デバイスプログラミング

2019年5月7日

担当：伊藤

デバイスプログラミング

- デバイス：コンピュータ周辺機器の総称（大まかに言うと）
- ゲームへのデバイスの利用
- 今回はPC内臓カメラを使用
- カメラから取得された画像情報を用いる
- OpenCVを使用したプログラミング




株式会社スパイス メディアワークス営業部 http://www.mocap.jp/
SONY http://www.jp.sonystyle.com/
ELECOM　http://www2.elecom.co.jp/
株式会社 3 DWIN http://www.3d-win.co.jp/

デバイスプログラミング

- デバイスごとにライブラリが存在
- ライブラリの使用方法を調査しながら使う
- デバイスの特性を理解する
- アイデアを膨らませる
  （ゲームにどのように利用しよう？？？）

講義の目的

- カメラで取得した情報を入力とし，ゲームを制御するための基礎知識の習得
- OpenCVを用いた画像処理プログラミング

本日取り組む内容

Step 1

- Webカメラから画像を取得

Step 2

- OpenCVを用いた画像処理
- 情報を抽出

Step 3

- 画像処理から得られた情報をゲームへ入力
- ゲーム内での処理
- 処理結果をゲーム画面へ出力

講義内容

- OpenCVのサンプルプログラムを実行
- OpenCVの基本的な関数と画像の扱い方を理解
  - 画像ファイルを開けるか？
  - 画像の中身を操作
  - カメラの入力を確認できるか？
  - カメラで取得された画像を操作
- テンプレートマッチング
  - 静止画を対象（授業内で全員で実施）
  - カメラから取得された画像を対象（レポート課題）

画像処理とOpenCV

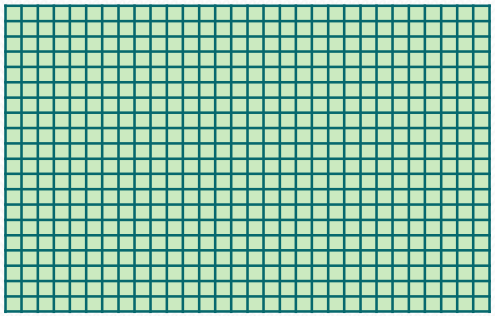
- 画像処理（Image Processing）
  - 例：デジタルカメラの顔認識
  - 動画：画像の連続
- OpenCV(最新版はOpenCV4.1.0（2019年5月5日現在），電算室は2系)
  - マルチプラットフォーム：OSの違いを吸収
    例 同じ関数でカメラを取り扱える
  - 関数：“cv”から始まる
  - 構造体やクラスなど：“Cv”から始まる
  - 定数やマクロ：“CV_”から始まる
  - opencv samples and documents：http://opencv.jp/

OpenCVで画像を扱う

例

30画素

20画素



IplImage *img;
↑構造体

高さ（y軸方向）：20画素
幅　（x軸方向）：30画素
画素数：600画素

- 1画素はBGR(青緑赤，8bitずつ計24bit)の値を持つ
- 画像は1次元配列で構成　img->imageData に画素情報が入っている

imageData[0]~imageData[89] 画像の横1列分：30画素分

B(0,0)	G(0,0)	R(0,0)	B(1,0)	G(1,0)	R(1,0)	B(2,0)	G(2,0)	R(2,0)	…	B(29,0)	G(29,0)	R(29,0)	…
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]		[87]	[88]	[89]	

…	B(599,599)	G(599,599)	R(599,599)
	[1797]	[1798]	[1799]

サンプルプログラムの確認①（1/3）

- ノートPC付属のWebカメラを利用する
- サンプルプログラムを動かして，カメラが認識されているか確認する
- 講義用プログラム

  デバイスプログラミング　サンプル

  →　**deviceprog2019.tar.gz**

  - 辻技術職員にご協力いただきました！感謝！！
- manabaからダウンロード
- 本日の作業用ディレクトリを作成し，そこで解凍

サンプルプログラムの確認①（2/3）

- 解凍　⇒　$ tar -zxvf deviceprog2019.tar.gz
  - 解凍後にできるディレクトリ　deviceprog2019の中
    1. image_cv　OpenCV で静止画像を開いて表示
    2. video_cv　OpenCV による動画像の表示
    3. image_sdl　OpenCV で画像を読み込みSDLで表示
    4. video_sdl　OpenCV で動画像を取得しSDLで表示
    5. templatematching　テンプレートマッチング用

