2022 年度 情報メディア基礎ユニット１
項目Ｂ：ヒューマンメディア１

3週目 ImageJプラグインによる画像処理・分析

担当教員 山内、長

1. プラグインの導入

2週目に取り扱ったマクロは、ImageJに用意されている機能を自作プログラムから呼び出して実行するというものでした。 では、ImageJにはない機能についてはあきらめるしかないのでしょうか？ 答えはノーです。 専用のエディタにプラグインという形式でJavaのプログラムを書いて、 エディタのメニュー「File」-「Compile and Run」を選択すれば、 コンパイル・実行してくれます。 コンパイルされてクラスファイルができると、ImageJ起動時に自動的にメニュー「Plugins」以下のクラスはその名前が項目が追加され、 選択して実行することができるようになります。

2. プラグインの開発方法

ImageJのプラグインは、クラス内にあらかじめ決められたインターフェース通りの関数を作成するだけで完成です。 インターフェースとは、関数の型・関数名・関数の引数の型などの関数定義を内部に持っており、インターフェースを実装するクラスは内部にその定義通りの関数を必ず作成する必要があります。 ちなみに、「実装」というのはプログラムを実際に作成することです。 これを破ると、コンパイルエラーになります。 ImageJのプラグインとなるクラスは、PlugInインターフェースかPlugInFilterインターフェースを実装することになっています。 また、プラグインのクラス名には1文字以上の半角アンダースコア(_)を入れる必要があります。 Javaの場合、publicなクラス名に拡張子[.java]をつけたものがソースファイルの名前になります。 例えば、Test_1クラスを作成しようとすると、ソースファイル名はTest_1.javaになります。

2.1. PlugInインターフェースによるプラグイン

PlugInインターフェースに定義されている関数は以下のようになります。

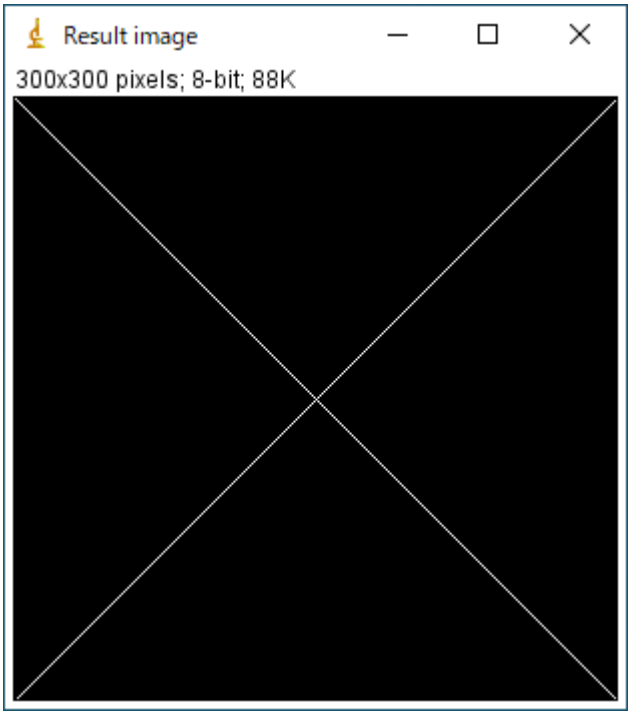
```
public void run(String arg)
```

では、ここで簡単なプラグインのサンプル(クラス名：Test_1, ソースファイル名：Test_1.java)を作ってみましょう。

```
import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.*;
import ij.plugin.frame.*;

public class Test_1 implements PlugIn {
    public void run(String arg) {
        int width = 300;
        int height = 300;
```

```
        ImageProcessor ip = new ByteProcessor(width, height);
        ip.setColor(Color.black);
        ip.fill();
        for(int x = 0; x < width; x++) {
            for(int y = 0; y < height; y++) {
                if (x == y || (width - x) == y) {
                    ip.putPixel(x, y, 255);
                }
            }
        }
        ImagePlus imp = new ImagePlus("Result image", ip);
        imp.show();
    }
}
```

