akichJ 0.4.0 リファレンス

core パッケージ
Stdim クラス
画像データの保持と画像データに関するメソッドを提供するクラス

クラスのメンバは以下

BufferedImage img :画像本体
int ID:画像の ID(今後のバージョンアップで使用される予定)

下記の readim()メソッドによって代入できる。
代入による初期化をしない場合は、null を代入をすると意図しない動作が起こる可能性があるので、必ず
Stdim 変数名 = new Stdim();
というようにメモリをしっかり確保すること。

setImg()メソッド
Stdim オブジェクトに BufferedImage オブジェクトをセットするメソッド

書式
(Stdim オブジェクト).setImg((BufferedImage img);

引数
BufferedImage img:Stdim オブジェクトにセットしたい BufferedImage オブジェクト

返り値
void

getImg()メソッド
Stdim の画像データを BufferedImage として返すメソッド

書式
(Stdim オブジェクト).getImg();

引数
なし

返り値
BufferedImage

getWidth()メソッド
Stdim オブジェクトに格納されている画像データの幅を返すメソッド

書式
(Stdim オブジェクト).getWidth();

引数
なし

返り値
int

getHeight()メソッド
Stdim オブジェクトに格納されている画像データの高さを返すメソッド

書式
(Stdim オブジェクト).getHeight();

引数
なし

返り値
int

getType()メソッド
画像のタイプを返すメソッド

書式
(Stdim オブジェクト).getType();

引数
なし

返り値
int

getID()メソッド
画像に指定されている ID を返す関数

書式
(Stdim オブジェクト).getID();

引数
なし

返り値
int

setID()メソッド
画像に指定されている ID をセットするメソッド

書式
(Stdim オブジェクト).setID(int id);

引数
int

返り値
void

Core クラス
低水準のメソッド群

readStdim()メソッド
画像を読み込み Stdim オブジェクトに格納するメソッド

書式
Core.readStdim(String filename);

引数
String filename:ロードしたい画像の名前

返り値
Stdim

readNImg()メソッド

Core.readStdim(String filename);
画像を読み込み NImg オブジェクトに格納するメソッド

引数
String filename:ロードしたい画像の名前

返り値
NImg

saveim メソッド
画像を保存するメソッド

書式
saveim(Stdim img,String filename,String format);

引数
Stdim img:保存したい画像
String filename:保存するときの名前（ディレクトリも含む）

String format:保存する画像の拡張子(png や jpg)

返り値
void

freeim()メソッド
Stdim オブジェクトのメモリを開放するメソッド

書式
Core.freeim(Stdim img);

引数
Stdim img:メモリを開放したい Stdim オブジェクト

返り値
void

copyim()メソッド
Stdim オブジェクトのコピーを作るメソッド(値渡し)

書式
Core.copyim(Stdim… args);

引数
Stdim の可変長引数を取る。
第一引数を元に、第二引数、第三引数、第四引数...にコピー(値渡し)を作る

返り値
void

cvtStdim2BI()メソッド
Stdim オブジェクトを BufferedImage オブジェクトに変換するメソッド

書式

Core.cvtStdim2BI(Stdim img);

引数
Stdimg img:BufferedImage オブジェクトに変換したい Stdim オブジェクト

返り値
BufferedImage

cvtNImg2BI()メソッド
NImg オブジェクトを BufferedImage オブジェクトに変換するメソッド

書式
Core.cvtNImg2BI(NImg img);

引数
NImg img:BufferedImage オブジェクトに変換したい NImg オブジェクト

返り値
NImg

Color クラス
色を保持するクラス。基本的にはオブジェクトを作らず、コンストラクタメソッドを使い、引数に Color オブジェクトを取るメソッドに渡す。

例
Example.function(Color(150,200,100));
Color クラスのコンストラクタメソッドの引数は int 型変数3つで、RGB の順番で渡す。
上の例の場合は、赤が 150、緑が 200、青が 100 になる。

