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

後編

竹田 大祐 / 高鹿 陽介

前回（2006年12月号，pp.142-144），前々回（2006年11月号，pp.186-192）に引き続き，OpenCVについて解説します．最終回となる今回は，オプティカル・フローを用いた動体検出のプログラムについて述べます．OpenCVを用いると，わずか50行ほどのコードでオプティカル・フローのプログラムを書くことができます．（筆者）

1 動体検出のための手法 ――オプティカル・フロー

オプティカル・フローとは，映っている対象の動きを時系列画像から検出する手法の一つです．いくつかの計算手法がありますが，主なものは，画面を構成する画素のもつ輝度値が微小時間後（実際は次のフレーム）にその輝度値を変えず，座標のみが変化したと考えて，画素の速度ベクトルを求める式を立てるという手法です．そのため，オプティカル・フローの計算結果は画像を構成する各画素の速度ベクトルの形になります．

2 オプティカル・フローのプログラムも簡単に書ける

OpenCVには，オプティカル・フローの処理を行う関数も用意されています．この関数を利用すると，オプティカル・フローを用いた動体検出プログラムを簡単に作成することができます．**リスト1**（sample4.c）のプログラムは，USBカメラで読み込んだ画像から速度ベクトルを計算し，線分にして元の画像に重ね合わせて表示するものです．画像の動きのある部分に線分が表示されることになります．

プログラムの主な流れを**図1**に示します．このプログラム（**リスト1**）は，保存した1フレーム前の画像と，取り込んだ現在の画像との2枚の画像からOpenCVの関数を用いて速度ベクトルを計算します．その後，速度ベクトルを線分にして画面上に描写し，カメラ画像に重ね合わせて表示させます．

OpenCVにはオプティカル・フロー関数が複数用意されています．ここで用いた手法は，『Lucas & Kanade法』と呼ばれるものです．そのほかの手法については，OpenCVのマニュアル（opencvman_old.pdf）の2-18項「Optical Flow」，またはドキュメント（¥docs¥ref¥opencvref_cv.htm）を参照してください．

それでは，今回使用した関数について解説します．

▶ cvCalcOpticalFlowLK

オプティカル・フロー関数

書式：cvCalcOpticalFlowLK(1フレーム前の画像, 現在の画像, ウィンドウ・サイズ, 速度ベクトルのx成分格納用変数, 速度ベクトルのy成分格納用変数)；

この関数は，1フレーム前の画像と現在の画像の2枚の時系列画像から，画素のもつ速度ベクトルを計算する関数です．入力する画像は8ビット画像，つまりグレー・スケール画像でなければなりません．また，このLucas & Kanade法は，近接する画素は同じ速度ベクトルをもつものとして，グループ化して計算することで計算量を減らしています．

この手法を用いて速度ベクトルを求める過程で，この関数は，ガウス関数を窓関数として用いています．第3引き数でcvSize関数を使用して指定しているのは，窓関数のウィンドウ・サイズです．詳しい説明は行いませんが，11×11程度のサイズを指定しておくと，うまくいくようです．

この関数は，計算した速度ベクトルの値を画像と同じIplimage型で宣言された変数に格納します．つまり，画像の輝度値の代わりに速度成分が格納されることになります．

