<FinalLabel_image->height;y++){
        for(x=0;x<FinalLabel_image->width;x++){
            p_binari = FinalLabel_image->imageData[y * FinalLabel_image->width + x];
            if(p_binari==j){
                Center_Cal_image->imageData[y * FinalLabel_image->width + x] = (unsigned char) 255;
            }
        }
    }

    cvMoments(Center_Cal_image,&moments,0);
    m_00 = cvGetSpatialMoment(&moments,0,0);
    m_10 = cvGetSpatialMoment(&moments,1,0);
    m_01 = cvGetSpatialMoment(&moments,0,1);
    Gravity[j][0] = (int)(m_10/m_00);
    Gravity[j][1] = (int)(m_01/m_00);

    cvZero(Center_Cal_image);

    fprintf(fp, "%d, %d, %d¥n", j, Gravity[j][0], Gravity[j][1]);
}

fprintf(fp, "¥n");
```

7. 画像とテキストファイルの保存

画像の表示

cvNamedWindow 関数

window の作成を行う

第 1 引数：window の名前，第 2 引数：window のサイズ

cvShowImage 関数

画像を指定した window に表示する

第 1 引数：window の名前、第 2 引数：表示したい画像の IplImage 構造体

cvWaitKey 関数

プログラムの 1 時停止

第 1 引数：遅延時間（msec）0 の場合 key の入力待ち

cvDestroyWindow 関数

window の破棄

第 1 引数：window の名前

画像の保存

cvSaveImage 関数

画像の保存

第 1 引数：保存名，

第 2 引数：保存する画像の IplImage 構造体

後処理（画像メモリーの解放）

cvReleaseImage 関数

画像メモリーの解放を行う

第 1 引数：解放したい画像の IplImage 構造体のポインタ

```
//window の作成
//create window
cvNamedWindow( "Orig", CV_WINDOW_AUTOSIZE );
cvNamedWindow( "Result", CV_WINDOW_AUTOSIZE );

//画像の表示
//show image
cvShowImage( "Orig", Orig_image );
cvShowImage( "Result", Result_image );

//wait input key
cvWaitKey(0);

//画像の保存
//save image
strcpy(out_name,"C:¥¥Shiigi¥¥Result");
strcat(out_name, num);
strcat(out_name, name2);
cvSaveImage(out_name, Result_image );

//windowの破棄
//destroy window
cvDestroyWindow( "Orig" );
cvDestroyWindow( "Result" );

//メモリーの解放
//release memory
cvReleaseImage( &Orig_image );
cvReleaseImage( &R_image );
cvReleaseImage( &G_image );
cvReleaseImage( &B_image );
cvReleaseImage( &Binary_image );
cvReleaseImage( &FinalLabel_image );
cvReleaseImage( &Center_Cal_image );
cvReleaseImage( &Result_image );
```