OpenCV Library の使い方

加藤 丈和 †

† 和歌山大学 システム工学部 情報通信システム学科

1 はじめに

OpenCV は，Intel が開発し，BSD ライセンスで公開している画像処理，コンピュータビジョンのためのライブラリで，現在は SouceForge で配布されている．

OpenCV では，一般に良く使われる RGB 各 8bit のカラー画像や 8bit の濃淡画像だけでなく，各画素が 8bit 以上のダイナミックレンジの広い画像，マイナスの値や実数値を持つ画像を扱うことができる．また画像のある領域だけを処理対象とする ROI (Region of Interest) や，カラー画像のある色成分だけを処理対象とする COI (Channel of Interest) を指定することができ，画像処理を効率良く行うプログラムを書くことができる．

OpenCV には，300 以上の関数が含まれているが，これらは大きく次の 4 つに分けることができる．

CXCORE OpenCV で扱うデータ構造の型の定義，及び基本的な画像の操作．画像同士の四則演算や画像のコピーなど．

CV 様々な画像処理やコンピュータビジョンアルゴリズムを実装した関数．これらの関数はさらに下記の 5 つの分野に分れている．

Image Processing
: 基本的な画像処理

Structural Analysis
: 輪郭や矩形領域の抽出などの構造解析

Motion Analysis and Object Tracking
: 動作解析，対象追跡

Pattern Recognition
: パターン認識

Camera Calibration and 3D Reconstruction
: カメラ校正と 3 次元復元

MLL Machine Learning Library 機械学習，パターン認識などの関数．

HighGUI 簡単な GUI 及び，ファイルの読み書き，画像キャプチャ等のユーティリティ関数．

本稿では，これらの関数のうち，ごく一部のみを紹介する．それ以外の関数の詳細は配布パッケージに含まれるドキュメントを参照されたい．

OpenCV はマルチプラットホームのライブラリなので，一部の機能を除いてほとんどの環境で使用可能であるが，特に本稿では， Linux と Windows をターゲットとして解説する．なお，Linux 環境では，ディストリビューションは Fedora Core 5 ，コンパイラは gcc-4.0 で動作確認を行ない，Windows 環境では，WindowsXP SP2，コンパイラは Visual C++ 6.0 で動作確認を行った．

2 OpenCV のインストール

本章では，Linux と Windows の場合について，それぞれ OpenCV のインストール方法を説明する．

2.1 Linux へのインストール

Linux へのインストール方法について説明する．動作確認は Fedora Core 4 で行ったが一般的な Linux ディストリビューションであれば，ほぼ同様の手順でインストールできる．

2.1.1 準備

まず，OpenCV が必要とするライブラリのインストールを行う．OpenCV のインストールには，gtk+ 2.x, libjpeg, zlib, libpng, libtiff, ffmpeg, libraw1394, libdc1394, v4l, などのライブラリが必要である．このうち ffmpeg と libdc1394 以外は，Fedora のパッケージに用意されているので，大抵の場合にはデフォルトでインストールされているはずである．インストールされていない場合には，適宜 yum を使ってパッケージからインストールを行って欲しい．

ffmpeg は，MPEG のエンコード，デコードのためのライブラリであり，MPEG 等の動画を読み書きするのに必要である．ffmpeg は Fedora Core 4 用のパッケージが http://stentz.freshrpms.net/で配布されているので，こちらからパッケージを探してインストールすると良い．また，ソースコードも
`http://ffmpeg.sourceforge.net/index.php`
から入手できる．なお，本稿の動作確認は 0.4.9pre1 で行った．

```
% tar zxvf ffmpeg-0.4.9pre1.tar.gz
% cd ffmpeg-0.4.9pre1
% ./configure
% make
% su
# make install
```

libdc1394は，IEEE1394インタフェースにつながるDCAM規格のカメラを扱うためのライブラリであり，DCAM規格のカメラからの画像の取り込みを行う場合に必要である．libdc1394は

```
http://sourceforge.net/projects/libdc1394/
```

から入手でき，以下の手順でインストールできる．なお，本稿では1.1.0で動作確認を行った．

```
% tar zxvf libdc1394-1.1.0.tar.gz
% cd libdc1394
% ./configure
% make
% su
# make install
```

2.1.2 OpenCVのインストール

OpenCVのソースは，

```
http://sourceforge.net/projects/opencvlibrary/
```

で配布されているので，ここからopencv-linuxをダウンロードすれば良い．原稿執筆時のLinux版の最新バージョンは1.0rc1(0.9.9)である．

```
% tar zxvf opencv-0.9.9.tar.gz
% cd opencv-0.9.9
% ./configure --with-apps
% make
% su
# make install
```

