Processingによる画像処理プログラミング

May 21st, 2011
名古屋CV・PRML勉強会2011 :07

藤吉弘亘 Hironobu Fujiyoshi
E-mail: hf@cs.chubu.ac.jp
Twitter: twitter.com/hf149

自己紹介

Hironobu Fujiyoshi

@hf149 Aichi, Japan

Computer Vision Researcher, Professor (Chubu University) コンピュータビジョン・画像認識の研究者, 中部大学教授

http://www.vision.cs.chubu.ac.jp/

FLAB

藤吉研究室

構成：ドクター1人、マスター6人、学部生8人、研究員1人、秘書1人

Twitter：@FLAB

→ 01 : proce55ing

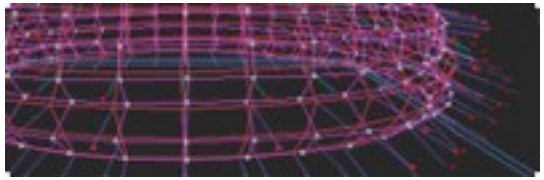
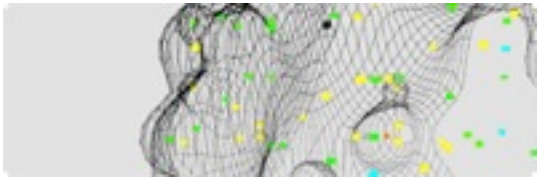
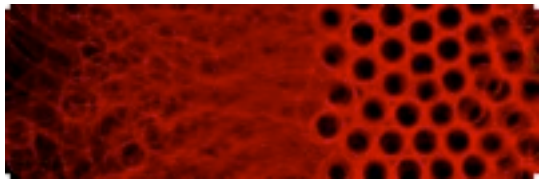
02 : image processing

03 : video processing

04 : open cv

processing

- JAVAをベースに開発されたプログラミング環境
  - Ben Fry(MIT Media Lab)
  - Casey Reas (Design | Media Arts Department at UCLA)
  - ビジュアライゼーションやメディアアートの作品制作等に利用

See→http://processing.org/

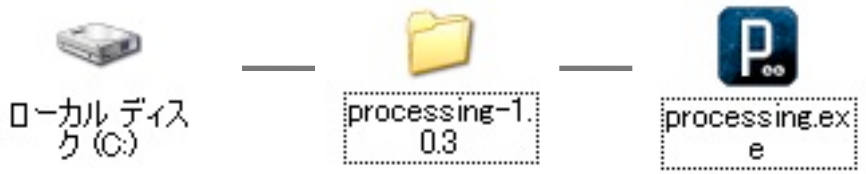
valence: http://acg.media.mit.edu/people/fry/valence/

Valence / Ben Fry

processingのダウンロード

- http://processing.org/download/より使用するOSに合ったものをダウンロード

Windows：解凍後、CもしくはDドライブへコピー

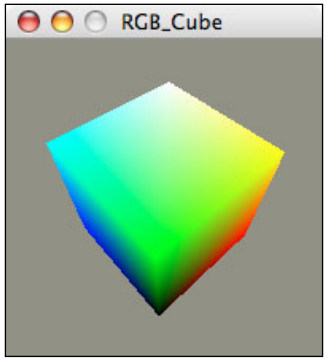


Mac：解凍後、アプリケーションフォルダへコピー

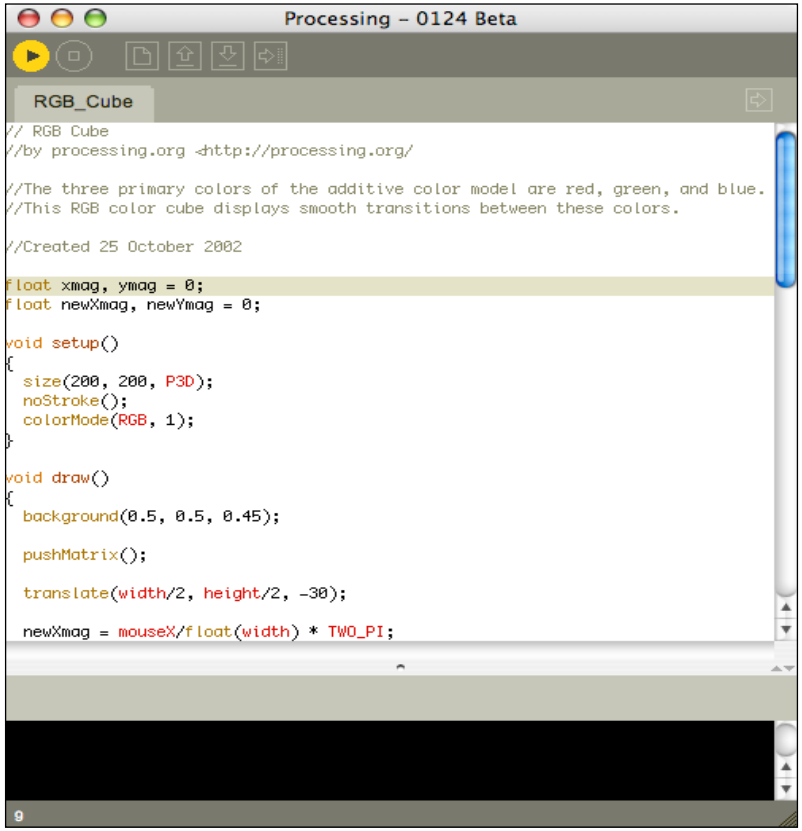
processingのインターフェース

	**実行(play)ボタン**	プログラムを実行する際に使用
	**停止(stop)ボタン**	プログラムを停止する際に使用
	**新規作成(new)ボタン**	新しいファイルのことをProcessingではスケッチと呼ぶ
	**読込み(Opens)ボタン**	スケッチを読み込む
	**保存(Saves)ボタン**	インターフェイス上に表示しているスケッチに名前をつけて保存する際に使用
	**出力(Exports)ボタン**	表示されているスケッチをJavaアプレットとして出力またその際にJavaアプレットを表示するために必要な最低限のHTMLタグを書き出す

