今回のテーマ

画像処理用ライブラリの紹介
簡単な画像処理プログラム

画像処理の基本

画像処理とは？

画像といっても、これまで点や四角を描いてきたのと同様に、
座標に色データが対応したものの集合になっている。
200x200の画像だったら40000個の各点に色がついていて、
それを並べたものと考える事ができる。
つまり、画像処理とは、この色の付いた各点に対して解析を行う事で
色を抜き出したり、二値化（モノクロ化）したり、形状を判断したり
と行った様々な処理を行う事を示している。
画像は点の集合でありプログラム上では単純に配列で表す事ができ、
動画はこれらが毎フレーム毎に変化したものである。

画像処理用ライブラリ

Processingでは、JMyronとOpenCVがよく使われているようである。
JMyron : http://webcamxtra.sourceforge.net/
OpenCV: http://ubaa.net/shared/processing/opencv/

OpenCVはWindows, Macintosh,Linuxなど多くのプラットフォームに
対応した良く使われているライブラリである。
Processing向けは、まだ一部のAPIしか実装されていないが、
今回はOpenCVを使う事にする。

なお、IntelMacでJMyronを使うときは、JMyron以外にad1に
IntelMacに対応したlibJMyron.jnilibが必要です。（ad1に置いてあります。）

インストール

各ファイルは、ad1に置いてある。
インストール方法は、上記OpenCVのページに書いてあるので、確認

しながら行う事。

Macintosh:　opencv-framework-1.1.dmg と opencv_01.zip をコピーする。
　　opencv-framework-1.1.dmg からopenCVをインストールする。
Windows:　OpenCV_1.0.exe と opencv_01.zipをコピーする。
　　OpenCV_1.0.exe からOpenCVをインストールする。

これで、OpenCVのライブラリはインストールされました。
Processing以外からも、C, C++, PythonなどからOpenCVを利用できます。
次にProcessing用にOpenCV用のライブラリをインストールします。
・ opencv_01.zipを展開します。
・ processingの環境設定でSketchbook locationフォルダを確認する。
・ 確認したフォルダにlibrariesフォルダを作成し、展開してできた
　OpenCVフォルダごとコピーする。
・ 必要に応じて、ad1からopencv_examples.zipをコピーし展開しておく。

インストールの確認
processingを一旦終了し、再起動する。
・メニューのSketch/Import Library... を選んで、OpenCVが現れるか確認する。
・現れたら選択し、以下の２行のプログラムを実行しエラーが出ないか確認する。

---
import hypermedia.video.*;
OpenCV opencv;

メニューに現れなかったり、エラーが出たら、libraryが正しい場所に
置かれていない可能性が高いので、置き直す。

画像処理プログラミング

ピクセル操作
OpenCVを使って、カメラからの画像を表示します。

---
import hypermedia.video.*;

OpenCV opencv;

```
int MW = 400; // サイズの指定　4:3
int MH = 300;

void setup(){
  size(MW, MH);      // ウインドウのサイズ

  opencv = new OpenCV(this);     // OpenCVオブジェクトの生成
  opencv.capture(MW, MH);        // サイズを指定してキャプチャ指定
}

void draw(){

  opencv.read();      // カメラからのフレームデータ読み込み

  background(opencv.image());     // 読み込んだ画像を背景に指定
}

void mousePressed(){
   opencv.stop();           // OpenCVの停止
}
```

練習問題１：
カメラから画像をキャプチャし、赤色の成分のみを表示する。

```
-------
import hypermedia.video.*;

OpenCV opencv;

int MW = 400;
int MH = 300;

void setup(){
  size(MW, MH);

  opencv = new OpenCV(this);
```

```
  opencv.capture(MW, MH);
}

void draw(){

  opencv.read();

  PImage img = opencv.image();  // キャプチャしたイメージを変数に入れる

  // 各ピクセルの色を知るには、
  // color c = img.pixels[i];
  // 赤色の値を知るには、
  // float red = red(img.pixels[i]);
  // あるピクセルの色を指定するには
  // img.pixels[i] = color(100,100,100);
  // すべてのピクセルについて赤の値を調べて、赤色の成分があれば
  // その他の色を 0 にした色をそのピクセルに指定する。

  image(img, 0, 0);

}

void mousePressed(){
   opencv.stop();
}
--------
```

練習問題２：
練習問題 1 をちょっとだけ改良して、赤の値が強い部分だけを表示する。
どのくらいの赤の強さから表示するか変更しながら試してみる。
このような閾値となるような値のことをスレッショルド(threshold)という。
OpenCVにも、threshold()という関数があります。

授業内課題１：
赤の色が強く、他の色が弱い部分を表示するプログラムを作成する。
赤だけで判断すると、単純に明るい部分も検出してしまうが、
他の色が弱いという制限をいれることで赤く見える部分だけを検出できる。

画像の重心
これまでで、赤い部分を検出することはできた。
赤いエリアの座標を決める事ができれば、カメラに写った画像を
追跡して、マウス代わりにすることができる。
簡単な方法として、エリアの重心を求める方法を用いる。
赤い部分を検出する際に、該当するピクセルのx座標、y座標の値の
平均を求めることで、重心となる座標が得られる。

練習問題３：
緑のエリアの重心を求めて、その座標値に赤い円を表示する。

授業内課題２：
前回習ったネットワークプログラミングと合わせる事で、
練習問題３で得た座標値をネットワーク経由で送り、
軌跡を表示するプログラムを作成する。

動きの検出
動いた部分だけを検出するということは、前回の画像と新しい画像を
比較して、違っている部分が動いた部分になるという方法を用いる。
OpenCVには、absDiff()という関数が用意されている。
まずは、referenceにあるサンプルを実行してみる。

練習問題４：
動きのあった部分の重心を求めて、その場所に円を表示する。
検出する色や明るさなどを調節すると、ある程度思った通りに
動かせるようになります。試行錯誤してみましょう。

顔認識
OpenCVには、顔認識のライブラリが用意されている。
detect()のサンプルを参照。コピーして実行してみる。

輪郭やblobの検出
輪郭や塊の抽出には、blob()が使える。
背景の除去や、動きや色抽出などと組み合わせて使うと、
効果的に物体を認識しやすい。

課題：
顔認識を利用して、顔を動かして玉を跳ね返すようなプログラムを作成する。

Areaクラスを利用し、inside()メソッドを利用する事で、
顔認識で顔の座標を求め、その座標にAreaオブジェクトを表示する。
適当な場所からランダムな方向に動き、壁で跳ね返る円を描画し、
Areaオブジェクトの範囲内に円が入ったら、跳ね返るようにする。

これまでをふまえて、各自の最終課題案の確認