2.2 Windowsへのインストール

Windows用には，インストールパッケージが用意され，必要なファイルが全て含まれているので，非常に簡単にインストールできる．なお，Windows版の最新バージョンは1.0rc1である．SourceFourgeのサイトからopencv-winをダウンロードすると，OpenCV_1.0rc1.exeというファイルがダウンロードされるので，あとはこれをダブルクリックし，幾つかの質問に答えると自動でインストールされる．

2.3 OpenCVのマニュアル，及び参考情報

OpenCVのリファレンスマニュアルは，Linux版では`/usr/local/share/opencv/doc/index.htm`，Windows版では，`C:\Program Files\OpenCV\docs\index.htm`にインストールされ，ブラウザで読むことができる．また，最後にリンクされているPDF版のマニュアルにはより詳しい手法の解説や，参考文献が記載されているので，手法を良く知りたい場合には有用である．ただし，このPDF版のマニュアルはバージョンが古いので，注意が必要である．

インターネット上のOpenCVの情報としては，まず開発元のIntelのオフィシャルページ [Intel]には，OpenCVの概要やコーディングガイド，FAQなどが公開されている．またYahoo GroupsのOpenCVのグループ [Yahoo]ではOpenCVに関する質問や，議論がされている．

日本語の情報としては，バージョンは若干古いが栗田氏による解説 [栗田] が網羅的で分りやすい．また，Bando氏のサイト [Bando] やkenta-ta氏のサイト [kenta-ta]では，いくつかの関数についてのリファレンスの日本語訳が公開されている．

Windows環境固有の情報としては，cygwinを使ってライブラリをソースからビルドする方法を，木村氏 [木村] が詳しく解説している．また，cygwin環境でビルドしたライブラリでUSBカメラを使う方法について，西口氏 [西口] が解説している．熊本大学の内村・胡研究室 [熊本大学] では，まだ更新中であるが，`Visual-C++`を使う場合のOpenCVのインストールや使い方について解説している．

Linux環境固有の情報としては，吉本氏のサイト [Yoshimoto]で，Linux環境でのOpenCVのインストールから，プログラムの書き方，コンパイルの仕方，IEEE1394で接続するDCAM規格のカメラの使い方まで詳しく解説されている．また，Yonekura氏のサイト [Yonekura]では，スクリプト言語のRubyからOpenCVを使うためのライブラリの配布と，その使い方について解説している．

3 サンプルプログラム

OpenCVには，そのまますぐ使えるサンプルプログラムがいくつか添付されている．本章では，それらについてコンパイルの仕方と使い方を解説する．

3.1 サンプルプログラムのビルド

Linux版では，サンプルプログラムは`/usr/local/share/opencv/samples`にインストールされるので，これを自分のホームディレクトリ等にコピーすると良い．ビルド方法はbuild.shを使うと全てのサンプルプログラムをまとめてビルドできる．

```
% cp -r /usr/local/shared/opencv/samples ./
% cd samples/c
% export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
% sh build.sh
```

3行目は，ライブラリやヘッダファイルの場所を自動で検索するpkg-configというツールのための環境変数の設定であり，自分が使っているシェルがbashなどのsh系の場合はこのままでいいが，tcshなどcsh系の場合は，この行を次のように変更する．

```
% setenv PKG_CONFIG_PATH /usr/local/lib/pkgconfig
```

また，pkg-configのないシステムを使用している

人は，build.sh を次のように書き換える必要がある．

```
#!/bin/sh
for i in *.c; do
    echo "compiling $i"
    g++ -O3 -I/usr/local/include/opencv -o \
`basename $i .c` $i -L/usr/local/lib -lcxcore \
-lcv -lhighgui -lcvaux;
done
```

Windows の場合は，
`C:\Program Files\OpenCV\samples\c` にあらかじめコンパイルされたサンプルプログラムとそのソースファイルがインストールされている．

3.2 サンプルプログラムの使い方

OpenCV には，1 に挙げる 20 種類のサンプルプログラムが添付されている．どれも数 10〜200 行程度の短いプログラムなので，自分でプログラムを作成するときの参考にして欲しい．また，いくつかはそのままでも十分実用的なプログラムである．

20 個のサンプルのうち，camshiftdemo, facedetect, laplace, lkdemo, motempl は，動画を対象としたプログラムであり，カメラを繋いでリアルタイムに処理を行なったり，AVI ファイルなどを読み込ませて処理させることができる．