processingの実行



Display Window



Menu

Text Editer

Message Area

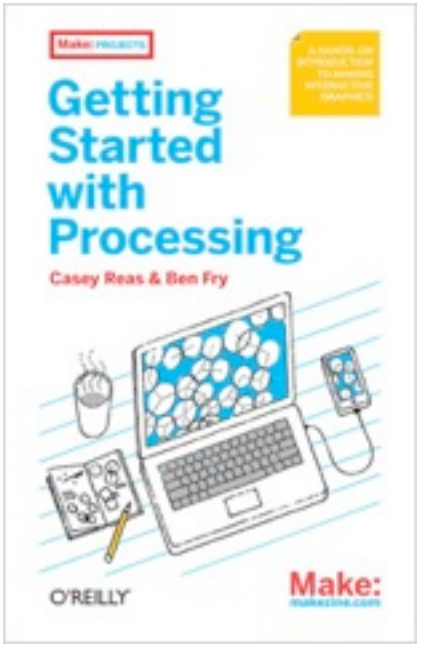
Text Area

URLs

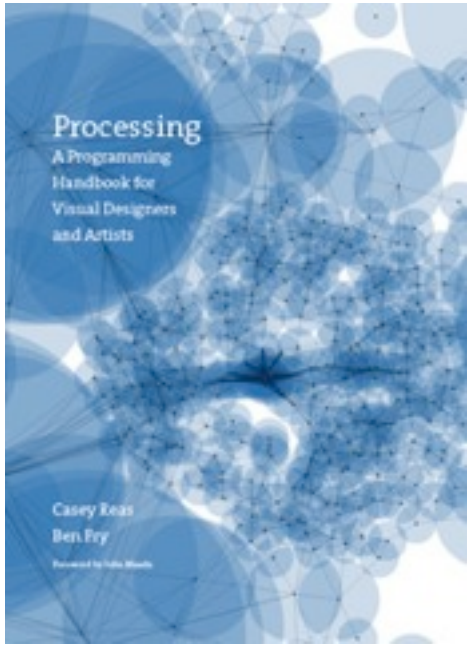
→ http://www.processing.org/
processingのサイト

→ http://www.vision.cs.chubu.ac.jp/P5/
講義資料アーカイブページ

Books



Getting Started with Processing
Casey Reas and Ben Fry.



Processing: A Programming Handbook for Visual Designers and Artists
Casey Reas and Ben Fry

Built with Processing
前川峻志 and 田中孝太郎

01 : proce55ing

→ 02 : image processing

03 : video processing

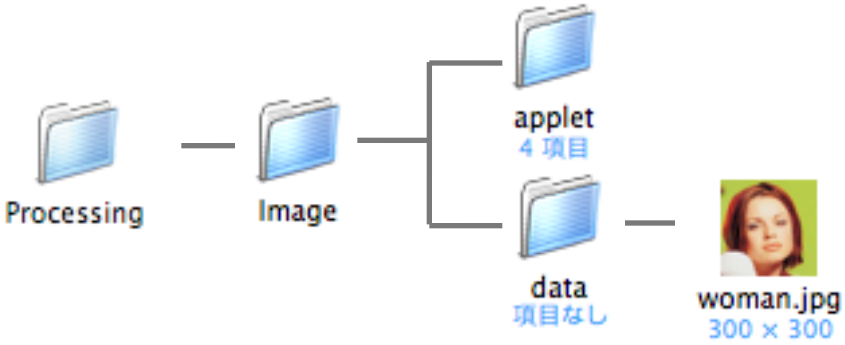
04 : open cv

画像の表示：画像ファイルの用意

1. imageという名前のスケッチを作成
2. Sketch→Add Fileより画像を選択
   (画像は、jpgまたはgif形式のみ)

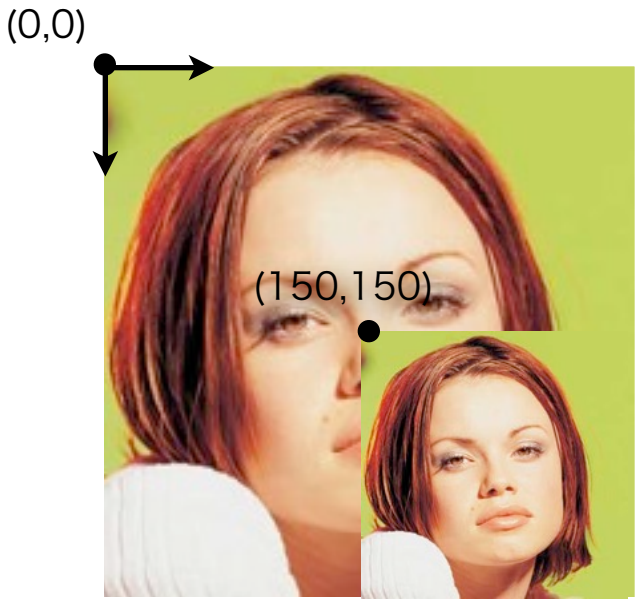
スケッチフォルダ内にdataフォルダが作成される

C:¥Documents and Setteing¥USER¥My Documents



画像の表示：image( )

```
PImage b;              //外部変数
void setup(){
      size(300,300);
      b = loadImage("woman.jpg");
}

void draw()
{
      //画像を座標(0,0)に表示
      image(b, 0, 0);
      //画像を(150,150)にサイズ150×150で表示
      image(b, 150, 150, 150, 150);
}
```



PImageはイメージを保持するためのデータタイプ。image関数が実際に画像をスクリーンに描画する。

image(PImage, x, y, width, height);

