付録 ImageJ による画像処理の実践例

画像処理の理解を深めるため，National Institute of Health（NIH）の研究者が開発し，NIH の Website（http://rsb.info.nih.gov/ij/download.html）から無料で入手できる ImageJ（画像表示・画像処理用アプリケーションソフトウェア）を用いた画像処理の実践例を記した．ImageJ のプラグインプログラムを使用するために Java 言語（Sun Microsystems 社）が必要なので，ImageJ bundled with Java をインストールすることを勧める．

臨床画像には，日本放射線技術学会の標準デジタル画像データベース（胸部単純 X 線画像）を用いた．この画像データベースは，日本放射線技術学会のホームページの www.jsrt.or.jp/web_data/english03.php で登録して無料ダウンロードするか，日本放射線技術学会叢書「標準ディジタル画像データベース［胸部腫瘤陰影像（CD-ROM）4 枚 組 ］（www.jsrt.or.jp/web_data/system/fws/ss_list.cgi?detail_view=on&record_number=3&category_number=ss_data01.csv）を購入するかのどちらかで入手できる．

標準デジタル画像データベースの胸部 X 線画像は，マトリックスサイズ 2,048×2,048，量子化ビット数 12 ビット，白黒反転像（白＝0，黒＝4,095）であるが，この付録では，処理速度やノート PC への表示を考慮して，マトリックスサイズ 512×512，量子化ビット数 10 ビット，白黒反転処理をした画像（白＝1,023，黒＝0）を用いる．

ImageJ で標準デジタル画像データベースの Raw 画像データを読み込むには，File → Import → Raw を選択して開きたいファイルを選んだ後，Import ダイアログに上から順に 16-bit Unsigned,2048,2048,0,1,0 と入力し，White is Zero のボックスをチェックしてから，OK ボタンを押せばよい．この原画像を，マトリックスサイズ 512×512，量子化ビット数 10 ビット，白黒反転（白＝1,023，黒＝0）の画像に変換するには，付録 1 のマクロプログラム（1）2048 to 512 を実行すればよい．

