ライブラリを使ってプログラミングの手間を省こう

OpenCVを利用した動画像処理プログラミング

中編

竹田 大祐 / 高鹿 陽介

さて，前回に続きOpenCVについて解説します．前回は，カメラとの接続と，カメラからの画像に図形を描画するところまで説明しました．今回はそれらを利用しつつ，さらに発展させて，テンプレート・マッチングによる図形認識について述べていきます．（筆者）

1 テンプレート・マッチングの概要

OpenCVには，画像処理アルゴリズムを実装するための関数も用意されています．今回は，そのなかで，テンプレート・マッチングによる画像認識を行ってみます．

テンプレート・マッチングは，認識しようとする対象の画像をテンプレートとしてもち，対象画像の各部分との類似度を計算することによって対象物の画像中の位置を求める処理です．類似度の計算には，相互相関などを用いることが一般的です．

テンプレート・マッチングは，画像認識としては基本的な処理です．OpenCVにはテンプレート・マッチングを行う関数が用意されているので，カメラの画像中から目標物を検出するというような処理が簡単に実現できます．

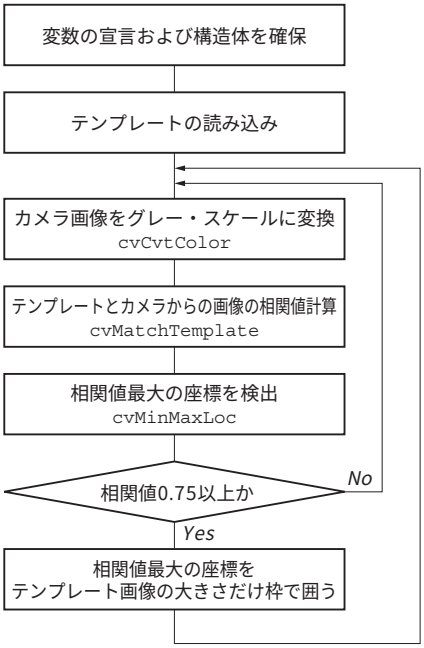


図1 テンプレート・マッチング・プログラムのフローチャート

図2 テンプレート画像

2 テンプレート・マッチングのためのプログラム── cvMatchTemplate関数を使用

● テンプレート・マッチング・プログラムの概要

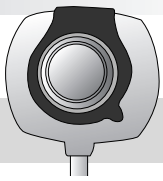
リスト1（sample3.c）に示すテンプレート・マッチングのプログラムは，カメラから取り込んだ画像中の目標物を検出し，四角い枠で囲うというものです．枠で囲う処理には，前編（2006年11月号，pp.186-192）の図形描写の例で用いた関数を使います．

テンプレート・マッチングに使用した関数は，cvMatchTemplate関数です．この関数は，テンプレート画像と入力画像の類似度を計算し，結果を指定した配列に出力する関数です．

テンプレート・マッチングとひと口にいっても，類似度の計算にはいくつかの方法があります．代表的な手法として2乗残差法と相互相関法が挙げられますが，今回は相互相関法を用いました．こちらの手法のほうが計算時間はかかりますが，より

図3 テンプレート・マッチング・プログラムの出力結果

KEYWORD──動画像処理，OpenCV，テンプレート・マッチング，相互関数，画像認識，Makefile，USBカメラ，色空間，グレー・スケール，正規化



確実な検出が行えるとされています．cvMatchTemplate関数もこの二つの手法を提供しています．

● プログラムの流れ

プログラムのフローチャートを**図1**に示します．簡単に説明すると，

(1) テンプレート画像の読み込みを行う
(2) カメラからの画像との相関値を計算する
(3) 相関値のいちばん高い座標を検索し，その場所に四角い枠を描画する

というものです．ただし，認識対象が画像中にない場合，相関値は低い値になります．この状態で値のいちばん高い部分を枠で囲っても意味がないので，しきい値を設定します．なお，今回は0.75に設定しました．