画像の表示サイズを決める高さ(height)と横幅(width)については省略することが可能である。省略した場合は元画像のサイズが適用される。

画像サイズの取得：image( )

```
PImage b;

void setup(){
        b = loadImage("woman.jpg");
        size(b.width, b.height);
}

void draw()
{
        image(b, 0, 0);
}
```

外部変数としてbを登録しておくと、どこからも使用可能
画像(woman.jpg)をロードした際、width, heightにその画像サイズが代入される
→これを利用してスケッチのサイズを決定すれば、サイズが違う画像に入れ変えても自動的に反映される

ピクセルデータの取得：get( )

get()は座標(x, y)のピクセルのRGB値を取得

get(x, y)

```
color c = get(150, 160);     →  RGBカラーを扱う変数cに(255, 200, 131)が代入
```

・座標(150, 160)の赤色の値だけを取り出すには
　int a = red(get(150, 160));　　→　aに255が代入される

・座標(150, 160)の緑色の値だけを取り出すには
　int a = green(get(150, 160));　　→　aに200が代入される

・座標(150, 160)の青色の値だけを取り出すには
　int a = blue(get(150, 160));　　→　aに131が代入される

ピクセルデータのセット：set( )

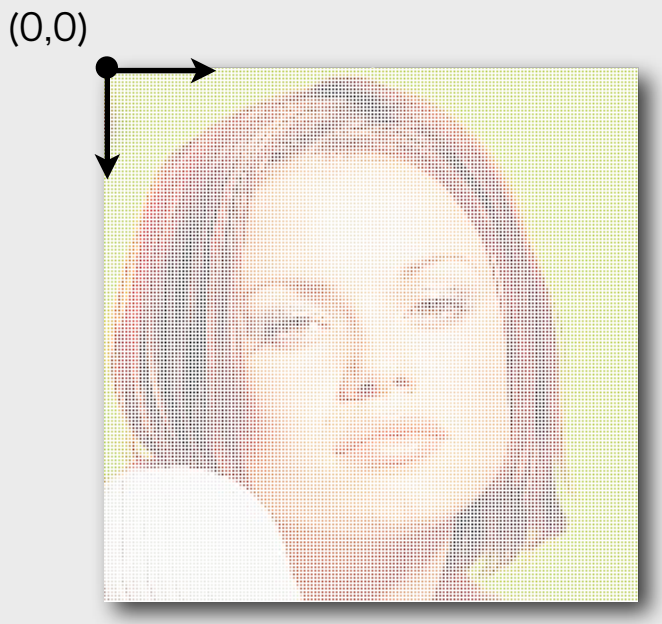
set()は座標(x, y)のピクセルにRGB値をセット

set(x, y, c)

```
color c = color(255, 255, 255);   →   RGBカラー変数cに白色(255,255,255)を代入
set(150, 160, c);                 →   座標(150,160)にカラー変数Cの値をセット
```

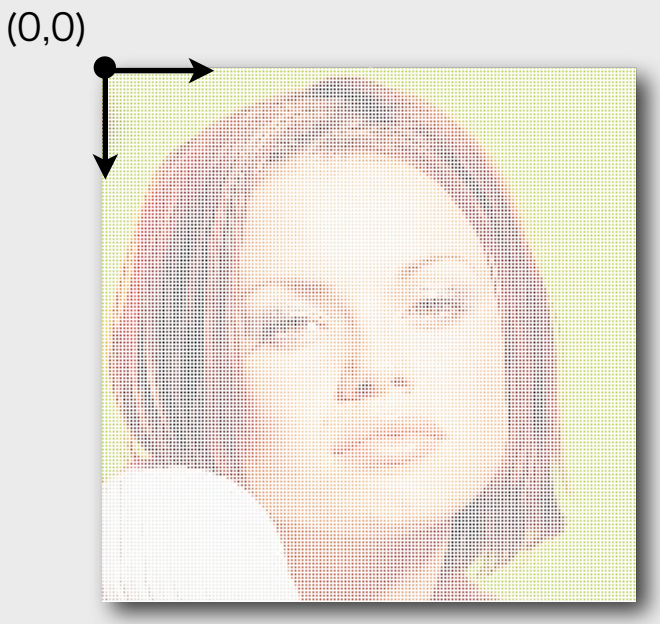
画像ピクセルデータの取得：get( )

```
PImage b;

void setup(){
        b = loadImage("woman.jpg");
        size(b.width, b.height);
}

void draw()
{
        background(255);
        for(int i=0; i<b.height; i+=2){
                for(int j=0; j<b.width; j+=2){
                        set(j, i, b.get(j,i));
                }
        }
}
```



画像ピクセルデータの取得：pixel[ ]

```
PImage b;

void setup(){
        b = loadImage("woman.jpg");
        size(b.width, b.height);
}

void draw()
{
        background(255);
        for(int i=0; i<b.height; i+=2){
                for(int j=0; j<b.width; j+=2){
                    int pos = j * b.width + i;
                    set(j, i, b.pixel[pos]);
                }
        }
}
```



反転：Inverse

```
PImage b;

void setup(){
	b = loadImage("woman.jpg");
	size(b.width, b.height);
}

void draw()
{
	color c;
	image(b, 0, 0);

	for(int y=0; y<b.height; y++){
		for(int x=0; x<b.width; x++){
			c = color(255-red(get(x,y)), 255-green(get(x,y)), 255-blue(get(x,y)));
			set(x, y, c);
		}
	}
}
```