Color クラスメンバ

```
class Color{
	private int red;
	private int green;
	private int blue;
	public Color(int red,int green,int blue){
		this.red = red;
		this.green = green;
		this.blue = blue;
	}
	public int getR(){
		return this.red;
	}
	public int getG(){
		return this.green;
	}
	public int getB(){
		return this.blue;
	}
}
```

Point クラス

座標を保持するクラス。Color クラスと同様に、基本的にはオブジェクトを作らずコンストラクタメソッドを使い、引数に Point オブジェクトを取るメソッドに渡す。

例

```
Example.function(Point(10,20));
```

Point クラスのコンストラクタメソッドは int 型変数2つで x 座標、y 座標の順番で渡す。

上の例の場合、x 座標が 10、y 座標が 20 になる。

Point クラスメンバ

```
class Point{
	int x;
```

```
        int y;
        public Point(int x,int y){
                this.x = x;
                this.y = y;
        }
        public int getX(){
                return this.x;
        }
        public int getY(){
                return this.y;
        }
}
```

Size クラス

画像のサイズを保持するクラス。Color、Point クラスと同様に、基本的にはオブジェクトを作らずコンストラクタメソッドを使い、引数に Point オブジェクトを取るメソッドに渡す。

例

Example.function(Size(200,100));

Size クラスのコンストラクタメソッドの引数は int 型変数2つで、width,height の順番で渡す。

上の例の場合は、width が 200、height が 100 になる。

Size クラスメンバ

```
class Size{
    int height;
    int width;
    public Size(int width,int height){
        this.height = height;
        this.width = width;
    }
    public int getHeight(){
        return this.height;
```

```
    }
    public int getWidth(){
        return this.width;
    }
}
```

ImageProcessing クラス
画像処理を行うメソッド群

negp()メソッド
ネガティブ変換を行うメソッド

書式
ImageProcessing.negp(Stdim imdate);

引数
Stdim imdate:ネガティブ変換を行いたい画像データ

返り値
void

blend()メソッド
２つの画像をブレンドするメソッド

書式
ImageProcessing.blend(Stdim im1,Stdim im2, int imp1,int imp2);

引数
Stdim im1:ブレンドを行いたい画像1枚目
Stdim im2:ブレンドを行いたい画像 2 枚目
int imp1:画像一枚目の強調度
int imp2:画像二枚目の強調度
(imp:imp2 の割合でブレンドされます。)

返り値
Stdim

gray()メソッド
画像をグレースケール変換するメソッドです。

書式
ImageProcessing.gray(Stdim img);
Stdim img:グレースケール変換をしたい画像
返り値は void

twoway()メソッド
画像を閾値変換するメソッドです。

書式
ImageProcessing.twoway(Stdim img,int value,int select);

引数
Stdim img:閾値変換を行いたい画像
int value:閾値処理の補正値 0~255
int select:
int select:0 を渡すと、通常の閾値変換。1 を渡すとネガティブ変換も
同時に行います。

返り値
void

smooth()メソッド
画像を平滑化するメソッド

書式
ImageProcessing.smooth(Stdim img);

引数
Stdim img:平滑化処理をしたい画像

返り値
void

sharping()メソッド
画像を鮮鋭化するメソッドです。

書式
ImageProcessing.sharping(Stdim img,int times);

引数
Stdim img:鮮鋭化を行いたい画像
int times;鮮鋭化処理の補正値

返り値
void

edge()メソッド
画像のエッジ抽出を行うメソッドです。

書式
ImageProcessing.edge(Stdim img,int value,int select);

引数
Stdim img:エッジ抽出を行いたい画像
int value:エッジ抽出の補正値(下記の説明)
int select:エッジ抽出の種類の選択(下記の説明)

int value について
引数 value の値が大きくなればなるほど弱いエッジは抽出されなくなり、より強いエッジが抽出されるようになります。

int select について
この引数に ImageProcessing.LONG_EDGE と渡すと、縦エッジ抽出を行います。
この引数に ImageProcessing.BESIDE_EDGE と渡すと、横エッジ抽出を行います。
この因数に ImageProcessing.ALL_EDGE と渡すと、全てのエッジを抽出を行います。

solari()メソッド
画像をソラリゼーション変換するメソッドです。

書式
ImageProcessing.solari(Stdim img);