● プログラムの実行

テンプレート画像には，**図2**のような簡単な図形を用いました．もちろん，これ以外の図形でもかまいません．

リスト2のMakefileを実行してみましょう．**図3**のようにテンプレート画像が認識され，赤い枠が表示されれば成功です．

3 テンプレート・マッチングのための関数

ここで使用している関数について解説します．なお，これらの関数もすべてdocsフォルダ以下のリファレンス中に説明があります．

リスト1　テンプレート・マッチング・プログラム(sample3.c)

```
/* OpenCV 動画のテンプレート・マッチング */

#include <stdio.h>

//OpenCV用のヘッダ・ファイル
#include <cv.h>
#include <highgui.h>

int main(void){

  int key_input;                //キー入力
  CvCapture* camera   = NULL;  //カメラ・デバイス

  IplImage *img  = NULL;  //画像データ格納用構造体
  IplImage *grey = NULL;  //処理用画像データ
                          //(取り込み画像のグレー・スケール)
  IplImage *temp = NULL;  //テンプレート用画像格納構造体
  IplImage *dst  = NULL;  //相関情報格納用構造体

  double max_interlinkage=0;  //相関値の最大値を代入する変数
  double min_interlinkage=0;  //相関値の最小値を代入する変数

  CvPoint max_point;    //相関値の最大値の座標を代入する変数
  CvPoint min_point;    //相関値の最小値の座標を代入する変数
  CvPoint corner_point; //ターゲットのもう一方の四隅座標

  //カメラ・デバイスの先頭ポインタを取得
  camera=cvCaptureFromCAM(-1);

  //カメラ・デバイスが見つからないときの処理
  if(camera==NULL){
      printf("Cannot Open Camera Device!¥n");
      return(0);
  }

  //テンプレート画像の読み込み(グレー・スケール)
  temp=cvLoadImage("cup.bmp",0);

  //Windowの生成
  cvNamedWindow("Show", CV_WINDOW_AUTOSIZE);

  //カメラ・デバイスから画像を取得
  img=cvQueryFrame(camera);

  //テンプレート・マッチングに用いる相関値データを格納する画像の領域確保
  //グレー・スケール画像用に領域確保
  grey=cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,1);
  dst =cvCreateImage(cvSize(img->width-temp->width+1, img-
>height-temp->height+1),IPL_DEPTH_32F,1);

  //動画キャプチャと表示のループの開始
  for(;;){

    //カメラ・デバイスから画像を取得
    img=cvQueryFrame(camera);

    //画像が取得できなかったときのエラー処理
    if(img==NULL){
      printf("Cannot Open Video Data!¥n");
      break;
    }


    //グレー・スケールに変換して格納
    cvCvtColor(img,grey, CV_BGR2GRAY);

    //テンプレート・マッチングのための相関値の計算
    cvMatchTemplate(grey,temp,dst,CV_TM_CCOEFF_NORMED);

    //テンプレート・マッチングでいちばんマッチしている部分を検索
    cvMinMaxLoc(dst,&min_interlinkage,&max_interlinkage,
                                  &min_point,&max_point,NULL);

   //相関値0.75以下ならばLostを表示して次に行く
    if(max_interlinkage>0.75){
      //座標を見やすく代入(直接代入でもよい)
      corner_point=cvPoint(max_point.x+temp-
                          >width,max_point.y+temp->height);

      //マッチング箇所を四角で描画
      cvRectangle(img,max_point,corner_point,CV_RGB
                                              (255,0,0),2);
      printf("Detection¥n");
    }
    else{
      printf("Lost¥n");
    }

    //画像表示
    cvShowImage("Show",img);

    //キー入力
    key_input = cvWaitKey(10);

    //ESCで終了
    if(key_input==0x1b){
      printf("ESC KEY Quit¥n");
      break;
    }
  }

  //メモリ開放
  cvReleaseCapture(&camera);
  cvDestroyWindow("Show");

  return(0);
```