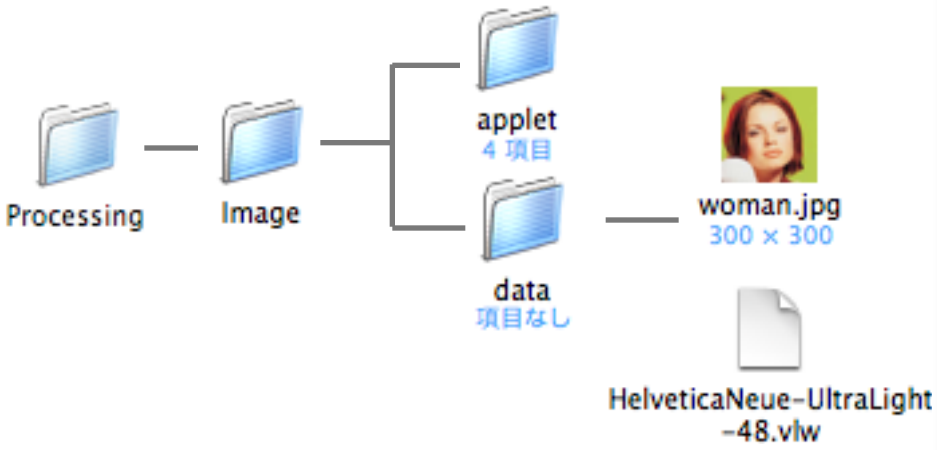
2値化：Binarization

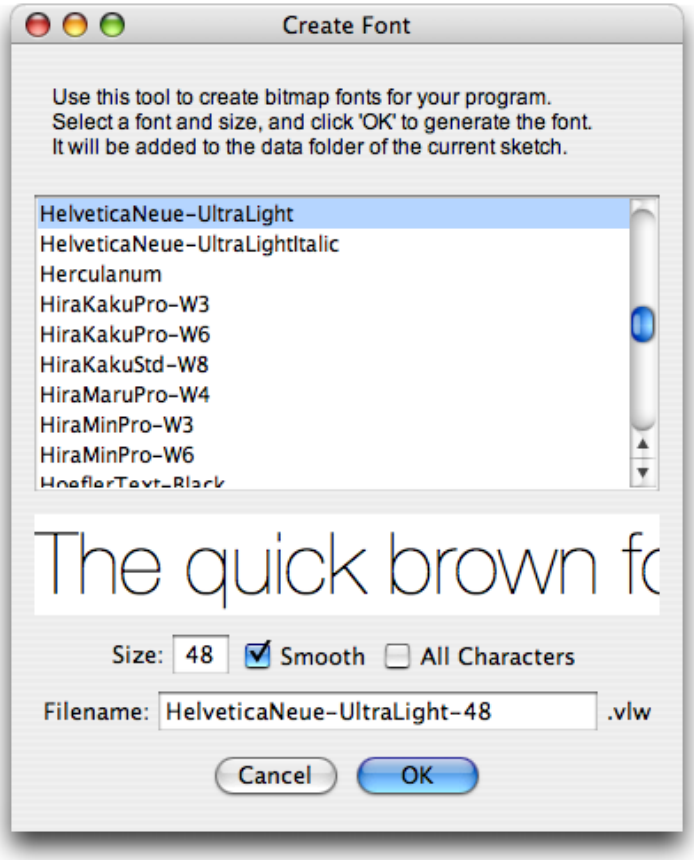
```
PImage b;
void setup(){
	b = loadImage("woman.jpg");
	size(b.width,b.height);
}
void draw()
{
	color c;
	image(b, 0, 0);
	for(int y=0; y<b.height; y++){
		for(int x=0; x<b.width; x++){
			if(red(get(x,y)) > 200){
				c = color(255,255,255);
			}
			else {
				c = color(0,0,0);
			}
			set(x, y, c);
		}
	}
}
```

フォントデータのセット

1. Tools→Create Fontを選択
2. 使用したいフォントを選択しOKボタンを押す

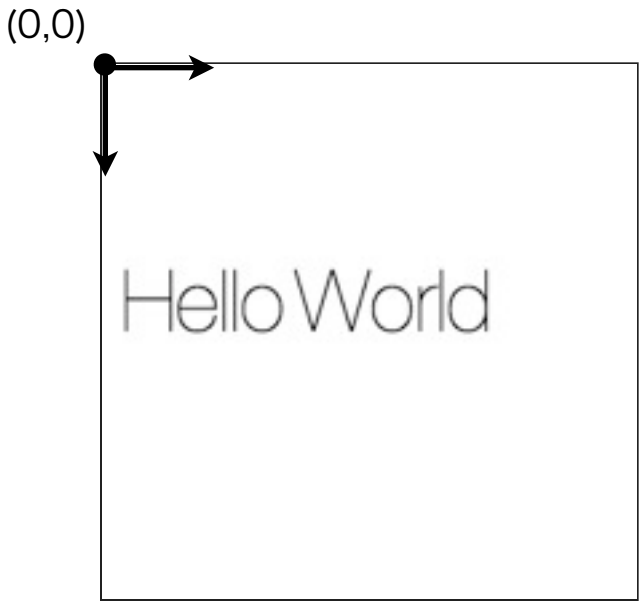
画像と同様にdataフォルダ内にフォントデータが作成される





タイポグラフィ：text( )

```
PFont f;
void setup(){
	size(300,300);
	f = loadFont("HelveticaNeue-UltraLight-48.vlw");
	textFont(f, 50);
}

void draw()
{
	background(255);
	fill(0);
	text(”Hello World”, 10, 150);
}
```