ここでは，特徴点追跡のプログラム lkdemo と，顔検出のプログラム facedetect を例に挙げて使い方を説明する．これらのプログラムでは，カメラを使うことができる．使用できるカメラ，キャプチャーカードは，Linux の場合では Video for Linux に対応するキャプチャーカード，もしくは，IEEE1394 経由でキャプチャーする DCAM 対応のカメラである．Windows の場合は，Video for Windows (VFW) 対応のカメラ，キャプチャーカ ードを使用できる．カメラ，キャプチャーカードの設定方法については，ここでは割愛する．使用できるカメラを持っていない場合は，AVI ファイルを読み込ませて処理させることもできる．

3.2.1 lkdemo の実行

lkdemo は，LK 法と呼ばれる Lucas と Kanade [Lucas 81] によって提案された特徴点追跡のアルゴリズムを実装したものである．LK 法はステレオマッチング，Structure from motion，オプティカルフローの計算，ロボットの動物体検出など様々な場面で利用されている．

lkdemo の実行方法は，カメラから画像を取り込む場合は引数なしで ./lkdemo と実行すれば良い．Windows の場合では lkdemo.exe をダブルクリックしてもいい．AVI ファイルから画像を読み込む場合は，AVI ファイルを引数として `./lkdemo hoge.avi` として実行する．Windows でもコマンドプロンプトを開いて同様に実行できる．

表 1: OpenCV に添付されているサンプルプログラム

プログラム名	アルゴリズム
camshiftdemo	CAMSHIFT [Bradski 98] による対象追跡
contours	二値画像の輪郭検出
convexhull	凸包の計算
delaunay	ドロネーグラフの計算
demhist	ヒストグラム変換
distrans	距離変換
drawing	図形の描画
edge	Canny エッジ抽出 [Canny 86]
facedetect	Cascade 型識別器と Adaboost [Viola 01] による顔検出
ffilldemo	領域の塗りつぶし
fitellipse	楕円フィッティング
kalman	カルマンフィルタ
kmeans	K-means クラスタリング
laplace	画像の空間微分
lkdemo	LK 法 [Lucas 81] による特徴点追跡
minarea	外接矩形の計算
morphology	Mathmatical Morphology
motempl	モーションテンプレート
pyramid_segmentation	領域分割
squares	長方形検出

lkdemo を実行すると，ウィンドウが開き，取り込んだ画像が表示される．この画像上でマウスの左ボタンをクリックすると，特徴点を追加，削除することができ，追加された特徴点の追跡を開始する．登録された，緑色の点で表わされる (1)．また，キーボードから「r」をタイプすると自動で追跡しやすい特徴点を検出し，追跡を開始する.「c」をタイプすると全ての特徴点を消去する．ESC キーで終了できる．

図 1: lkdemo の実行画面 (カラーページ参照)

3.2.2 facedetect の実行

facedetect は，Viola と Michael [Viola 01] が提案した，Adaboost によって学習した，Cascade 型の識別器によって，顔を検出するアルゴリズムを実装したプログラムである．あらかじめ顔画像と非顔画像のトレーニングデータを与えて学習した識別器を用いて，入力画像から顔画像を検出するアルゴリズムであり高速，かつ安定に動作することから，顔検出，認識の分野では近年注目されている手法である．

facedetect では，まず顔画像と非顔画像を学習する必要があるが，ここではその方法は割愛し，OpenCV に添付されている，あらかじめ学習した識別器の情報を用いる方法を紹介する．学習方法について興味のある人は，ソースファイルの `apps/haartraining/doc/haartraining.htm` に学習方法のマニュアルがあるので，そちらを参考にされたい．

facedetect を実行するには，識別器の情報を指定する必要がある．識別器の情報は Linux 版では `/usr/local/share/opencv/haarcascades`, Windows では `C:\Program Files\OpenCV\data\haarcascades` に置いてあるので，この中の XML ファイルのうち好きなファイルを指定すると良い．例えば，正面顔を学習した haarcascade_frontalface_alt.xml を指定するには以下のように実行する．

```
% ./facedetect --cascade="/usr/local/share/opencv\
/haarcascades/haarcascade_frontal_alt.xml"
```

Windows の場合には，facedetect.cmd という名前でオプションを付けて実行するバッチファイルが用意してあるので，これを適宜書換えてダブルクリックして実行すれば良い．なお，カメラから画像を取り込む変りに AVI ファイルから読み込むには，lkdemo と同様に，最後にファイル名を指定すれば良い．

facedetect を実行すると，lkdemo と同様にウィンドウが開き入力画像が表示され，2 のように，顔を検出すると赤い枠で表示される．この図のように明さや顔の大きさ，多少の顔の向きが異なっていても検出することができる．

図 2: facedetect の実行画面 (カラーページ参照)