引数
Stdim img:ソラリゼーション変換を行いたい画像

返り値
void

post()メソッド
画像をポスタリゼーション変換するメソッドです。

書式
ImageProcessing.post(Stdim img);

引数
Stdim img:ポスタリゼーション変換をしたい画像

返り値
void

separ()メソッド
画像を RGB 値で分離するメソッドです。

書式
ImageProcessing.separ(int select, Stdim… args);

引数
int select:分離したい RGB を選びます(下に詳しい説明)
Stdim:これは可変長引数で、分離するときのベースになる画像を一番目に、
その後は R,G,B の順番にこのメソッドが吐き出す Stdim を受け取る、Stdim オブジェクトを渡してください。

Int select の詳しい説明

ImageProcessing.ONRY_RED:これを渡すと、画像を赤だけに分離した画像を返します。

ImageProcessing.ONLY_GREEN:これを渡すと、画像を緑だけに分離した画像を返します。

ImageProcessing.ONLY_BLUE:これを渡すと、画像を青だけに分離した画像を返します。

ImageProcessing.RED_GREEN:これを渡すと、画像を赤と緑に分離した2枚の画像を返します。

ImageProcessing.RED_BLUE:これを渡すと、画像を赤と緑に分離した2枚の画像を返します。

ImageProcessing.GREEN_BLUE: これを渡すと、画像を緑と青に分離した2枚の画像を返します。

ImageProcessing.ALL_COLOR: これを渡すと、画像を赤と緑と青に分離した3枚の画像を返します。

コード例

```
//画像を赤、緑、青の三色に分離します
public class example {
	public static void main(String[] args){
		Stdim img =  Core.readim(“example.png”);
		Stdim red = new Stdim();
		Stdim green = new Stdim();
		Stdm blue = new Stdim();
		ImageProcessing.separ(ImageProcessing.ALL_COLOR,
		red, green, blue);
		Stdgui.showim(img);
		Stdgui.showim(red);
		Stdgui.showim(green);
		Stdgui.showim(blue);
	}
}
```

返り値

void

line()メソッド
画像に線を引くメソッドです。

書式
ImageProcessing.line(Stdim img, Point st, Point ed, Color col, int r);

引数
Stdim img:線を引きたい画像
Point st:線を引き始める座標
Point ed:線を引き終わる座標
Color col:線の色
int r:線の太さ

返り値
void

bright()メソッド
画像の明度を操作するメソッドです。

書式
ImageProcessing.bright(Stdim img,int pixel);

引数
Stdim img:明度を操作したい画像
int pixel:何ピクセル調整するか(-255 ~ 255)

返り値
void

mix()メソッド
RGB の値をミックスするメソッドです。

書式
ImageProcessing.mix(Stdim img,int select);

引数
Stdim img:RGB 値をミックスしたい画像
int select:ミックスの仕方の指定(下に詳しい説明)

int select について
通常の並びを RGB とすると、以下の引数で数通りのミックス方法を使えます

MIX_GBR:GBR
MIX_GRB:GRB
MIX_BGR:BGR
MIX_BRG:BRG
MIX_RGB:RGB(元画像と変わらない)
MIX_RBG:RBG
返り値
void

flip()メソッド
画像を回転させるメソッドです。

書式
ImageProcessing.flip(Stdim img,int select);

引数
Stdim img:回転させたい画像
int select:回転の種類(下に詳しい説明)

int select について
FLIP_180:画像を 180 度回転
FLIP_MIRROR:画像を鏡に写したように回転
FLIP_90LEFT:左回りに 90 度回転
FLIP_90RIGHT 右回りに 90 度回転
FLIP_180MIRROR:画像を 180 度回転し、それを鏡に写したように回転