ImageJのメニュー「PlugIns」-「New」-「PlugIn」を選択し、 表示されるエディタ内のプログラムを上記と同じ内容になるように編集します。 そして、ImageJシステム直下の「plugins」フォルダーに「Test_1.java」という名前で保存します。 エディタのメニュー「File」-「Compile and Run」を選択します。 コンパイルされ、エラーがなければプラグインが実行されます。 つぎのようなウィンドウが表示されるでしょう。



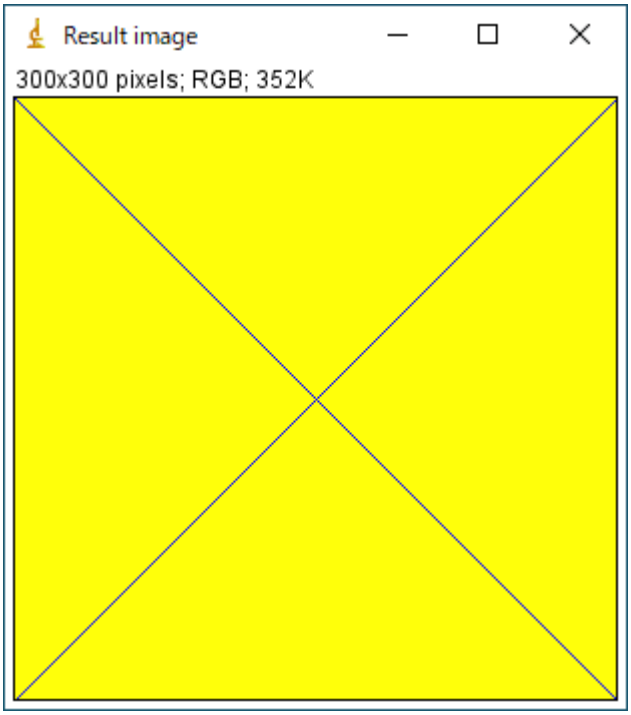
ImageJのメニュー「Help」-「Refresh Menus」を選択するか、 ImageJを再起動することにより、 「Test 1」項目が「PlugIns」メニューに追加されます。

プログラムを簡単に説明します。 ByteProcessorクラスは、ImageProcessorクラスを継承しており、 グレースケール画像の処理を行うことができます。 ここではまず、色を黒に設定し、全体を塗りつぶしています。 そして、for文とif文を用いて、対角線を1ピクセルずつ白でプロットしています。 最後の、ImagePlusクラスは画像を表現するものであり、 コンストラクタにウィンドウのタイトルとImageProcessorオブジェクトを指定し、 showメソッドを呼び出すことで処理結果を画像としてウィンドウに表示することができます。

次にこれを、背景を黄色、対角線を青で描画するように改造してみます(クラス名：Test_2, ソースファイル名：Test_2.java)。

```
import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.*;
import ij.plugin.frame.*;

public class Test_2 implements PlugIn {
    public void run(String arg) {
        int width = 300;
        int height = 300;
        ImageProcessor ip = new ColorProcessor(width, height);
        ip.setColor(new Color(255,255,10));
        ip.fill();
        Color plotColor = new Color(10, 10, 255);
        for(int x = 0; x < width; x++) {
            for(int y = 0; y < height; y++) {
                if (x == y || (width - x) == y) {
                    ip.putPixel(x, y, plotColor.getRGB());
                }
            }
        }
        ImagePlus imp = new ImagePlus("Result image", ip);
        imp.show();
    }
}
```

コンパイル・実行すると、つぎのようなウィンドウが表示されるでしょう。



プログラムを簡単に説明します。 グレースケールと異なる箇所は、ByteProcessorクラスがColorProcessorクラスになっている点、 色の指定です。 ip.setColor()はjava.awt.Colorオブジェクトを与えられるので、 R,G,Bを