4 OpenCV プログラミングの基礎

本章では，OpenCV によるプログラミングの基礎として，OpenCV での画像の扱い方とコンピュータビジョンアルゴリズムを実装した関数の使い方について解説する．

```
#include <stdio.h>
#include <stdlib.h>
// OpenCV のヘッダファイル
#include <cv.h>
#include <highgui.h>

int main(int argc, char **argv){
  IplImage *image; //画像を扱う構造体
  char *pixel;     // 画素のポインタ
  int x,y,c;       //ループカウンタ

  //引数の個数のチェック
  if(argc<3) {
    fprintf(stderr,"usage:%s input output\n",argv[0]);
    exit(1);
  }

  // 画像ファイルの読み込み
  image = cvLoadImage(argv[1],1);
  // 画像を処理する
  for(y=0;y<image->height;y++)
    for(x=0;x<image->width;x++){

      // (x,y) の座標の画素のアドレスを計算
      pixel = image->imageData
        + y*image->widthStep
        + x*image->nChannels;

      // 各色成分に対して処理
      for(c=0;c<image->nChannels;c++)
        // 各色を反転
        pixel[c] = 255-pixel[c];

    }

  // 処理した画像をファイルに書き込む
  cvSaveImage(argv[2],image);
  //画像の解放
  cvReleaseImage(&image);
  return 0;
}
```

図 3: 画像のネガ反転のプログラム (Example1.c)

まずは，3 のプログラムを見て欲しい．このプログラムは，画像をファイルから読み込み，各色成分を反転して，ネガ画像を作成し，ファイルに書き込むプログラムである．このプログラムは Linux でも Windows でもコンパイル，実行が可能である．

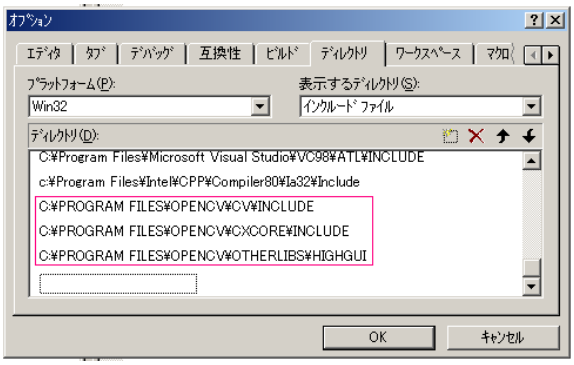
4.1 プログラムのコンパイルと実行

Linux では以下のようにコンパルできる．

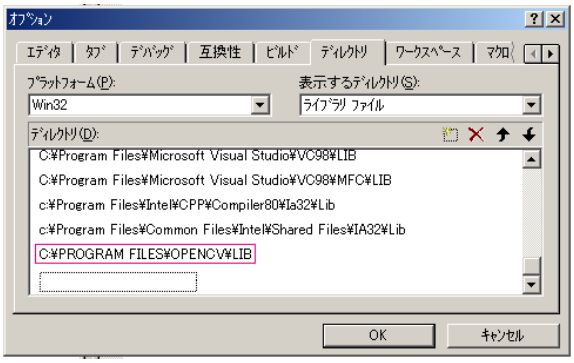
```
% g++  `pkg-config --cflags opencv` Example1.c \
-o Example1 `pkg-config --libs opencv`
```

pkg-config のないシステムでは，インクルードファイルのディレクトリとライブラリを指定するように適宜書き換えて欲しい．

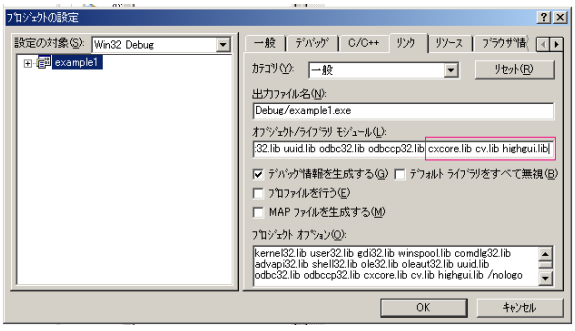
Windows の場合は，Visual C++ で Win32 Console Application としてプロジェクトを作成し，Example1.c をプロジェクトへ追加すればよい．ビルドするときには,「ツール」メニューの「オプション」を選び「ディレクトリ」タブでインクルードファイルとライブラリに OpenCV のディレクトリを加え (4(a),(b)),「プロジェクト」の「設定」の「リンク」タブで cxcore.lib, cv.lib, highgui.lib を加えておく (4(c)).