付録 1

このマクロプログラムは，次のようにして実行することができる．

・画像を読み込む．
・テキスト形式で下記のマクロプログラムを入力し，Startupmacro.txt で フ ォ ル ダ C:￥Program Files￥ImageJ￥macros に保存する．ただし，ImageJ をインストールする際に特定のディレクトリを選んだ場合は，そのディレクトリ内の macros フォルダ内に保存する．
・Plugins → Macros を選ぶと下部に読み込んだマクロのリストが表示されるので，実行したいマクロを選んで実行する．

```
// Startupmacro.txt
macro "-" {} // separator
macro "(1) 2048 to 512" {
   run("Scale...", "x = 0.25 y = 0.25 width = 512 height
   = 512 create title = cln_512.img");
   run("Divide...", "value = 4.001");
   run("Invert");
}
macro "-" {} // separator
macro "(2) Linear LUT" {
      reds = newArray(256);
      greens = newArray(256);
      blues = newArray(256);
      for (i = 0; i < 256; i ++){
            reds[i]= 0;
               greens[i]= 0;
               blues[i]= 0;
         if (i >= 140 && i < 225){
               reds[i]= 3 * i-420;
               greens[i]= 3 * i-420;
               blues[i]= 3 * i-420;
         }
         else if(i >= 225) {
               reds[i]= 255;
               greens[i]= 255;
               blues[i]= 255;
         }
      }
      setLut(reds, greens, blues);
}
macro "(3) Invert LUT"{
      reds = newArray(256);
      greens = newArray(256);
      blues = newArray(256);
      getLut(reds, greens, blues);
      for (i = 0; i < 256; i ++){
            reds[i] = 255-reds[i];
            greens[i] = 255-greens[i];
            blues[i] = 255-blues[i];
      }
      setLut(reds, greens, blues);
}
macro "(4) SigmoidLUT " {
      reds = newArray(256);
      greens = newArray(256);
      blues = newArray(256);
      for (i = 0; i < 256; i ++){
               sig = 1/(1 + exp(-(i-180)/20.0));
               reds[i]= round(sig * 255 + 0.5);
               greens[i]= round(sig * 255 + 0.5);
               blues[i]= round(sig * 255 + 0.5);
      }
      setLut(reds, greens, blues);
}
macro "(5) Reset LUT" {
      reds = newArray(256);
```

```
    greens = newArray(256);
    blues = newArray(256);
    for (i = 0; i < 256; i ++){
        reds[i]= i;
        greens[i]= i;
        blues[i]= i;
    }
    setLut(reds, greens, blues);
}
macro "-" {} // separator
macro "(6) Sharpen with Laplacian" {
run("Duplicate...", "title = org.img");
run("32-bit");
run("Duplicate...", "title = Laplacian.img");
run("Convolve...", "text1 =[0 1 0¥n1 -4 1¥n0 1 0¥n]");
run("Enhance Contrast", "saturated = 0.5");
imageCalculator ("Subtract create", "org. img",
"Laplacian.img");
rename("Sharp.img");
run("Enhance Contrast", "saturated = 0.5");
selectImage(2);
close();
run("Tile");
}
macro "-" {} // separator
macro "(7) Lowpass filtering" {
run("Duplicate...", "title = org.img");
run("32-bit");
run(" Colors...", " foreground = white background =
black selection = yellow");
run("FFT");
run("Specify...", "width = 64 height = 64 x = 256 y =
256 oval centered");
run("Clear Outside");
run("Select All");
run("Inverse FFT");
selectImage(2);
close();
run("Tile");
}
macro "(8) Highpass filtering" {
run("Duplicate...", "title = org.img");
run("32-bit");
run(" Colors...", " foreground = white background =
black selection = yellow");
run("FFT");
run("Specify...", "width = 64 height = 64 x = 256 y =
256 oval centered");
run("Clear");
run("Inverse FFT");
selectImage(2);
close();
run("Tile");
}
macro "(9) Bandpass filtering" {
run("Duplicate...", "title = org.img");
run("32-bit");
run(" Colors...", " foreground = white background =
black selection = yellow");
run("FFT");
run("Specify...", "width = 16 height = 16 x = 256 y =
256 oval centered");
run("Clear");
run("Specify...", "width = 48 height = 48 x = 256 y =
256 oval centered");
run("Clear Outside");
run("Select All");
run("Inverse FFT");
selectImage(2);
close();
run("Tile");
}
macro "-" {} // separator
macro "-" {} // separator
macro "(10) Labeling and Feature extraction" {
run("Duplicate...", "title = org.img");
run("Duplicate...", "title = input.img");
run("32-bit");
run("Make Binary")
run("Set Measurements...", "area mean standard min
centroid perimeter circularity feret's skewness kurtosis
display redirect = org.img decimal = 9");
run(" Analyze Particles...", " size = 25-Infinity
circularity = 0.00-1.00 show =[Count Masks] display");
selectImage(2);
close();
run("Tile");
}
```

1．画像の拡大・縮小

読み込んだ画像の拡大・縮小を行うには，原画像を読み込み表示してから Image → Scale(ctrl＋E)を選択し Scale ダイアログを表示させ，X Scale，Y Scale に拡大率を入力するか，Width，Height に希望のマトリックスサイズを入力して OK を押せばよい．補間は，Interpolate ボタン on＝線形補間，off＝最近傍法である．また，バイキュービック補間法による画像の拡大・縮小処理のプラグインプログラムを付録 2 に記す．

付録 **2**

このプラグインプログラムを実行できるようにするには，

・画像を読み込む．
・テキスト形式で付録のマクロプログラムを入力し，bicubic_interpolation.java で フ ォ ル ダ C:￥Program Files￥ImageJ￥plugins に保存する．