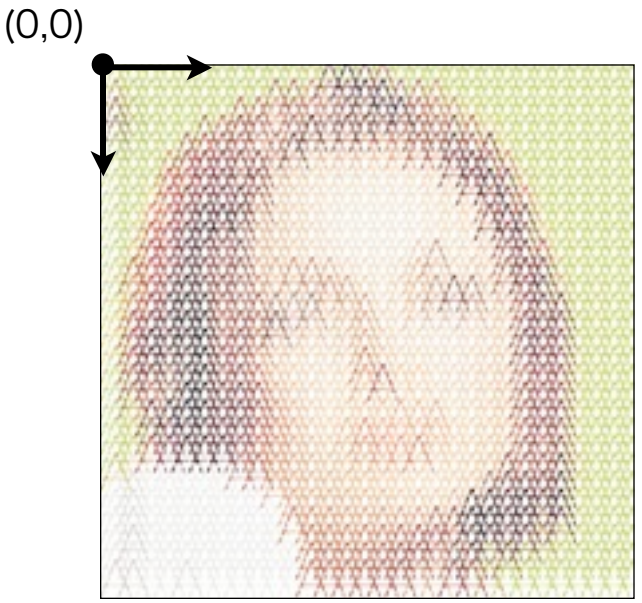
PFont f = loadFont("フォント名"); フォントファイルを変数fに格納

textFont(f, size); テキストの種類とその大きさを指定、テキストを描画する前に宣言

text("文字列", x, y); テキストを指定位置に描画

アスキーアート？

```
PImage b;
PFont f;

void setup(){
        b = loadImage("woman.jpg");
        size(b.width, b.height);
        f = loadFont("HelveticaNeue-UltraLight-48.vlw");
        textFont(f, 30);
}

void draw()
{
        background(255);
        for(int i=0; i<b.height; i+=10){
                for(int j=0; j<b.width; j+=10){
                        fill(b.get(j,i));
                          text("A", j, i+10);
                }
        }
}
```



ラプラシアンフィルタによる鮮鋭化

```
PImage a;
PImage b;

void setup(){
  a = loadImage("woman.jpg");
  b = a.copy();
  size(b.width,b.height);
}

int grey(color c){
  float r = red(c);
  float g = green(c);
  float b = blue(c);
  return((int)(0.299*r + 0.587*g + 0.114*b));
}

void draw()
{
    background(255);
    LapracianFilter(a,b);
    image(b,0,0);
}
```

```
void LapracianFilter(PImage F, PImage G)
{
  int m=1;
  int[][] H = { {0,  1, 0},
                {1, -4, 1},
                {0,  1, 0} };

   for(int i=m; i<F.height-m; i++) {
    for(int j=m; j<F.width-m; j++){
      int sum = 0;
      for(int k=-m; k<(m+1); k++) {
        for(int l=-m; l<(m+1); l++){
          sum += grey(F.get(i+k,j+l)) * H[k+m][l+m];
        }
      }
      int c = grey(b.get(i,j)) - sum;
      G.set(i,j,color(c,c,c));
    }
  }
}
```

移動平均フィルタによる平滑化

```
PImage a;
PImage b;

void setup(){
  a = loadImage("woman.jpg");
  b = a.copy();
  size(b.width,b.height);
}

int grey(color c){
  float r = red(c);
  float g = green(c);
  float b = blue(c);
  return((int)(0.299*r + 0.587*g + 0.114*b));
}

void draw()
{
    background(255);
    MovingAverageFilter(b,a,2);
    image(a, 0, 0);
}
```

```
void MovingAverageFilter(PImage F, PImage G, int m)
{
  for(int y=m; y<F.height-m; y++) {
    for(int x=m; x<F.width-m; x++) {
      int sum=0;
      int cc=0;
      for(int i=-m; i<m+1; i++){
        for(int j=-m; j<m+1; j++){
            sum += grey(F.get(x+i,y+j));
            cc++;
        }
      }
      int c = sum/cc;
      G.set(x,y,color(c,c,c));
    }
  }
}
```