サンプルプログラムの確認①（3/3）

- 動作確認
- Makefileでコンパイル

　　⇒　各ディレクトリに移動し，$ make

①OpenCVのみ

- $ ./image_cv　　OpenCV で静止画像を開いて表示
- $ ./video_cv　　OpenCV でカメラのキャプチャ動画像の表示

②SDLも含めて

- $ ./image_sdl　　OpenCV で画像を読み込みSDLで表示
- $ ./video_sdl　　OpenCV でカメラのキャプチャ動画像を取得しSDLで表示

サンプルプログラムの確認②（1/5）

- image_cv.c をエディタで開く
- 画像ファイルを読み込み，画像を操作する

```
/*_________ INCLUDES_____________________________________*/
#include <stdio.h>
#include <opencv2/imageproc/iamgeproc_c.h>
#include <opencv2/highgui/highgui_c.h>
```

- iamgeproc_c.h，highgui_c.h：
  OpenCVの関数を使うためのヘッダファイル

サンプルプログラムの確認②（2/5）

```
int x, y;
uchar p[3];
IplImage *img;

img = cvLoadImage("test.jpg", CV_LOAD_IMAGE_COLOR);
if (img == NULL) {
    fprintf(stderr, "*Error* cannot open test.jpg¥n");
    return;
}
```

- int x, y;　：画像の画素を扱うための変数
- uchar p[3];　画素値を格納する配列
- IplImage *img;　IplImage構造体へのポインタ
- cvLoadImage　：画像を読み込む関数
- test.jpg：読み込む画像ファイル

IplImage構造体

```
Typedef struct _IplImage
{
        int         nSize;              IplImageのデータサイズ
        int         nChannels;          チャンネル数
        int         depth;              ピクセルごとのデプス
        int         dataOrder;          0：インタリーブカラーチャンネル，1：分離カラーチャンネル
        int         origin;             0：左上原点，1：左下原点
        int         width;              画像の幅　（ピクセル数）
        int         height;             画像の高さ（ピクセル数）
        struct  _IplRIO   *roi;         関心領域へのポインタ
        int         imageSize;          画像データのバイト数
        char        *imageData;         画像データへのポインタ
        int         widthStep;          画像データの横１行ごとの幅（バイト幅）
        char        *imageDataOrigin;   元々の画像データへのポインタ
}
```

使い方の例

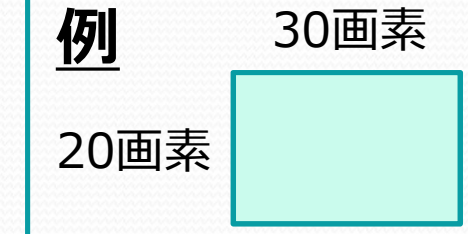
```
IplImage *img;

img = cvLoadImage("test.jpg", CV_LOAD_IMAGE_COLOR);
if (img == NULL) {
    fprintf(stderr, "*Error* cannot open test.jpg¥n");
    return;
}
```

サンプルプログラムの確認②（3/5）

```
for (y = 0; y < img->height; y++) {
        for (x = 0; x < img->width; x++) {
            p[0] = img->imageData[img->widthStep * y + x * 3];          // B
            p[1] = img->imageData[img->widthStep * y + x * 3 + 1];      // G
            p[2] = img->imageData[img->widthStep * y + x * 3 + 2];      // R
```

例



高さ　img->height：20
幅　　img->width：30
画素数：600画素
画像の横1行のバイト幅　img->widthStep：90(byte)

imageData[0]~imageData[89] 画像の横1列分：30画素分

B(0,0)	G(0,0)	R(0,0)	B(1,0)	G(1,0)	R(1,0)	B(2,0)	G(2,0)	R(2,0)	…	B(29,0)	G(29,0)	R(29,0)	…
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]		[87]	[88]	[89]	

…	B(599,599)	G(599,599)	R(599,599)
	[1797]	[1798]	[1799]

サンプルプログラムの確認②（4/5）

```
// Image Processing
        img->imageData[img->widthStep * y + x * 3] =
            cvRound(p[0] * 0.0);
        img->imageData[img->widthStep * y + x * 3 + 1] =
            cvRound(p[1] * 0.0);
        img->imageData[img->widthStep * y + x * 3 + 2] =
            cvRound(p[2] * 1.0);
```

- *imageData　：IplImage型のimgのメンバ変数
  画像データへのポインタ
- widthStep　：IplImage型のimgのメンバ変数
  画像データの横1行ごとの幅（バイト数）
- cvRound：引数に最も近い整数値を返す関数