・Plugins → Compile and Run を選び，bicubic_interpolation.java を選択し実行する．
（ImageJ を再起動した後からは，Plugins メニューリストの下部に bicubic_ interpolation が現れているので，画像を読み込んだ後実行すればよい．）

```
// bicubic_interpolation.java
import ij. ＊ ;
import ij.process. ＊ ;
import ij.gui. ＊ ;
import java.awt. ＊ ;
import ij.plugin.filter. ＊ ;
import java.lang.Math. ＊ ;

public class bicubic_interpolation implements PlugInFilter
{
        ImagePlus imp;

        public int setup(String arg, ImagePlus imp){
                this.imp = imp;
                return DOES_16;
        }

        public void run(ImageProcessor ip){
                int mszx, mszy, outmszx, outmszy,intX,
                intY;
                int i, j, ii, jj, u,v;
                double bicubic, buffer, floatX, floatY;
                double ratio = 0.4;

                short [] img =(short [])ip.getPixels
                ();

      // 拡大率入力
                GenericDialog gd = new GenericDialog
                ("reduce factor", IJ.getInstance());
                gd.addNumericField("enlarge factor",
                ratio, 1 );
                gd.showDialog();
                if( gd.wasCanceled()){
                        return;
                }
                ratio = (double) gd. getNextNumber
                ();


      // 拡大・縮小画像の領域確保
                mszx = ip.getWidth();
                mszy = ip.getHeight();
                outmszx = (int)((double)mszx ＊
                ratio);
                outmszy = (int)((double)mszy ＊
                ratio);
                ImageProcessor FilteredIp = new
                ShortProcessor(outmszx,outmszy);
                ImagePlus FilteredImp = new
                ImagePlus( "Enlarged Image ",
                FilteredIp );

// 拡大・縮小処理
        for(j = 0; j < outmszy; j ++){
                floatY = (double)j / ratio;
                intY = (int)floatY;
                for(i = 0; i < outmszx; i ++){
                        floatX = (double)i / ratio;
                        intX = (int)floatX;

  if(intX -1 >= 0 && intX + 2 < mszx&&intY-1 >
  = 0 && intY + 2 < mszy){
                        bicubic = 0.0;
                        for (jj = 0; jj <= 3; jj ++){
                                v = intY - 1 + jj;
                                buffer = 0.0;
                                for (ii = 0; ii <= 3;ii +
                                +){
                                u = intX - 1 + ii;
                                buffer = buffer + img
                                [v ＊ mszx + u] ＊
                                cubic(floatX- (double)
                                u);
                        }
                                bicubic = bicubic +
                                buffer ＊ cubic(floatY-
                                (double)v);
                        }
                        FilteredIp. putPixelValue
                        (i, j, (short)bicubic );

  }
                                else FilteredIp.
                                putPixelValue(i, j, 0);
                }
        }
                FilteredIp.resetMinAndMax();
                FilteredImp.show();
  }

// バイキュービック補間
    private double cubic(double x) {
            double z;
                if (x < 0.0)x = -x;
                z = 0.0;
                        if(x < 1.0)z = x ＊ x ＊
                        x-2.0 ＊ x ＊ x + 1.0;
                        else if (x < 2. 0)z =
                        -x ＊ x ＊ x + 5.0 ＊ x ＊ x-
```

```
                    8.0 * x + 4.0;
                return z;
        }

}
```

2. 階調処理

LUT を用いた階調処理

付録 1 のマクロ(2)Linear LUT，(3)Invert LUT，(4)Sigmoid LUT は，それぞれ，線形コントラスト強調，白黒反転，非線形コントラスト強調をするためのプログラムである．また，マクロプログラムメニューの(5)Reset LUT は，LUT を元に戻すためのマクロである．これらのマクロを用いて階調処理をすることができる．

ヒストグラム平坦化

画像を読み込んだ後 Process → Enhance Contrast を選び，Equalize Histogram のボックスをチェックして Alt キーを押しながら OK を押すことで，ヒストグラム平坦化が行える．

3. 空間フィルタ処理

画像の平滑化

空間フィルタにより平滑化をする手順を以下に記す．

・画像ファイルを読み込む．
・メニューから Process → Filters → Convolve を選び，フィルタの係数を入力する．
・ImageJ は，フィルタ係数を整数で入力する仕様になっているので，たとえば，平均値フィルタを用いる場合は，係数を 25 倍しすべての係数を 1 として数値入力し，OK ボタンを押すと処理できる．なお，係数は半角数値とし，数値間は“半角スペース”で区切って各行の終わりで改行する．ただし，最終行だけは改行しなくてよい．Normalize Kernel のボックスをチェックすれば，入力した係数の総和で割る正規化が行われる．