画像処理結果

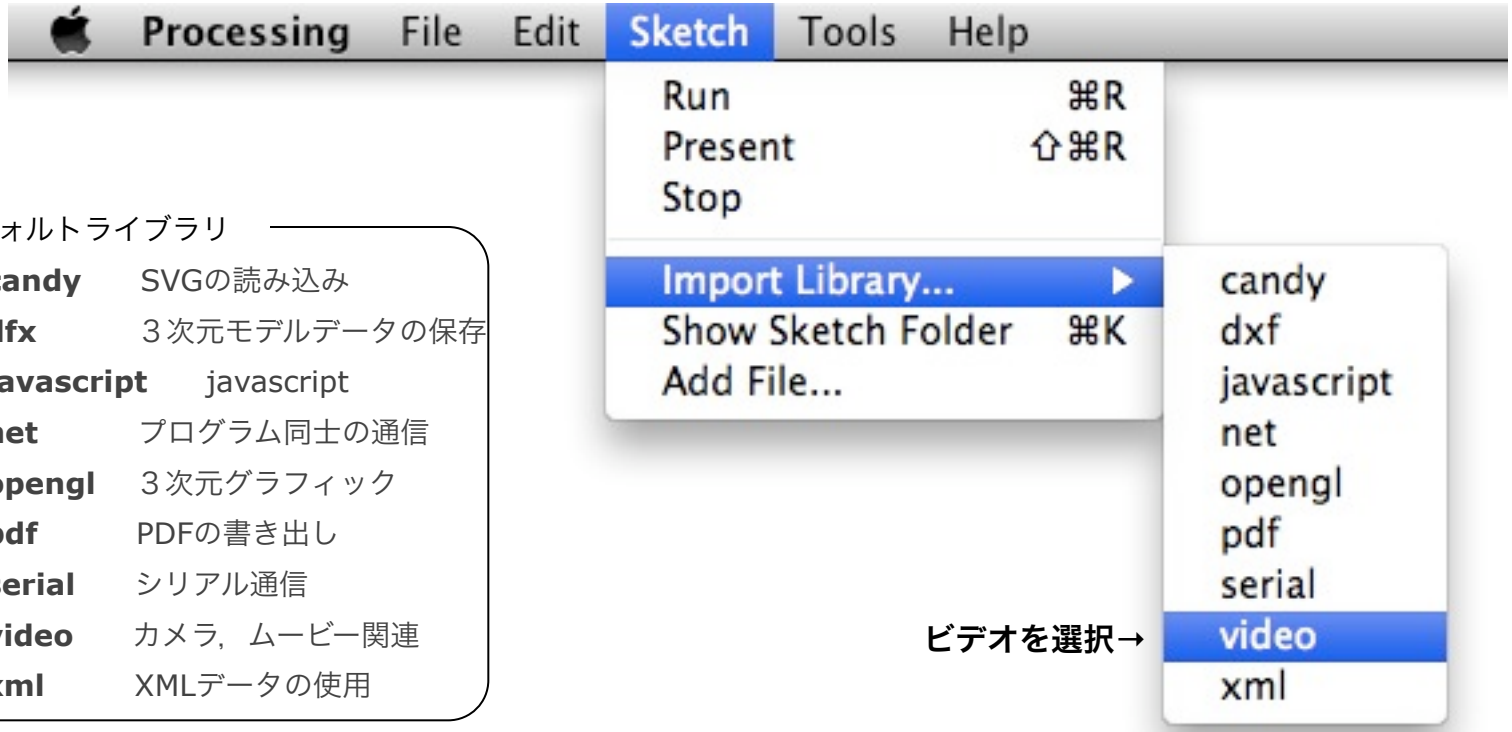
ラプラシンフィルタによる鮮鋭化

移動平均フィルタによる平滑化

01 : proce55ing
02 : image processing
→ 03 : video processing
04 : open cv

ビデオライブラリの使用

既にインストール済みのライブラリを使用する場合は，メニューから使いたいライブラリを選べば import 〜; という命令が挿入され，ライブラリを使用する準備が整う.



デフォルトライブラリ

candy	SVGの読み込み
dfx	３次元モデルデータの保存
javascript	javascript
net	プログラム同士の通信
opengl	３次元グラフィック
pdf	PDFの書き出し
serial	シリアル通信
video	カメラ，ムービー関連
xml	XMLデータの使用

カメラを使うための準備(Windowsの場合)

Windowsでvideoライブラリのキャプチャ機能を使用する場合は，VDIGソフト(QuickTimeとハードウェアの仲介役のようなもの)をインストールする必要がある.
WinVDIGというフリーウェアがあるので，下記のサイトより**バージョン1.0.1**をダウンロードし，インストールする．(他のバージョンでは不具合の可能性有り)

Improved glm code　http://www.eden.net.nz/7/20071008/

既にWinVDIGの他のバージョンがインストール済みの場合はアンインストールし，以下のファイルを削除してからバージョン1.0.1をインストールする.

windows/system32/VsDump.ax
windows/system32/QuickTime/ (フォルダの中身全て)
QuickTimeフォルダ/QTComponent/VsVDIG.qtx

カメラ映像の取得(videoライブラリ)

Capture　カメラを操作するためのオブジェクト．親オブジェクト名(通常はthis)，カメラ画像の幅・高さ，カメラデバイス名，フレームレートを指定する．

```
Capture camera = new Capture(this, width, height [, "device name"], frame rate]);
```

captureEvent()　Captureオブジェクトのイベントハンドラ名．カメラの画像が更新されるとこのイベントが呼ばれる．

read()　現在のカメラ画像を読み出す．主にイベントハンドラ内で使う．

```
void captureEvent() {
        camera.read();
}
```

stop()　キャプチャを中止する

```
camera.stop();
```

crop()　カメラ画像の一部だけを取り込むようにする．

```
camera.crop(x, y, width, height);
```

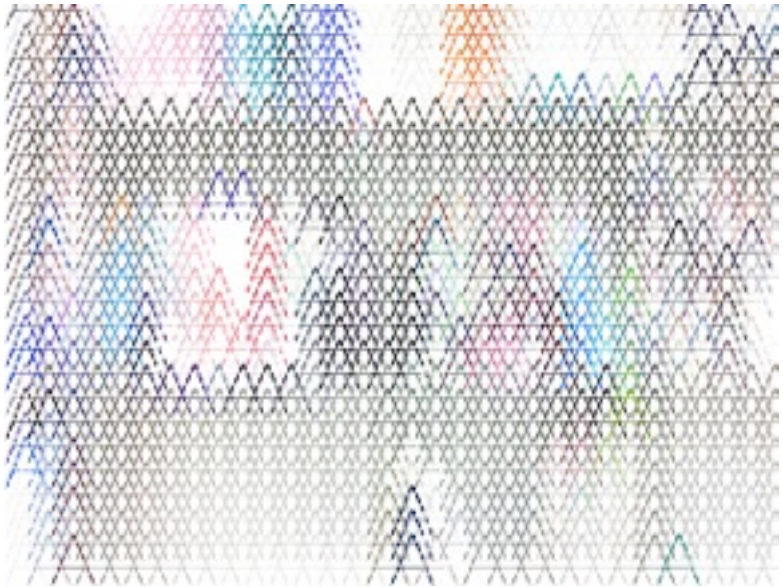
カメラ映像の表示

```
import processing.video.*;

Capture camera;

void setup(){
  size(320, 240);
  camera = new Capture(this,320,240,15);
}

void draw(){
  image(camera, 0, 0);
}

void captureEvent(Capture camera){
  camera.read();
}
```

カメラ映像によるアスキーアート？

```
import processing.video.*;

PFont f;
Capture camera;

void setup(){
  size(320, 240);
  camera = new Capture(this,320,240,15);
  f = loadFont("HelveticaNeue-UltraLight-48.vlw");
  textFont(f, 30);
}

void draw(){
  background(255);
  for(int i=0; i<camera.height; i+=10){
    for(int j=0; j<camera.width; j+=10){
      fill(camera.get(j,i));
      text("A", j, i+10);
    }
  }
}
```

```
void captureEvent(Capture camera){
  camera.read();
}
```