返り値

void

RGBavr()メソッド
画素値の平均を返すメソッドです。

書式
ImageProcessing.RGBavr(Stdim img);

引数
Stdim img:画素値の平均を求めたい画像

返り値
float[3]

float[0]:赤の平均値
float[1]:緑の平均値
float[2]:青の平均値

sum_twoims()メソッド
2つの画像の和差の画像を返すメソッド。

書式
ImageProcessing.sum_twoims(Stdim img1,Stdim img2,Stdim dst,int select);

引数
Stdim img1:足される画像または引かれる画像
Stdim img2:足す画像または引く画像
Stdim dst:画像の和差が吐き出される Stdim オブジェクト
int select:和を求めるのか差を求めるかを指定する(下に詳しい説明)

int select について
IMAGE_ADDITION:画像の和を求める
IMAGE_SUBTRACTION:画像の差を求める

返り値
void

mask()メソッド
指定した画素値に該当するピクセルをすべて黒で塗りつぶすメソッド

書式
ImageProcessing.mask(Stdim img,Color st,Color ed);

引数
Stdim img:マスク処理を行いたい画像
Color st:色の範囲の指定(1)
Color ed:色の範囲の指定(2)

返り値
void

outline()メソッド
画像の輪郭を好きな色で抽出するメソッド

書式
ImageProcessing.outline(Stdim img, int value,Color col);

引数
Stdim img:輪郭を抽出したい画像
int value:輪郭抽出の補正値（下に詳しい説明）
Color col:輪郭を抽出するときの色

int value について
大きい値を与えれば与えるほど、濃い輪郭が抽出され、薄い輪郭は抽出されません。逆に小さい値を与えれば与えるほど、薄い輪郭も抽出されるようになります。

返り値
void

rect()メソッド
指定した矩形部分の RGB 値を変化させるメソッド

書式
ImageProcessing.rect(Stdim img, Point st, Point ed, Color col);

引数
Stdim img:指定した矩形部分の RGB 値を変化させたい画像
Point st:矩形の左上の座標
Point ed:矩形の右下の座標
Color col:変化させたい RGB 値

返り値
void

hough()メソッド
画像をハフ変換するメソッド
(注意:このメソッドはまだ不完全で処理にかなり時間がかかってしまいます)

書式
ImageProcessing.hough(Stdim img, Color col);

引数
Stdim img:ハフ変換を行いたい画像
Color col:ハフ変換を行った際に、引く直線の色

返り値
void

zero()メソッド
画像の RGB 値を 0 で埋めるメソッド

書式
ImageProcessing.zero(Stdim img);

引数
Stdim img:RGB 値を 0 で埋めたい画像

返り値
void

sepia()メソッド
画像をセピア変換するメソッド

書式
ImageProcessing.sepia(Stdim img);

引数
Stdim img:セピア変換を行いたい画像

返り値
void

noise()メソッド
画像のノイズを消すメソッド

書式
ImageProcessing.noise(Stdim img);

引数
Stdim img:ノイズを消したい画像

返り値
void

thin()メソッド
線細化を行うメソッド

書式

ImageProcessing.thin(Stdim img, int select);

引数
Stdim img:ノイズを消したい画像
int select:最後に画像をネガティブ変換を行うかどうか
(ImageProcessing.THIN_WHITELINE か
ImageProcessing.THIN_BLACKLINE をわたし、前者は行い、後者は
行わない)

返り値
void

psecol()メソッド
画像を擬似カラー化するメソッド

書式
ImageProcessing.psecol(Stdim img);

引数
Stdim img:擬似カラー化したい画像

返り値
void

cut()メソッド
画像を指定されたサイズにカットするメソッド

書式
ImageProcessing.cut(Stdim img, Size size);

引数
Stdim img:カットしたい画像
Size size:カット後のサイズ

返り値

void

circle()メソッド
画像に指定されたサイズの円を描画するメソッド

書式
ImageProcessing.circle(Stdim img, Color col, Point center, int r, int thin);