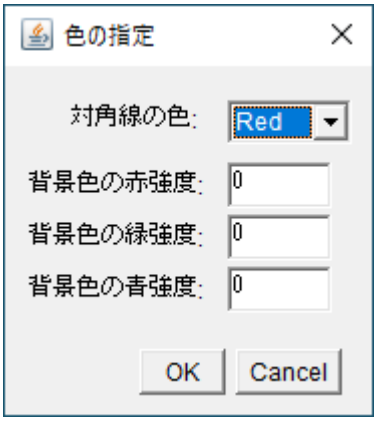
コンストラクタに与えて塗りつぶしの色を設定しています。 一方、ip.putPixel()はプロットする色を整数で与えなければならないので、 java.awt.Colorオブジェクトから色を整数で取得するgetRGB()メソッドを使用しています。

ここで、ダイアログを表示して、背景の色と対角線の色を設定できるようにしてみます。 先にプログラムを示します(クラス名：Test_3, ソースファイル名：Test_3.java)。

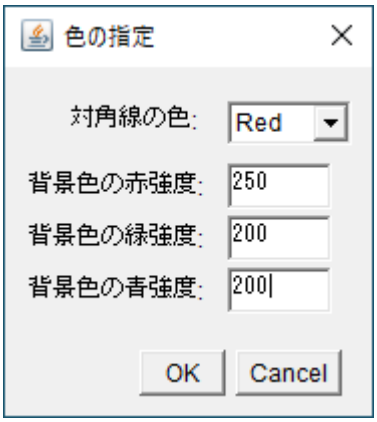
```
import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.*;
import ij.plugin.frame.*;

public class Test_3 implements PlugIn {
    boolean dialogCanceled = false;
    Color plotColor;
    int bgRed = 0;
    int bgGreen = 0;
    int bgBlue = 0;

    private void doDialog() {
        GenericDialog gd = new GenericDialog("色の指定");
        gd.addChoice("対角線の色: ", new String[]{"Red", "Green", "Blue"}, "Red");
        gd.addNumericField("背景色の赤強度:", bgRed, 0);
        gd.addNumericField("背景色の緑強度:", bgGreen, 0);
        gd.addNumericField("背景色の青強度:", bgBlue, 0);
        gd.showDialog();

        if (gd.wasCanceled()) {
            dialogCanceled = true;
        } else {
            String s1 = gd.getNextChoice();

            if (s1.equals("Red")) {
                plotColor = new Color(255, 10, 10);
            }
            else if (s1.equals("Green")) {
                plotColor = new Color(10, 255, 10);
            }
            else {
                plotColor = new Color(10, 10, 255);
            }

            bgRed = (int)gd.getNextNumber();
            bgGreen = (int)gd.getNextNumber();
            bgBlue = (int)gd.getNextNumber();
        }
    }
```

```
    public void run(String arg) {
        doDialog();
        if (dialogCanceled) return;

        int width = 300;
        int height = 300;
        ImageProcessor ip = new ColorProcessor(width, height);
        ip.setColor(new Color(bgRed, bgGreen, bgBlue));
        ip.fill();
        for(int x = 0; x < width; x++) {
            for(int y = 0; y < height; y++) {
                if (x == y || (width - x) == y) {
                    ip.putPixel(x, y, plotColor.getRGB());
                }
            }
        }
        ImagePlus imp = new ImagePlus("Result image", ip);
        imp.show();
    }
}
```