アルゴリズム開発には

- リアルタイム性 VS 再現性
  - リアルタイム性の評価→カメラストリーム入力
  - デバッグには再現性→連番ファイル入力

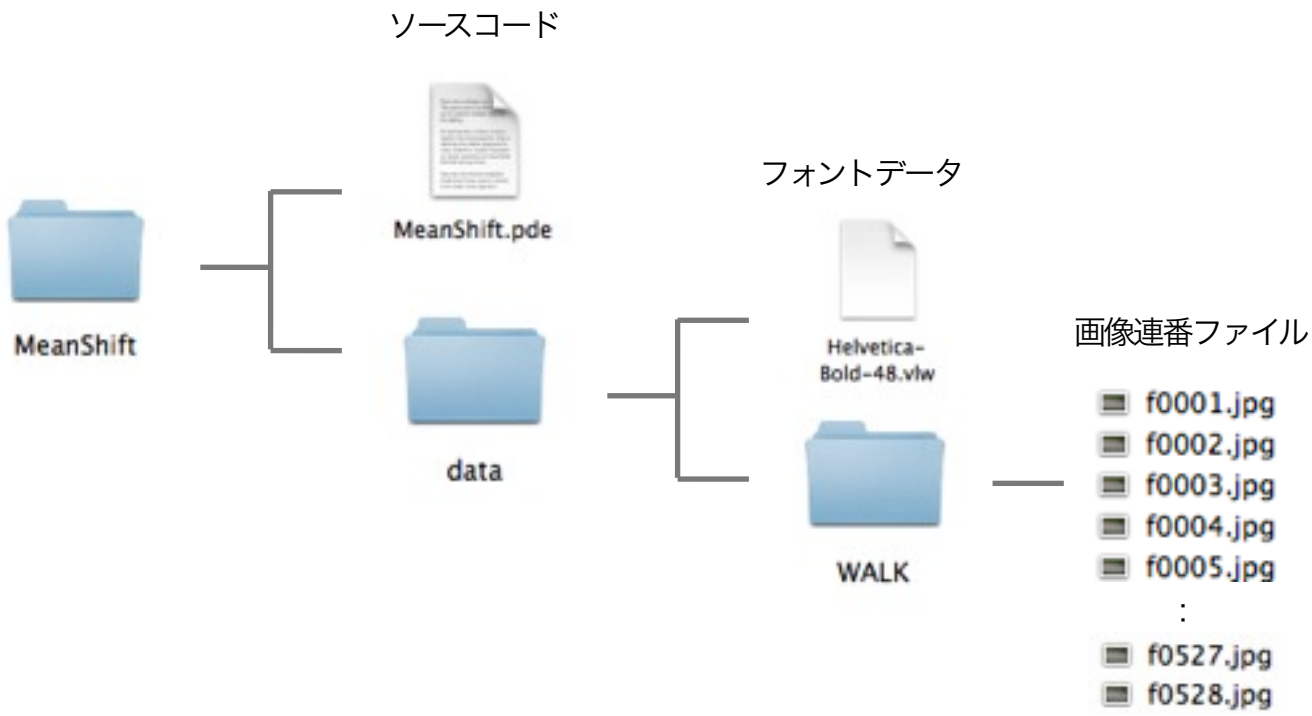
http://www.vision.cs.chubu.ac.jp/VU/

- 動画像処理アルゴリズム
  - ソースコード＆Java Applet
  - 画像連番データ
  - テキスト

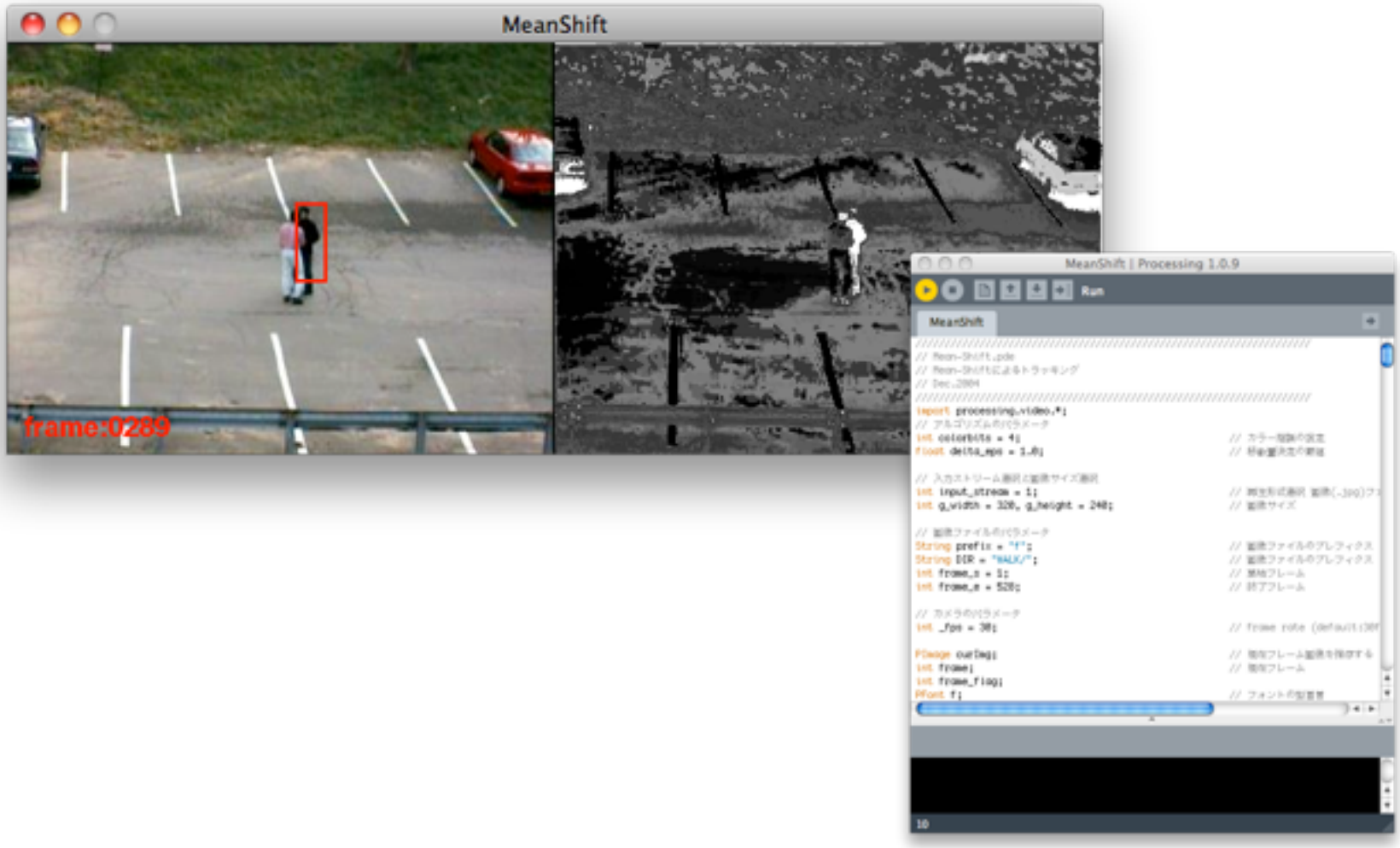
- Mean-Shift追跡のデモ＆データをダウンロード
  - http://www.vision.cs.chubu.ac.jp/~hf/MeanShift.zip