引数
Stdim img:円を描画したい画像
Color col:描画する円の色
Point center:円の中心座標
int r:円の半径
int thin:円の太さ

返り値
void

Matrix クラス
画像を一つの配列として扱うクラス
Stdim オブジェクトよりも処理速度が速く低リソース

Matrix クラスメンバ

```
////////////////////////////////////////////////////////////////////////////////////////////////////
    private float[][][] mat;
    private int width;
    private int height;
    private int channels;
    private int depth;
////////////////////////////////////////////////////////////////////////////////////////////////////
```

Matrix オブジェクトのコンストラクタメソッド
Matrix オブジェクトのコンストラクタメソッドは 2 つ存在する。
1つ目(普通のコンストラクタ)
Matrix(int width,int height,int channels,int depth);
引数
int width:作成したい画像の幅
int height:作成した画像の高さ
int channels:画像のチャンネル数
int depth:画像を構成する画素値の型(今後更新予定)

2つ目(Stdim オブジェクトからの変換)
Matrix(Stdim img);

引数
Stdim img:Matrix に変換したい Stdim オブジェクト

その他のメソッド

MatAvg()メソッド
Matrix オブジェクトに保存されている画像の画素値の平均を float の配列で返すメソッド

書式

```
float[] sample = new float[4];
Stdim img = Core.readim(“Example.png”);
Matrix mat = new Matrix(img);
sample = mat.MatAvg();
```

返り値
float[]

quarterAvg()メソッド
画像の 1/4 の部分ごとの画素値の平均値を返すメソッド

書式
Matrix.quaeterAvg(Matrix mat);

引数
Matrix mat:画素値の平均を求めたい画像

返り値

float[4]
通常、要素数が 0,1,2,3 の順に、右上、左上、左下、右下の平均値が入っている。

mt パッケージ

NImg クラス
マルチスレッド処理に対応し Stdim オブジェクト、Matrix オブジェクトよりも速いが、画像の入出力が複雑で、速度は Stdim オブジェクトに劣る

NImg クラスメンバ

//

int[][][] image　　//画像
int height;　　//画像の高さ
int width;　　//画像の幅
boolean wait;　　//待機中か否か
short wait_ID;　　//他の画像スレッドとの連携のための ID
Deque<Order> order;　　//画像処理方法を登録しておくデキュー)
※Order クラスについては下に仕様が書いてあります
Execute ex;　　//画像スレッド

//

NImg のコンストラクタメソッドは 2 つ存在します
1 つ目:サイズを指定しオブジェクトを生成
書式
NImg img = new NImg(int width, int height, int depth);

引数
int width:画像の幅
int height:画像の高さ
int depth:RGB を扱う場合は 3、グレースケール画像を扱う場合は 1

2 つ目:元になる BufferedImage オブジェクトを用い、オブジェクト生成

書式
NImg img = new NImg(BufferedImage im);

引数
BufferedImage im:NImg オブジェクトのもとになる画像

NImg オブジェクト用画像処理メソッド
NImg オブジェクトの画像処理メソッドはメンバ関数として定義されているため、呼び出すときは
(NImg オブジェクト).メソッド名(引数);
このように呼び出す。

SI 系メソッド
(実際には実行せず登録のみし、後で Schedule.execute()メソッドを呼び出し、マルチスレッドで画像処理を実行する)

SInega()メソッド
NImg オブジェクトにネガティブ変換をすることを登録するメソッド

書式
(NImg オブジェクト).SInega();

引数
なし

返り値
void

SIgray()メソッド
NImg オブジェクトにグレースケール変換をすることを登録するメソッド

書式
(NImg オブジェクト).SIgray();

引数
なし

返り値
void

SItwoway()メソッド
NImg オブジェクトに二値化をすることを登録するメソッド

書式
(NImg オブジェクト).SItwoway();

引数
なし

返り値
void

NI 系メソッド
(画像処理をその場で実行し、マルチスレッドに展開しない)