(a) ヘッダファイルのパス



(b) ライブラリのパス



(c) ライブラリ

図 4: Visual C++の設定

Example1 は，入力画像と出力画像のファイル名を指定して実行する．入出力の画像のフォーマットは BMP, JPEG, PNG, PNM などに対応している．例として，OpenCV のサンプルプログラムについてきた lena.jpg を入力とした場合の実行方法は以下の通りである．

```
% ./Example1 lena.jpg lena-nega.jpg
```

(a) 入力画像　(b) 出力画像

図 5: Example1 の実行結果 (カラーページ参照)

4.2 プログラムの解説

4.2.1 OpenCV のヘッダファイル

3 のプログラムを順に解説していく．まずは，最初のヘッダファイルのインクルードを見て欲しい．

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

cv.h には，OpenCV の関数の宣言や，各種構造体の定義などが収められている．OpenCV ライブラリを使ったプログラムを書くには，必ずこのファイルをインクルードする．また次の highgui.h は，OpenCV に附属の GUI ツールキットや，画像の保存，読み込みなどの各種ユーティリティ関数の宣言が含まれている．Example1.c では，cvSaveImage, cvLoadImage を使うためにこのファイルをインクルードしている．

4.2.2 IplImage 構造体と画素へのアクセス

次に main 関数を見ていくと IplImage へのポインタ型の変数が定義されている．IplImage は OpenCV で画像を扱うための構造体であり，この構造体には画像サイズや画素の型，画像データへのポインタなどの情報が収められている．6 に IplImage 構造体の定義の一部を示す．なお，この定義では説明しないメンバは省略しており，メンバ定義の順序やコメントは適宜書換えてある．

これらのメンバを使って，画像の各画素へ直接アクセスすることができる．画像のデータは IplImage 構造体のメンバ imageData に収められており，画像中のある座標の画素が収められているメモリアドレスの計算方法は次のとおりである．

```
typedef struct _IplImage
{
    int  nChannels;     /* チャンネル数 */
    int  width;         /* 画像の幅 */
    int  height;        /* 画像の高さ */
    int  imageSize;     /* 画像のサイズ (バイト数) */
    char *imageData;    /* 画像のデータへのポインタ */
    int  widthStep;     /* 1 ラインのバイト数 */

    /* ... 省略 ... */
}
IplImage;
```

図 6: IplImage 構造体の一部

```
pixel = image->imageData
      + y 座標 * image->widthStep
      + x 座標 * image->nChannels
```

IplImage 構造体のメンバ widthStep は，画像の 1 ラインのバイト数である．また nChannels は色成分の数であり，例えば RGB のカラー画像なら 3，濃淡画像なら 1 となる．

上記の式で計算した pixel のアドレスから連続した領域に画素の各色値が収められている．例えば，RGB カラー画像なら `pixel[0]` が赤色成分，`pixel[1]` が緑色成分，`pixel[2]` が青色成分となる[1]．

4.2.3 画像のロードとセーブ

cvLoadImage() と cvSaveImage() は，それぞれファイルからの画像の読み込みと，ファイルへの保存の関数である．使い方は以下の通りである．

```
image = cvLoadImage("ファイル名",isColor)
cvSaveImage("ファイル名",image)
```

cvLoadImage() は指定したファイルから画像を読み込み，IplImage 構造体へのポインタを返す．最後の引数 isColor は，カラー画像として読み込むか，濃淡画像として読み込むかのフラグであり，正の整数を指定するとカラーとして読み込み，0 を指定すると濃淡画像として読み込む．また，負の整数を指定した場合は画像ファイルから自動で判別する．cvLoadImage() は，画像を収める領域を確保して返すので，プログラムで使い終ったときには，cvReleaseImage() でメモリ領域を解放する必要がある．

cvSaveImage() は，指定したファイルに画像を保存する関数である．このときのファイルフォーマットは，ファイル名から自動判定される．

[1]このプログラムは各色成分が 8bit 符号なしと仮定しているが，それ以外の場合は適切な型のポインタでアクセスする必要がある．また，画素毎の色成分を連続した領域に収めない扱い方も存在する．詳しくはリファレンスマニュアル参照のこと．

4.2.4 新しい画像の作成

Example1.c では，画像をファイルから読み込んでいたが，新しく画像を作成することもできる．新しく画像を作成するには，cvCreateImage() を使う．

```
cvCreateImage(cvSize(画像の幅，画像の高さ)，画素の型，
チャンネル数)
```

cvCreateImage() は指定された引数に基づいて画像を作成し，IplImage 構造体へのポインタを返す．画素の型は，各画素をどのような型として扱うかを示すフラグであり，2 に示す 8 種類のマクロで指定することができる．例えば，640x480 の RGB カラー画像で各色成分が 8bit 符号なしの画像を作成するには次のように指定する．

```
IplImage *new_image;
new_image = cvCreateImage(cvSize(640,480),
                          IPL_DEPTH_8U, 3);
```