画像連番データのセット

– 画像連番ファイル：528枚 f0001.jpg - f0528.jpg



まずは実行！


MeanShift
frame:0289
MeanShift | Processing 1.0.9
Run
MeanShift

カメラ入力に変更

```
// 入力ストリーム選択と画像サイズ選択
int input_stream = 2;                    // 再生形式選択 画像(.jpg)ファイル = 1, カメラ = 2
int g_width = 320, g_height = 240;       // 画像サイズ

// 画像ファイルのパラメータ
String prefix = "f";                     // 画像ファイルのプレフィクス
String DIR = "WALK/";                    // 画像ファイルのプレフィクス
int frame_s = 1;                         // 開始フレーム
int frame_e = 528;                       // 終了フレーム
```

http://www.vision.cs.chubu.ac.jp/VU/

第2章 移動体検出

第3章 領域クラスタリング

第4章 物体追跡

第5章 画像から実画像へのマッピング

第6章 特徴量抽出

第7章 物体識別

01 : proce55ing
02 : image processing
03 : video processing
→ 04 : open cv

OpenCV Library for Proce55ing

- OpenCV release version 1.0

OpenCV
ROI()
absDiff()
allocate()
blobs()
blur()
brightness()
capture()
cascade()
contrast()
convert()
copy()
detect()
flip()
image()
interpolation()
invert()
jump()
loadImage()
movie()
pixels()
read()
remember()
restore()
stop()
threshold()

インストール手順

1. http://ubaa.net/shared/processing/opencv/から、ライブラリをダウンロード
    1. For Windows, download the OpenCV release version 1.0
    2. For MacOS X, dowload the opencv-framework-1.1.dmg

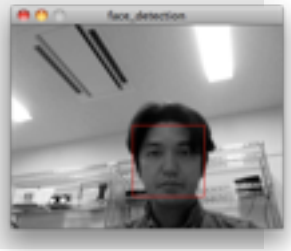
2. OpenCVライブラリをプロセッシングのライブラリフォルダの移動

3. サンプル：OpenCV Processing examples をダウンロード

face_detect

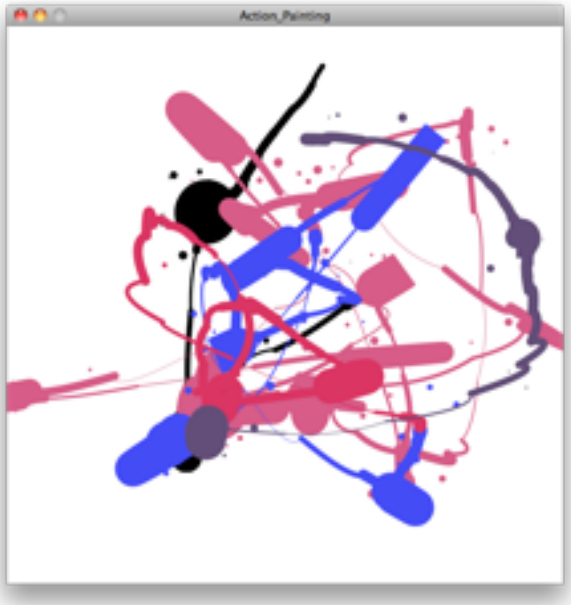
```
import hypermedia.video.*;
import java.awt.Rectangle;


OpenCV opencv;

void setup() {

  size( 320, 240 );
  opencv = new OpenCV( this );
  opencv.capture( width, height );
  opencv.cascade(
        OpenCV.CASCADE_FRONTALFACE_ALT );

}

public void stop() {
  opencv.stop();
  super.stop();
}
```

```
void draw() {

  // grab a new frame
  opencv.read();

  // proceed detection
  Rectangle[] faces = opencv.detect( 1.2, 2,
  OpenCV.HAAR_DO_CANNY_PRUNING, 40, 40 );

  // display the image
  image( opencv.image(), 0, 0 );

  // draw face area(s)
  noFill();
  stroke(255,0,0);
  for( int i=0; i<faces.length; i++ ) {
    rect( faces[i].x, faces[i].y,
          faces[i].width, faces[i].height );
  }
}
```



おまけ：PDFによる書き出し

```
import processing.pdf.*;

void setup()
{
  省略
  beginRecord(PDF, "Draw.pdf");
}

boolean ON=false;

void draw()
{
  省略
}

void keyPressed(){
  if(key == 's'){
    endRecord();
    exit();
  }
}

void mousePressed(){  省略  }

void mousePressed(){  省略  }
```



出力したPDFファイル

import processing.pdf.*;　　PDFライブラリを使う

beginRecord();　　PDFファイルに記録開始
endRecord();　　PDFファイルへの記録終了

keyPressed()　　キーが押されたときに実行される