NInega()メソッド
NImg をその場でネガティブ変換をするメソッド

書式
(NImg オブジェクト).NInega();

引数
なし

返り値
void

NIblend()メソッド
２つの NImg をその場で合成するメソッド

書式
(NImg オブジェクト).NIblend(NImg blendim, int p1, int p2);

引数
NImg blendim:合成したい画像
int p1:呼び出した側の画像の割合
int p2:渡した画像の割合

返り値
void

NIgray()メソッド
2 つの画像をその場でグレースケール変換するメソッド

書式
(NImg オブジェクト).NIgray();

引数
なし

返り値
void

NItwoway()メソッド
2 つの画像をその場で二値化するメソッド

書式
(NImg オブジェクト).NItwoway();

返り値
void

Order クラス
NImg オブジェクトに追加される画像処理の命令をためておくクラス
普通は NImg オブジェクトの SI 系メソッドが管理するのでユーザーが
扱う必要はない

Order クラスメンバ

//

short function_code;　　//画像処理命令の種類を記憶しておく変数
Deque<Objects> args;　　//画像処理メソッドへの引数を記憶しておく変数

//

Order クラスのコンストラクタは2つ存在します

Order(short function_code);
Order(short function_code, Objects… objs);

メンバメソッド

getFunction_code()メソッド
ファンクションコードが返るメソッド

書式
(Order オブジェクト).getFunction_code();

引数
なし

返り値
short

getArgs()メソッド
args の 0 番目の要素を返し、削除するメソッド

書式
(Order オブジェクト).getArgs();

引数
なし

返り値
Object

getArgs_keep()メソッド
args の 0 番目の要素を返し、0 番目の要素は削除しないメソッド

書式
(Order オブジェクト).getArgs_keep();

引数
なし

返り値
Object

kernel パッケージ

Execute クラス
NImg の画像処理命令を実行するクラス
※基本的にプログラマが操作する必要はない

Execute メンバ
//
Nimg im:渡された NImg を一時的に参照するための NImg オブジェクト
//

Schedule クラス
画像ごとのスレッドに関するクラス

execute()メソッド
画像ごとにスレッドを生成し、NImg オブジェクトのメンバの
Deque<Order>を読み取り画像処理を行うメソッド

書式
Schedule.execute(NImg… NImgs);

引数
NImg の可変長引数
Nimg Nimg:画像処理を実行したい画像

返り値
void

gui パッケージ

GUI メソッド群
Stdgui クラス

showim()メソッド
画像を表示するメソッド

akichJ0.4.0 Testing は複数の、画像を保持できるクラスが存在するため、どのクラスでも正しく表示できるよう 2 通りにオーバーロードしています。

書式
Stdgui.showim(Stdim img, String name); //Stdim オブジェクト用
Stdgui.showim(NImg img, String name); //NImg オブジェクト用

引数

//Stdim オブジェクト用//
Stdim img:表示したい画像
String name:ウィンドウのメニューバーに表示する文字列
//

//NImg オブジェクト用//
NImg img:表示したい画像
String name:ウィンドウのメニューバーに表示する文字列
//

返り値
void

自動で画像の高さ、幅を認識しそれにあったサイズのウィンドウを生成し画像を表示します。

DrawGragh クラス

draw()メソッド
画像の RGB 値に関するグラフを描画し表示するメソッド

書式
DrawGragh.draw(Stdim img, String windowName, String graghName, boolean Gragh_Type);

引数
Stdim img:グラフを描画するための画像
String windowName:グラフを描画するウィンドウのメニューバーに表示する文字列
String graghName:描画するグラフのタイトル
boolean Gragh_Type:数値計算の方法(下に詳細)

返り値
void

boolean Gragh_Type について
DRAWGRAGH_LONG を渡すと、1つの x 座標あたりのすべての RGB 値の
平均のグラフを描画します。
DRAWGRAGH_LONG を渡すと、1つの y 座標あたりのすべての RGB 値の
平均のグラフを描画します。