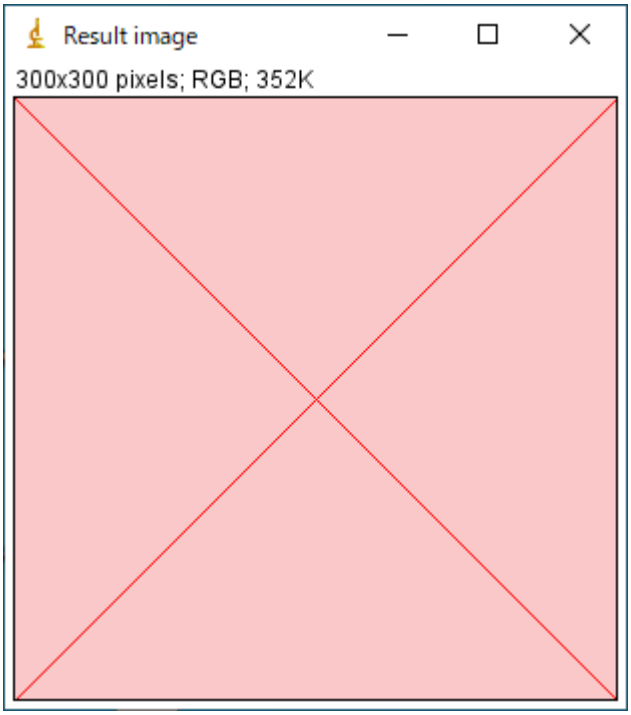
「Test 3」を実行すると、まず以下のようなダイアログが表示されます。



背景色を(250,200,200)に設定してみます。



「OK」ボタンをクリックすると、以下のような画像が生成されます。



課題1
ダイアログで画像の幅を入力させ、日の丸画像をを生成するプラグインを作成せよ。 なお、 日本の国旗 - Wikipedia によれば、 「国旗国歌法の本則における日章旗の制式は、縦横比を2対3、旗の中心（対角線の交点）を中心とし、 縦の長さの5分の3を直径（縦を2とした場合r=0.6）とした円（日章、日輪）を描き、白地に紅色の日章とするのが正式である。」 とあり、これに従って作成してください。

2.2. PlugInFilterインターフェースによるプラグイン

PlugInFilterインターフェースに定義されている関数は以下の2つになります。

```
public int setup(String arg, ImagePlus imp)
public void run(ImageProcessor ip)
```

では、ここでPlugInFilterインターフェースを用いて、ダイアログから与えた閾値によるグレースケール画像の二値化プラグイン(クラス名：Test_4, ソースファイル名：Test_4.java)を作ってみましょう。

```
import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.*;
import ij.plugin.frame.*;

public class Test_4 implements PlugInFilter {
    boolean dialogCanceled = false;
    ImagePlus imp;
    int th = 0;
```

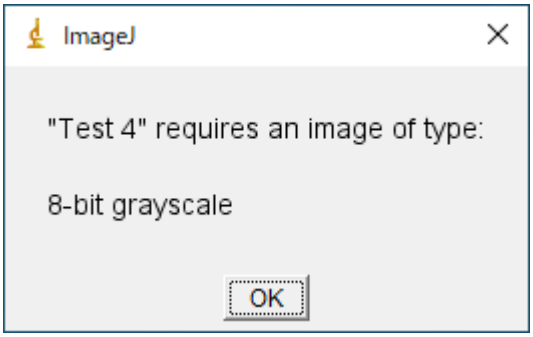
```
    private void doDialog() {
        GenericDialog gd = new GenericDialog("二値化の閾値の指定");
        gd.addNumericField("閾値:", th, 0);
        gd.showDialog();

        if (gd.wasCanceled()) {
            dialogCanceled = true;
        } else {
            th = (int)gd.getNextNumber();
        }
    }

    public int setup(String arg, ImagePlus imp) {
        this.imp = imp;
        return DOES_8G;
    }

    public void run(ImageProcessor ip) {
        doDialog();
        if (dialogCanceled) return;

        int height = ip.getHeight();
        int width = ip.getWidth();
        ImageProcessor ip2 = new ByteProcessor(width, height);
        ip2.setColor(Color.black);
        ip2.fill();

        for(int x = 0; x < width; x++) {
            for(int y = 0; y < height; y++) {
                int val = ip.getPixel(x, y);
                if (val > th) {
                    ip2.putPixel(x, y, 255);
                }
            }
        }

        ImagePlus imp2 = new ImagePlus("二値化(" + th + ")", ip2);
        imp2.show();
    }
}
```