サンプルプログラムの確認②（5/5）

```
cvNamedWindow("Image", CV_WINDOW_AUTOSIZE);
cvShowImage("Image", img);
cvWaitKey(0);

cvDestroyWindow("Image");
cvReleaseImage(&img);
```

- cvNamedWindow：ウィンドウの生成
- cvShowImage：画像の表示
- cvWaitKey：キー入力を待つ　Esc or 指定値ミリ秒待ち
- cvDestroyWindow：ウィンドウの破棄
- cvReleaseImage：メモリの解放

サンプルプログラムへ追加

- image_cv.cを書き変えて，原画像も表示させてみよう
- 画像を複製する関数：cvCloneImageを使用

```
IplImage   *img2;
img2 = cvCloneImage (img);

cvNamedWindow("Image2", CV_WINDOW_AUTOSIZE);
cvShowImage("Image2", img2);

cvDestroyWindow("Image2");
cvReleaseImage(&img2);
```

Image
画像処理後の画像

Image2
原画像

テンプレートマッチング(1/4)

- パターン（pattern）

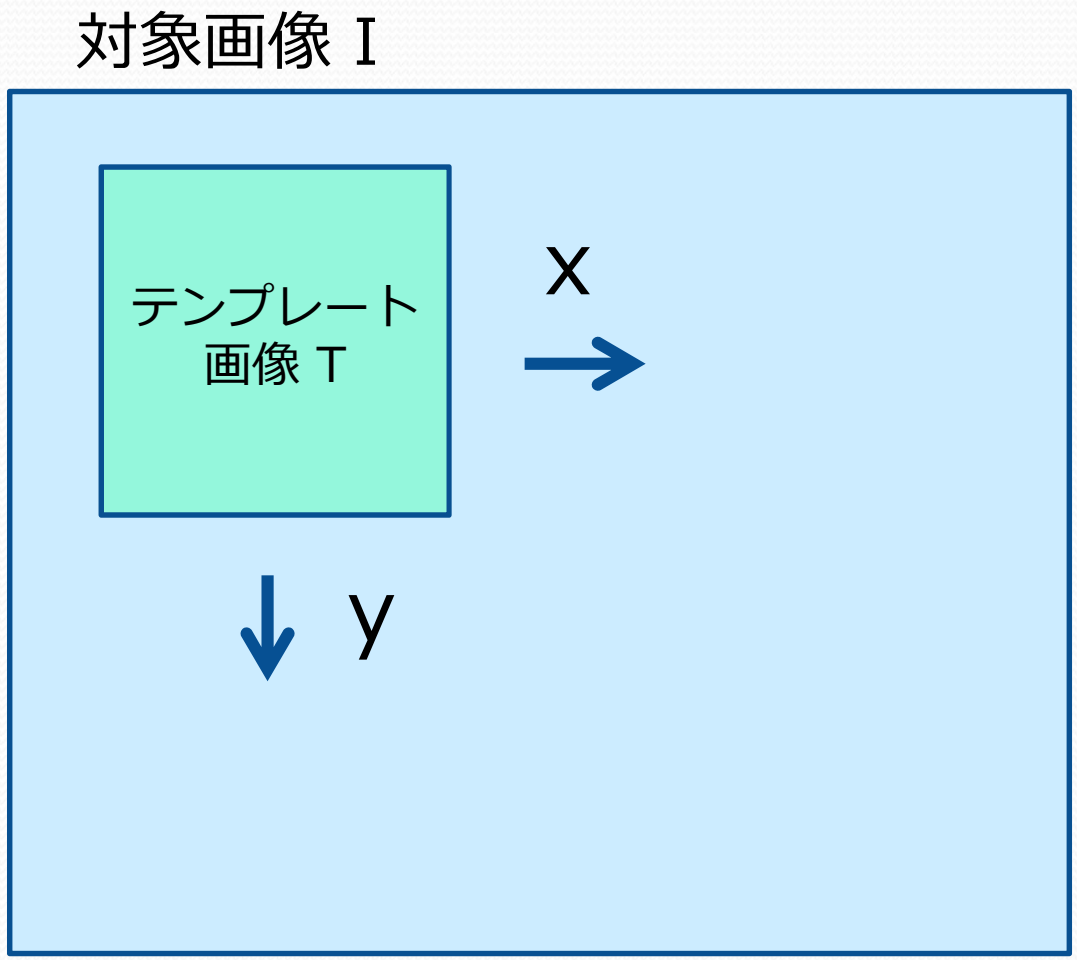
  画像の視覚的特徴や画素値そのもの
- パターンマッチング（pattern matching）