メディアンフィルタ処理

メニューから，Process → Filters → Median を選び，Radius（フィルタの半径）を入力して OK を押す．ImageJ のメディアンフィルタは，直径 2 * Radius+1 で円形である．

画像の鮮鋭化

付録 1 の(6)にラプラシアンを用いた鮮鋭化処理のマクロプログラムを掲載した．

4. エッジ検出

ImageJ では，画像を読み込んだ後，Process → Find Edges を選ぶことによって，Sobel フィルタによるエッジ強調ができる．

5. 空間周波数フィルタ処理

空間周波数フィルタ処理

空間周波数フィルタ処理のマクロプログラムを付録 1 の(7)～(9)に付記した．

6. 画像の 2 値化

判別分析法

判別分析法でしきい値を求め 2 値化をするプラグインプログラムを付録 3 に記した．

付録 3

このプラグインプログラムを実行できるようにするには，

・画像を読み込む．
・テキスト形式で付録のマクロプログラムを入力し，Thresholding_.java でフォルダ C:￥Program Files ￥ImageJ￥plugins に保存する．
・Plugins → Compile and Run を選び，Thresholding_.java を選択し実行する．

（ImageJ を再起動したら，Plugins メニューリストの下部に Thresholding が現れているので，画像を読み込んだ後実行すればよい．）

```
// Thresholding_.java
import ij. * ;
import ij.process. * ;
import ij.gui. * ;
import java.awt. * ;
import ij.plugin.filter. * ;
import java.lang.Math. * ;
import ij.text.TextWindow;

public class Thresholding_ implements PlugInFilter{
        ImagePlus imp;

        public int setup(String arg, ImagePlus imp) {
                this.imp = imp;
                return DOES_16;
        }

        public void run (ImageProcessor ip) {
                int width, height;
                int i, j, k;
                double avet, vart, w1, w2, u1, u2, varb,
                varw, thr;
              int maxpix = 1024;
              int threshold;

                short [] img = (short [])ip.getPixels
                ();
      // 画像の階調数入力----------------------------
                GenericDialog gd = new GenericDialog
                ("parameter", IJ.getInstance());
                gd. addNumericField(" 階調数 ",
                maxpix, 0 );
                gd.showDialog();
                if( gd.wasCanceled()){
```

```
                return;
            }
            maxpix = (int)gd. getNextNumber
            ();

// 配列宣言----------------------------------------
            double histf[] = new double[maxpix];
            double area1[] = new double[maxpix];
            double ave1[] = new double[maxpix];
            double ave2[] = new double[maxpix];
            double var1[] = new double[maxpix];
            double var2[] = new double[maxpix];
            double eta[] = new double[maxpix];

// 配列ゼロセット----------------------------------
            for( i = 0; i < maxpix; i ++){
              histf[i]= 0.0;
              area1[i]= 0.0;
              ave1[i]= 0.0;
              ave2[i]= 0.0;
              var1[i]= 0.0;
              var2[i]= 0.0;
              eta[i]= 0.0;
            }

// 画像のマトリックスサイズ取得----------------
            width = ip.getWidth();
            height = ip.getHeight();


// 画像の正規化ヒストグラム作成および平均,
分散の計算----------------------------------------
        for (i = 0; i < width * height; i ++){
            histf[img[i]]= histf[img[i]]+ 1.0;
        }

        for (i = 0; i < maxpix; i ++){
            histf[i]= histf[i]/(double)(width *
            height);
        }


        avet = 0.0;
        for (i = 0; i < maxpix; i ++){
            avet = avet +(double)i * histf[i];
        }

        vart = 0.0;
        for (i = 0; i < maxpix; i ++){
            vart = vart +((double)i-avet) *
            ((double)i-avet) * histf[i];
        }
```

```
// 判別分析法によるしきい値決定----------------
        for (k = 0; k < maxpix; k ++){

            for (i = 0; i < k; i ++){
                area1[k]= area1[k]+ histf[i];
            }
            w1 = area1[k];  // クラス 1 の面積
            w2 = 1.0-w1;     // クラス 2 の面積

            for (i = 0; i < k; i ++){
                ave1 [k] = ave1 [k] +(double)i *
                histf[i];
            }
            u1 = ave1[k]/w1; // クラス 1 の平均
            u2 =(avet-ave1[k])/w2;// クラス 2 の
            平均

            for (i = 0; i < k; i ++){
                var1 [k] = var1 [k] +((double)i-
                u1) *((double)i-u1) * histf [i]/
                w1;
            }
            for (i = k; i < maxpix; i ++){
                var2 [k] = var2 [k] +((double)i-
                u2) *((double)i-u2) * histf [i]/
                w2;
            }
            varw = w1 * var1[k]+ w2 * var2[k]; //
            クラス内分散
            varb = w1 * w2 *(u1-u2) *(u1-u2);
            // クラス間分散

            eta[k]= varb/varw;              // 判別指標

            }

            thr = 0.0;
            threshold = 0;
            for (k = 0; k < maxpix; k ++){
              if(eta[k] > thr) {
                    thr = eta[k];
                    threshold = k;
              }
            }

// 2 値画像作成の作成----------------------------
              ImageProcessor FilteredIp = new
              ShortProcessor(width, height);
              ImagePlus FilteredImp = new
              ImagePlus("Binary Image_"+
              threshold, FilteredIp);

            for (j = 0; j < height; j ++){
```

```
                for (i = 0; i < width; i ++){
                  if(img[j * width + i] > threshold)
                  FilteredIp.putPixel(i, j, 1);
                  else FilteredIp.putPixel(i, j, 0);
                }
              }
//-------------------------------------------------------------
              FilteredIp.resetMinAndMax();
                FilteredImp.show();
        }
}
```