プログラムを簡単に説明します。 まず、PlugInFilterインターフェースを用いた場合は、 処理対象の画像が読み込まれたウィンドウがアクティブになっていることが前提となります。 setup()関数の返り値がDOES_8Gとなっていますが、 これはグレースケール画像が処理対象である場合はこの後でrun()関数が呼ばれて処理が行われることを示しています。 カラー画像が処理対象である場合には、エラーメッセージを表示したダイアログを表示するだけで、 run()関数は呼ばれません。

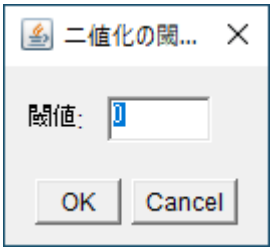
run()関数では、引数のImageProcessorオブジェクトが処理対象の画像になっています。 getWidth()メソッド、およびgetHeight()メソッドで、処理対象画像の幅と高さをそれぞれ取得することができます。 また、getPixel()メソッドでピクセル値の取得、putPixel()メソッドでピクセル値の設定ができます。

画像を開いていない状態で「Test 4」を実行すると、次のようなダイアログが表示されます。

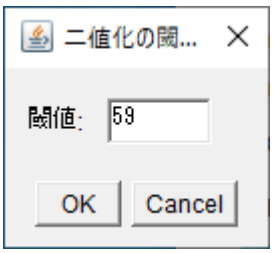
カラー画像を開いた状態で「Test 4」を実行すると、次のようなダイアログが表示されます。



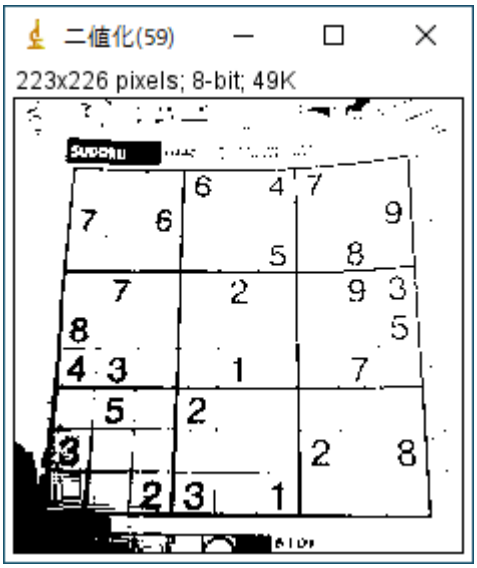
グレースケール画像を開いた状態で「Test 4」を実行すると、次のようなダイアログが表示されます。



閾値として「59」を設定します。



「OK」ボタンをクリックすると、次のような処理結果のウィンドウが表示されます。



3. 問題の続きの続き

2週目で扱った、背景の明るさが変動しているようなグレースケール画像に対して二値化を行う問題を、 ここでも扱いたいと思います。 2週目では、背景の明るさの変動を除去する前処理を検討しました。 ここでは、局所的に閾値を変化させながら二値化を行う方法を検討します。 具体的には、局所的に算出したピクセルの平均値を目安に閾値を決める方法で、 適応的二値化法などと呼ばれています。 以下に、アルゴリズムを示します。

1. 局所を二値化対象のピクセルを中心として上下左右n[pixel]の範囲とする。
2. 上記で定めた局所内のピクセル値の平均値を算出する。
3. この平均値からcを差し引いた値を閾値とし、二値化対象のピクセル値が閾値を上回っていたら255,閾値以下である場合には0を設定する。

課題2
適応的二値化法を、PlugInFilterインターフェースを実装したプラグインとして作成せよ。 ただし、ダイアログで局所を定義するn、平均値から差し引く値cを入力できるようにすること。