表 2: OpenCV で扱える画素の型

マクロ名	画素の型
IPL_DEPTH_8U	8bit 符号なし整数
IPL_DEPTH_8S	8bit 符号付き整数
IPL_DEPTH_16U	16bit 符号なし整数
IPL_DEPTH_16S	16bit 符号付き整数
IPL_DEPTH_32U	32bit 符号なし整数
IPL_DEPTH_32S	32bit 符号付き整数
IPL_DEPTH_32F	32bit 浮動小数点数
IPL_DEPTH_64F	64bit 浮動小数点数

4.2.5 画像処理関数の利用

Example1.c では，画像の各画素を直接アクセスして画像処理を行ったが，OpenCV には各種画像処理関数が豊富に用意されており，画像処理プログラムを手軽に作成することができる．

7 に示すプログラムは，Example1.c の画像の処理部分を変更し，Canny のエッジ抽出 [Canny 86] の関数を使用したものである．コンパイル方法，実行方法は Example1.c と同様である．

このプログラムでは，cvLoadImage() の最後の引数を 0 として，濃淡画像として画像を読み込み，cvCanny() 関数でエッジ抽出を行っている．後の 3 つのパラメータは，Canny アルゴリズムの閾値とマスクサイズとなっており，詳細はマニュアルを参照されたい．8 にこのプログラムの実行結果を示す．

この他にも OpenCV には，様々な画像処理やコンピュータビジョンアルゴリズムを実装した関数が含まれているので，高度な画像処理プログラムを非常に簡単に作成することができる．どんな関数があるか，またその詳細は，マニュアルを参照していろいろ試してみると良い．

```
IplImage *image,*edge;
// 画像ファイルを濃淡画像として読み込む
image = cvLoadImage(argv[1],0);
// 同じサイズの濃淡画像を用意する
edge = cvCreateImage(cvGetSize(image),
                     IPL_DEPTH_8U,1);
// エッジ抽出
cvCanny(image,edge,10,128,3);
// 処理した画像をファイルに書き込む
cvSaveImage(argv[2],edge);
```

図 7: エッジ抽出のプログラム (Example2.c 抜粋)

図 8: Example2.c の実行結果

5 簡易 GUI ライブラリ

本章では，OpenCV に添付されている，マルチプラットフォームの簡易 GUI である HighGUI の使い方について解説する．HighGUI は，基本的な機能だけを持った GUI なので，より高度なインターフェースを構築するにはプラットフォーム固有の GUI ツールキットを使うのをお勧めするが，使い方が簡単であり全く同じソースファイルで，異なる環境で動く GUI プログラムが作成できるので，プログラムの試作や，Linux でも Windows でも動くプログラムを作成したいときには便利である．ここでは，ごく基本的な使用法のみを紹介するので，詳細はマニュアルを参照されたい．

まずは，Example3.c を見て欲しい．このプログラムは，入力画像を濃淡画像として読み込み，cvTreshold() で二値化を行うプログラムである．二値化の閾値をトラックバー (スクロールバー) でコントロールできるようになっている．プログラムのコンパイル方法，実行方法は，Example1.c などと同じで，Linux でも Windows でも同じソースファイルから GUI プログラムを作成できる．

HighGUI の使い方は非常にシンプルである．まずは cvNamedWindow() で名前を付けたウィンドウを作成する．そして cvCreateTrackbar() でトラックバー (スクロールバー) を付けることができる．

```
// ウィンドウを作る
cvNamedWindow(wndname,1);
// トラックバーを作る
cvCreateTrackbar("threshold", wndname, &thresh,
                 255, on_trackbar);
on_trackbar(0);
// 「q」が押されるまで待つ
while(cvWaitKey(0) != 'q');
```

```
#include <stdio.h>
#include <stdlib.h>
// OpenCV のヘッダファイル
#include <cv.h>
#include <highgui.h>

char *wndname;            //ウィンドウの名前
IplImage *image,*binary; //画像を扱う構造体
int thresh = 128;
// トラックバーのコールバック
void on_trackbar(int t){
  cvThreshold(image,binary,(double)thresh,
              255,CV_THRESH_BINARY);
  cvShowImage(wndname,binary);
}

int main(int argc, char **argv){
  char *pixel; // 画素のポインタ
  int x,y,c;   //ループカウンタ

  //引数の個数のチェック
  if(argc<2) {
    fprintf(stderr,"usage:%s input output\n",
            argv[0]);
    exit(1);
  }
  // ウィンドウの名前をセット
  wndname=argv[0];
  // 画像ファイルを濃淡画像として読み込み
  image = cvLoadImage(argv[1],0);
  // 同じサイズの濃淡画像を用意する
  binary = cvCreateImage(cvGetSize(image),
                         IPL_DEPTH_8U,1);
  // ウィンドウを作る
  cvNamedWindow(wndname,1);
  // トラックバーを作る
  cvCreateTrackbar("threshold", wndname, &thresh,
                   255, on_trackbar);
  on_trackbar(128);
  // 「q」が押されるまで待つ
  while(cvWaitKey(0) != 'q');
  // 画像の解放
  cvReleaseImage(&image);
  cvReleaseImage(&binary);
  return 0;
}
```