2 値化処理

ImageJ で 2 値化をするには，画像を表示して Process → Binary → Make Binary を選べばよい．しきい値は，ある画素値で画像の濃度ヒストグラムを 2 分割しながら，2 つの領域に属す画素の平均画素値の平均を求め，2 領域に分割した画素値＞(左側のヒストグラムの平均値＋右側のヒストグラムの平均値)/2 を満たすときの画素値が用いられる．手動でしきい値を決めて 2 値化をする場合は，Image → Adjust → Threshold を選択し，スライドバーを調整し画像を見ながらしきい値を決定して Apply を選べばよい．Set Background Pixels to NaN というダイアログは，処理の演算結果が不正(Not a number)のときの処理を指定するものであるが，このボックスは外しておく．こうして得られる 2 値画像は，しきい値より小さい画素は 0 で，大きい画素は 255 となる．

7. ラベリング

ラベリングをするには，2 値画像を作成した後，Analyze → Analyze Particles を選んでダイアログを開き，Show の選択を Count Masks にすればよい．陰影ごとにラベル番号が付された画像が表示される．

8. 特徴量分析

濃度と形状に関する特徴量を求めるマクロプログラムは，付録 1(10)に記載した．

特徴量を用いた分類方法

特徴量を用いた分類は，無料で入手できる「WEKA」や「R」といったソフトウェアを用いて実行することができる．WEKA は www.weka-jp.info から，R は www.okada.jp.org/RWiki からそれぞれダウンロードすることができる．

9. モルフォロジカルフィルタ

膨張と収縮

2 値画像を表示させた状態で，Process → Binary → Dilate で膨張処理，Process → Binary → Erode で収縮処理ができる．

濃淡画像の場合でも，Process → Filters のメニュー中の Maximum，Minimum を選択実行することにより，それぞれ，膨張，収縮処理をすることができる．

オープニングとクロージング

2 値画像を表示させた状態で，Process → Binary → Open でオープニング処理を，Process → Binary → Close でクロージング処理をすることができる．

濃淡画像の場合，Process → Filters のメニュー中にある，Maximum(最大値フィルタ)と Minimum(最小値フィルタ)を組み合わせて使い処理することができる．

10. 画像間演算

ImageJ で 2 つの画像間演算を行うには，Process → Image Calculator を選び，処理したい 2 枚の画像と演算の種類を選べばよい．

(石田隆行)