  パターンの存在や位置を検出すること
- テンプレート（template）

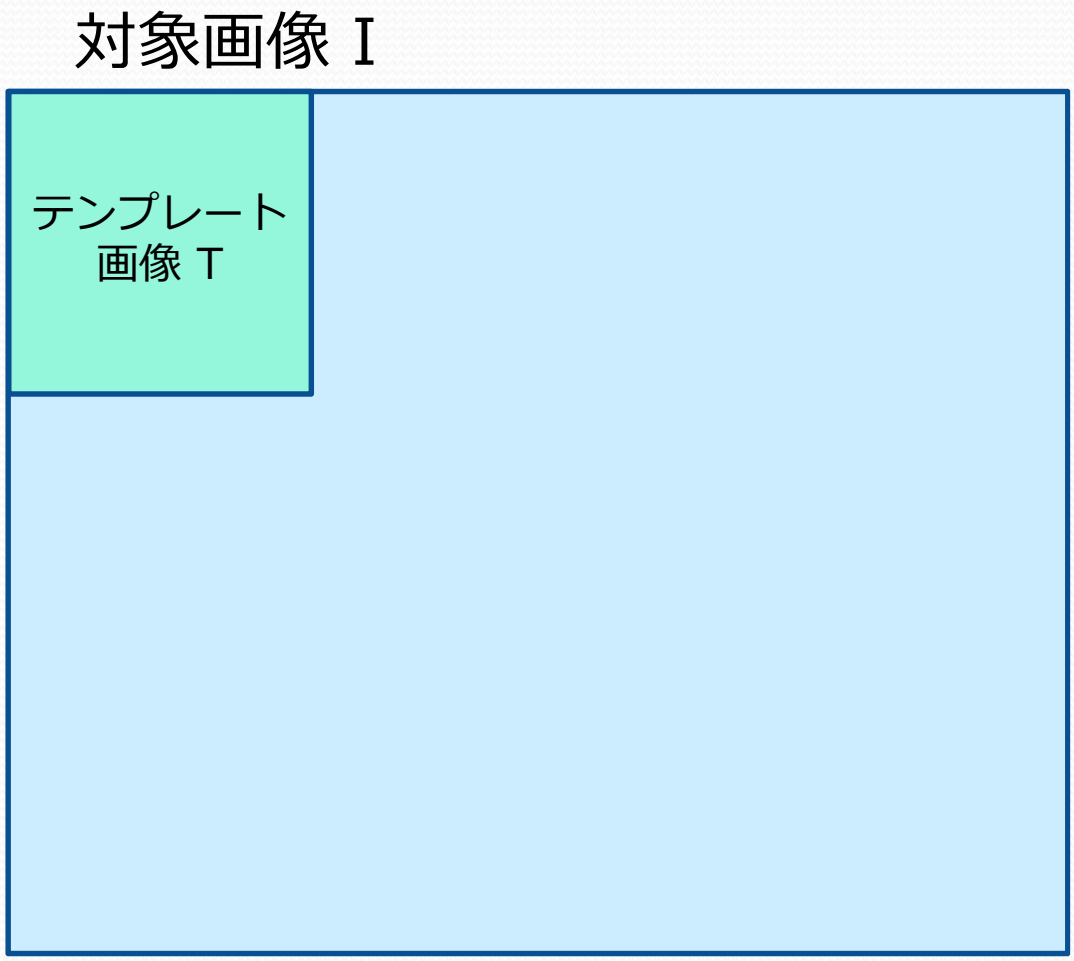
  予め準備する標準パターン
- テンプレートマッチング（template matching）

  テンプレートを用いて入力画像との
  マッチングを行う

テンプレートマッチング(2/4)



テンプレートマッチング



テンプレートマッチング(3/4)

- 類似度（similarity measure）

　　２つの画像がどの程度似ているかを表す

テンプレートの大きさ：M×N
テンプレートの位置(i,j)における画素値：T(i,j)
テンプレートと重ね合わせた対象画像の画素値：I(i,j)

正規化相互関数：NCC（Normalised Cross-Correlation）

$$R_{NCC} = \frac{\sum_{j=0}^{N-1}\sum_{i=0}^{M-1} I(i,j)T(i,j)}{\sqrt{\sum_{j=0}^{N-1}\sum_{i=0}^{M-1} I(i,j)^2 \times \sum_{j=0}^{N-1}\sum_{i=0}^{M-1} T(i,j)^2}}$$

テンプレートマッチング(4/4)