図1 オプティカル・フローのプログラム（リスト1）の処理手順

KEYWORD――動画像処理，OpenCV，オプティカル・フロー，Lucas & Kanada法，make，Makefile，USBカメラ，x86

リスト1　オプティカル・フローのプログラム(sample4.c)

```
#include <stdio.h>
#include <cv.h>
#include <highgui.h>

int main(void){

  CvCapture* camera   = NULL; // 画像処理に必要なIplImageなどの宣言
  IplImage *imgA      = NULL; // 生画像
  IplImage *grayA     = NULL; // グレー・スケール画像
  IplImage *grayA_old = NULL; // グレー・スケール画像(1フレーム前)
  IplImage *velx      = NULL; // オプティカル・フロー計算結果X成分格納
  IplImage *vely      = NULL; // オプティカル・フロー計算結果Y成分格納

  int key;                    // キー入力格納
  int flag=0;                 // キャプチャ枚数のフラグ格納

  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(-1);  // エラー終了
  }

  // 1枚画像を取得し，imgAのさまざまな情報を取得
  imgA = cvQueryFrame(camera);

  // 必要な画像領域の生成
  grayA    =cvCreateImage(cvGetSize(imgA),IPL_DEPTH_8U,1);
  grayA_old=cvCreateImage(cvGetSize(imgA),IPL_DEPTH_8U,1);
  velx     =cvCreateImage(cvGetSize(imgA),32,1);
  vely     =cvCreateImage(cvGetSize(imgA),32,1);

  // ウィンドウの生成
  cvNamedWindow("OpticalFlow",CV_WINDOW_AUTOSIZE);

  for(;;){

    imgA = cvQueryFrame(camera);         // 動画像の取得
    grayA->origin     = imgA->origin;    // キャプチャ画像(new)の
                                            原点情報(左下)にコピー
    grayA_old->origin = imgA->origin;    // キャプチャ画像(old)の
                                            原点情報(左下)にコピー
    cvCvtColor(imgA,grayA,CV_BGR2GRAY); // キャプチャ画像を
                                            グレー・スケールに変換する

    // 1回目の画像キャプチャ時の処理
    if(flag==0){
      flag=1;
      cvCopy(grayA,grayA_old,NULL);
    }
    // 2回目以降の画像キャプチャ時の処理
    else{

      // オプティカル・フローの計算

cvCalcOpticalFlowLK(grayA_old,grayA,cvSize(11,11),velx,vely);

      // オプティカル・フローの値をキャプチャした画像に描画する(線)
      for (int i=0;i<imgA->height;i+=20){
        for (int j=0;j<imgA->width;j+=20){
          int dx = (int)cvGetReal2D(velx,i,j);
          int dy = (int)cvGetReal2D(vely,i,j);
          cvLine(imgA,cvPoint(j,i),cvPoint(j+dx,i+dy),
                                    CV_RGB(255,255,255),1,8,0);
        }
      }
    }

    cvShowImage("OpticalFlow",imgA); // キャプチャした画像の表示
    cvCopy(grayA,grayA_old,NULL);    // キャプチャした画像(old)を保存

    key = cvWaitKey(10);             // キー入力
    if(key==0x1b){                   // ESCで終了
      printf("ESC key Quit\n");
      break;
    }
  }

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

  return(0);  // 正常終了
}
```

リスト2
Makefile

```
CC  = g++ -O -o
CC_OPT   = -Wall
CV_LIB   = -L/usr/local/lib -L/usr/lib
CV_INC   = -I/usr/local/include/opencv -I/usr/include
CV_OPT   = -lcxcore -lhighgui -lcv -lcvaux
TARGET   = sample4.c
OBJECT   = sample4
all:
    $(CC)  $(OBJECT) $(TARGET) $(CV_INC) $(CV_LIB) $(CV_OPT) $(CC_OPT)
clean:
    rm *~ *.o $(OBJECT) core sample.exe
```

▶ **cvGetReal2D**

配列の値を取り出す関数

書式：cvGetReal2D(配列,インデックス1,インデックス2)

この関数は，指定した配列の要素を出力する配列です．計算した各画素のもつ速度ベクトルを描画するために使用しています．

▶ **cvCopy**

配列のコピーを行う関数

書式：cvCopy(コピー元名，コピー先名，NULL)

この関数は，主にIplImage形式の配列をコピーするために用います．

3　オプティカル・フローのプログラムを実行してみよう

それでは，オプティカル・フローのプログラム(**リスト1**)を実行してみましょう．**リスト1**のプログラムを，**リスト2**のMakefileを使ってmakeしてください．**図2**のように出力されれば成功です．

ところで，今回は実行結果を簡単に線分で表示したのですが，これでは動きの方向がわかりません．そこで，**リスト3**のような関数を追加すると，**図3**のように表示されます．