図 9: GUI プログラムの例 (Example3.c)

(a) Linux での実行結果

(b) Windows での実行結果

図 10: Example3.c の実行結果

wndname はウィンドウの名前の文字列である．
thresh は int 型で，トラックバーで値をコント

ロールする変数を指定する．その後の 255 はトラックバーの最大値であり，on_tracker はコールバック関数である．on_tracker は以下のように定義されている．

```
char *wndname;            //ウィンドウの名前
IplImage *image,*binary; //画像を扱う構造体
int thresh = 128;
// トラックバーのコールバック
void on_trackbar(int t){
  cvThreshold(image,binary,(double)thresh,255,
              CV_THRESH_BINARY);
  cvShowImage(wndname,binary);
}
```

cvTreshold() は，濃淡画像の二値化を行う関数であり，thresh で閾値を決定している．cvShowImage() は wndname で指定した名前のウィンドウに画像を表示する．

また，main 関数の中の cvWaitKey(0) で何かキーが押されるのを待つことができる．

```
  while(cvWaitKey(0) != 'q');
```

このプログラムでは，q が押されると終了する．この cvWaitKey(0) の戻り値を if 文などで判別することによって，押されたキーによって，色々な処理を行わせることができる．

6 画像キャプチャとリアルタイム処理

本章では，cvCapture による，カメラからの画像キャプチャの方法とリアルタイムの画像処理の方法について解説する．OpenCV は，カメラからの取り込みをサポートする cvCapture 関数群が用意されている．これらの関数を用いることで，Linux と Windows で動くリアルタイムのプログラムを簡単に作成できる．

Linux 環境では，Video for Linux に対応したキャプチャーカードと，IEEE1394 インターフェース経由でキャプチャする DCAM 規格のカメラをサポートしている．Windows 環境では，Video for Windows と MIL(Matrox Image Library) をサポートしている．ここでは，それぞれの環境におけるカメラの設定自体については割愛するので，インターネット上の情報 [Yoshimoto, Knorr, Sakurai] などを参考に設定して欲しい．

11 は，Example3.c を，画像をカメラから入力しリアルタイムで処理を行うように変更したものである．コンパイル，実行方法はいままでの例と同様である．

CvCapture 構造体は，カメラやキャプチャーカードの情報を持っており cvCaptureFromCAM で初期化する．

```
  CvCapture *capture=0;
  //カメラからのキャプチャ
  capture = cvCaptureFromCAM(0);
```

cvCaptureFromCAM の引数はカメラのインデックスを表しており，0 を指定した場合には自動で

```
#include <stdio.h>
#include <stdlib.h>
// OpenCV のヘッダファイル
#include <cv.h>
#include <highgui.h>

int main(int argc, char **argv){
  char *wndname;            //ウィンドウの名前
  IplImage *image=0,*gray=0,*binary=0; //画像を扱う構造体
  int thresh = 128;
  CvCapture *capture=0;

  //カメラからのキャプチャ
  capture = cvCaptureFromCAM(0);
  // ウィンドウの名前をセット
  wndname=argv[0];
  // ウィンドウを作る
  cvNamedWindow(wndname,0);
  // トラックバーを作る
  cvCreateTrackbar("threshold", wndname,
                   &thresh, 255, NULL);
  // 「q」が押されるまで繰り返す
  while(cvWaitKey(10) != 'q'){
    // キャプチャ
    image = cvQueryFrame(capture);
    // キャプチャできなければ終了
    if(!image) break;
    // 同じサイズの濃淡画像を用意する
    if(!gray)
      gray = cvCreateImage(cvGetSize(image),
                           IPL_DEPTH_8U,1);
    if(!binary)
      binary = cvCreateImage(cvGetSize(image),
                             IPL_DEPTH_8U,1);
    // 濃淡画像に色変換
    cvCvtColor(image,gray,CV_BGR2GRAY);
    // 二値化
    cvThreshold(gray,binary,(double)thresh,255,
                CV_THRESH_BINARY);

    cvShowImage(wndname,binary);
  }
  // 画像の解放
  cvReleaseImage(&gray);
  cvReleaseImage(&binary);
  return 0;
}
```

図 11: リアルタイム処理の例 (Example4.c)