- 相違度（dissimilarity measure）

  ２つの画像がどの程度似ていないかを表す

差の2乗和（ユークリッド距離の2乗）：SSD（Sum of Squared Difference）

$$R_{SSD} = \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} (I(i, j) - T(i, j))^2$$

差の絶対和（市街地距離）：　SAD（Sum of Absolute Differnce）

$$R_{SSD} = \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \left| I(i, j) - T(i, j) \right|$$

やってみよう

- サンプルプログラム
  - templatematching:
    templatematching.c
    makefile
  - video_cv:
    video_cv.c
    makefile

★テンプレートマッチングに使用する画像がありません！

まずはプログラムを見てみよう

カメラから画像を取得

- “video_cv.c”を用いて，カメラから取得した画像を保存するプログラムを作成する

★Escキーを押すと終了 → 終了前に画像保存

画像保存のための関数

cvSaveImage(“ファイル名”,frame, 0);

正面を向いた顔が入るようにして保存する

ファイル名：image.png

テンプレート画像の作成

★画像編集ソフトを使用し，テンプレート画像を作成

-アプリケーション

-グラフィックス

-GIMP → 起動

GIMP

メニューから先程保存した画像を開いて，
ツールボックスの矩形選択で顔の領域を切り出す
(C-xで切り取り，別の画像にC-vで貼り付け)
画像の生成→クリップボードから
Export as…で“template.png”としてtemplate.cと同じ
ディレクトリに保存
さきほどvideo_cvで保存した“image.png”も同じディレクトリに保存

その後、make して、./templatematching で実行

レポート課題（1/2）

1. image_cv.cを参考に，画像の左半分の領域のみ色を変換し，表示させるプログラムcv_kadai01.cを作成せよ.
   - 1.1 作成したプログラム本文をレポートに示せ.
   - 1.2 結果画像を示せ.
2. 本日の内容を参考に，前回使用したvideo_cv.c に追記する形で，カメラから取得した画像におけるテンプレートマッチングを実現する，“video_template.c”を作成せよ．使用するテンプレート画像は，本日作成した正面顔の“template.png”をとする．なお、テンプレートおよび探索対象画像の2値化画像を確認できるようにすること（実行中に表示するためのウィンドウを作成し，ESCキーを押したときに保存されるように記述する）.
   - 2.1 作成したテンプレート画像を示せ.
   - 2.2 テンプレートマッチングを実施した結果について，成功例，失敗例を示せ．また，失敗例について，なぜ失敗したのか考察せよ．なお，カラー画像に加え、2値化画像をレポートに示して考察せよ.

レポート課題（2/2）

3. 画像の平滑化について以下の項目を実施せよ．

   3.1 画像の平滑化とは何か調査し、延べよ．

   3.2 cvSmooth関数を用いて本日撮影したテンプレート作成用画像（テンプレートを切り出す前の画像）を平滑化するプログラムを作成せよ．また，作成したプログラムをレポートに示せ．フィルタは何を用いても良い．関数の詳細は
   http://opencv.jp/opencv-2svn/c/imgproc_image_filtering.html
   等を参考にすること．パラメータの値は複数試し、出力画像の変化を確認すること．

   3.3 3.2で平滑化した結果、原画像と比較してどのような画像特性となっているか述べよ．

4. 本日紹介したソフト実験で利用可能なデバイス（カメラ含む）の中から１つデバイスを選択し，そのデバイスを用いたゲームを考えて，自分のアイデアをまとめよ．ゲームの概要が分かる図も入れること．

   4.1 選択したデバイス名を示せ．

   4.2 選択したデバイスの特徴と，使用するための開発環境（ライブラリや必要なツールなど）を調査し，まとめよ．

   4.3 選択したデバイスを利用したゲームを考え，図を入れて説明せよ．
   （１ヶ月程度で実現可能な規模のゲームとし，可能であれば，実際の個人開発でその内容を開発してみる）

レポート提出時の注意事項

- 必要なファイルを全てまとめて，**圧縮アーカイブファイル**として提出
- レポートは**PDFファイル**で提出すること.
- レポートには**処理結果の画像を必ず入れる**こと.
  その時，画像サイズは適宜変更すること.
  レポートはPDFファイルのみで良いので，その他の不要なファイルは削除しておくこと.
- レポート以外に，課題ごとにそれぞれディレクトリを作成し，その中にmakeファイル，プログラム本体，使用した画像（カメラから入力を取る場合は不要）を入れ，**採点者がコンパイルできる環境を構築**しておくこと．整っていない場合は採点できません.

締切：５月１４日（火）12:50 まで