使用可能なカメラを探して初期化を行う．複数のカメラがある場合などは，この引数を変えることで，カメラを指定することができる．OpenCV では，カメラの操作が動画ファイルの操作を統合されており，ここを cvCaptureFromAVI("ファイル名") に変えると，他の部分はそのままで AVI などの動画ファイルから画像を読み込んで処理させることができる．

実際にカメラから画像を取り込むには cvQueryFrame() を使う．

```
    image = cvQueryFrame(capture);
```

この関数は，引数に CvCapture 構造体へのポインタを指定すると，カメラから一枚画像を取り込んで，IplImage 構造体へのポインタを返す．ここで

返ってくる画像のメモリは cvCaptureFromCAM の中で確保されているので，プログラムの途中で解放してはいけない点に注意する．

その他の部分は今までのプログラムとほぼ同様であるが，cvWaitKey() の引数が 10 となっているのは，キー入力待ちでプログラムが止まらないようにするためであり，10 の数字はタイムアウト時間 (ミリ秒) である (0 を指定した場合はタイムアウトしない)．

7 まとめ

本稿では，Intel が開発し，オープンソースで公開している OpenCV の使い方について概説した．OpenCV は様々な画像処理やコンピュータビジョンアルゴリズムを簡単に使えるだけでなく，シンプルな GUI や画像取り込みの機能が含まれているため，従来敷居が高いと思われていたような，高度な画像処理プログラムやリアルタイムの画像処理のプログラムを，非常に簡単に作成することができる．また，世界で多くの画像処理，コンピュータビジョンの専門家が利用しているため，より効率的なアルゴリズムへの書き変えや，バグの修正なども頻繁に行われている．

コンピュータビジョンの専門家以外には，高度なアルゴリズムを手軽に利用するための環境として有用であり，また専門家にとっても OpenCV を用いて自分が提案したアルゴリズムを実装して公開すれば，多くの人に試してもらえることが期待できる．画像処理，コンピュータビジョンの専門家や，専門家ではないが画像を扱う必要のある人はぜひ試してみて欲しい．

なお，ここで紹介したのは OpenCV の機能のごく一部であり，他にも様々な関数，機能が存在している．自分でアルゴリズムを実装する前に，リファレンスマニュアルを読んで自分が使いたい機能が含まれていないか確認して欲しい．

参考文献

[Intel] Intel, : *Open Source Computer Vision Library*, Intel Corp., http://www.intel.com/technology/computing/opencv/

[Yahoo] Yahoo, : *Yahoo Groups OpenCV*, Yahoo! Inc., http://groups.yahoo.com/group/OpenCV/

[栗田] 栗田 雄一：IPL，OpenCV を使った画像処理プログラミング， http://robotics.aist-nara.ac.jp/~yuuich-k/HowToIpl-euc/index.html

[Bando] Bando, T.: *OpenCV*, http://hawaii.aist-nara.ac.jp/ ~takash-b/pukiwiki/index.php?OpenCV

[kenta-ta] kenta-ta, : *OpenCV Documentation(draft)*, http://kenta-ta.kir.jp/reference.html

[木村] 木 村 誠：*OpenCV on cygwin*, http://www.dh.aist.go.jp/~kimura/opencv/opencv-0.9.6.html.ja

[西口] 西 口 敏 司：*Tips/OpenCV-0.9.6*, http://www.mm. media.kyoto-u.ac.jp/members/nishigu/pukiwiki.php?Tips/OpenCV-0.9.6

[熊本大学] 熊本大学 内村・胡研究室マルチカメラ班：OpenCV を使った Visual C++ Programming, http://navi.cs.kumamoto-u.ac.jp/~ryu/opencv_howto.html

[Yoshimoto] Yoshimoto, H.: Linux+OpenCV+1394 カメラ HOWTO, http://limu.is.kyushu-u.ac.jp/~yosimoto/work/opencv-howto/

[Yonekura] Yonekura, M.: *Ruby/OpenCV*, http://blueruby. mydns.jp/opencv/

[Knorr] Knorr, G.: *Video4linux*, http://linux.bytesex.org /v4l2/

[Sakurai] Sakurai, T.: テレビキャプチャカードの設定, http://park15.wakwak.com/~unixlife/linux/app-tv.html

[Bradski 98] Bradski, G.: Computer vision face tracking as a componento of a perceptual user interface, in *Workshop on Applications of Computer Vision*, pp. 214–219 (1998)

[Canny 86] Canny, J.: A. Computational Approach to Edge Detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679–698 (1986)

[Lucas 81] Lucas, B. and Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision, in *Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674–679 (1981)

[Viola 01] Viola, P. and Jones., M. J.: Rapid Object Detection using a Boosted Cascade of Simple Features., in *Proc. of Computer Vision and Pattern Recognition (CVPR2001)*, Vol. 1, pp. 511–518 (2001)