Supplementary Text

```
import ij.*;
import ij.gui.*;
import ij.process.*;
import ij.measure.ResultsTable;
import ij.measure.Measurements;
import ij.plugin.*;
import ij.plugin.filter.*;
import ij.plugin.filter.ParticleAnalyzer;
import ij.plugin.frame.ThresholdAdjuster;
import java.awt.*;
import java.awt.image.*;
import java.lang.Math.*;

public class Enumerate_Cells implements PlugInFilter {
  private boolean LOG=true;
  ImagePlus imp;
  ImageStack stack=null;
  int histMax=5000; int histMin=0; int nBins=100;
  double maxThres=103.0f;

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

  public void run(ImageProcessor ip) {
    int whichObject = 2;
    GenericDialog gd = new GenericDialog("Enumerate cells and Count debriss");
    String radioList[] = {"Cells","Debris","Both"};
    gd.addRadioButtonGroup("Which objects enumerate ?",
                           radioList,1,3, radioList[whichObject]);
    gd.addNumericField("maxThres for debris: ",maxThres,0);
    gd.addCheckbox("Show logY", LOG);
    gd.showDialog();
    if (gd.wasCanceled()) return;

    String radio = (String)gd.getNextRadioButton();
    maxThres  = (double)gd.getNextNumber();
    LOG = (boolean)gd.getNextBoolean();

    whichObject=0;
    while(!radio.equals(radioList[whichObject])){ whichObject++; }
    int WH=100;

    ByteProcessor bp = ip.convertToByteProcessor();
    ColorModel cm = LookUpTable.createGrayscaleColorModel(false);
    stack = new ImageStack(WH,WH,cm);

    Plot pt;
    if (LOG)
      pt = new Plot("Histogram of area", "Area (pixels)","# of cells (LOG)");
    else
      pt = new Plot("Histogram of area", "Area (pixels)","# of cells");
    pt.setLogScaleY();
    double xMin=0; double xMax=histMax;
    if (LOG) pt.setLimits(xMin, xMax, 0, 10);

    ResultsTable rt1=null;
    ResultsTable rt2=null;
    // For Cells
    if (whichObject==0||whichObject==2){
      rt1 = extractCells(bp);
      if (rt1.getCounter()==0){IJ.showMessage("rt1 is 0");return;}
```

```
     IJ.log("rt1: "+rt1.getCounter());
     float[] x1 = new float[nBins];
     float[] y1 = new float[nBins];
     makeAreaHistogram(rt1,x1,y1,LOG);
     pt.setColor(Color.blue); pt.setLineWidth(2);
     pt.addPoints(x1, y1, Plot.LINE);
     if (whichObject==0) IJ.showMessage("Cells: "+rt1.getCounter());
   }
   // For Debris
   if (whichObject==1||whichObject==2){
     rt2 = extractDebris(bp);
     if (rt2.getCounter()==0){IJ.showMessage("rt2 is 0");return;}
       float[] x2 = new float[nBins];
          float[] y2 = new float[nBins];
          makeAreaHistogram(rt2,x2,y2,LOG);
          pt.setColor(Color.red); pt.setLineWidth(3);
          pt.addPoints(x2, y2, Plot.LINE);
          if (whichObject==1) IJ.showMessage("Debris: "+rt2.getCounter());
   }
   if (whichObject==2)
          IJ.showMessage("Cells:"+rt1.getCounter()+"¥n"+
                              "Debris: "+rt2.getCounter());
   pt.show();

   // For Cells
   if (whichObject==0||whichObject==2){
     if (stack.getSize()!=0){
          int nSlices = stack.getSize();
          int columns = (int)Math.sqrt(nSlices);
          int rows = columns;
          int n = nSlices - columns*rows;
          if (n>0) columns += (int)Math.ceil((double)n/rows);
            double scale = 1.0;
            ImagePlus imps = new ImagePlus("stack",stack);//imps.show();
            MontageMaker mm = new MontageMaker();
            mm.makeMontage(imps,columns,rows,scale, 1,nSlices,1,3,false);
          }
   }
 }

 void makeAreaHistogram(ResultsTable rt, float[] x, float[] y, boolean flag){
   float[] fArea = rt.getColumn(0);
   short[] sArea = new short[fArea.length];
   for(int i=0;i<sArea.length;i++) sArea[i] = (short)fArea[i];
     ColorModel cm = LookUpTable.createGrayscaleColorModel(false);
     ShortProcessor sp = new ShortProcessor(1,sArea.length,sArea,cm);
     ImagePlus imp1 = new ImagePlus("area",sp);
     int mOptions = 0;
     ImageStatistics stats=
         imp1.getStatistics(mOptions, nBins, histMin, histMax);
     long[] his = stats.getHistogram();
     for(int i=0;i<x.length;i++){
          x[i]=(float)i*(float)(histMax-histMin)/(float)nBins;
          if (flag) y[i]=(float)Math.log10((double)his[i]);
          else y[i]=(float)his[i];
     }
 }

 ResultsTable extractCells(ImageProcessor bp){
   ImagePlus imp2 = new ImagePlus("Cells",bp);
   ResultsTable rt = new ResultsTable();
   int options = ParticleAnalyzer.SHOW_RESULTS|
                    ParticleAnalyzer.EXCLUDE_EDGE_PARTICLES|
                    ParticleAnalyzer.CLEAR_WORKSHEET;
                    //ParticleAnalyzer.INCLUDE_HOLES;
```

```
    int measurements = Measurements.RECT|Measurements.AREA;

    bp.setAutoThreshold(AutoThresholder.Method.Triangle,true);//true=Dark Background
    int minSize=100; int maxSize=5000;
    ParticleAnalyzer pa =
      new ParticleAnalyzer(options, measurements, rt, minSize, maxSize,0.2,1.0);
    pa.setHideOutputImage(true);
    pa.analyze(imp2);
    int WH=100;
    int width=bp.getWidth(); int height=bp.getHeight();
    for(int i=0;i<rt.getCounter();i++){
      int bx = (int)rt.getValue("BX",i);
      int by = (int)rt.getValue("BY",i);
      int w = (int)rt.getValue("Width",i);
      int h = (int)rt.getValue("Height",i);
      if ( (w<=WH) && (h<=WH) ){
        if ( (bx+w/2+WH/2<=width) && (by+h/2+WH/2<=height) ){
            int cbx = (bx+w/2)-WH/2;
            int cby = (by+h/2)-WH/2;
            if ( cbx>=0 && cby>=0 ){
              bp.setRoi(cbx, cby, WH, WH);
              ImageProcessor crop = bp.crop();
              stack.addSlice(crop);
            }
          }
      }
    }
    imp2.close();
    return rt;
  }

  ResultsTable extractDebris(ByteProcessor bp){
    ImagePlus imp2 = new ImagePlus("Debris",bp);
    double minThres=0.0f;
    bp.setThreshold(minThres, maxThres, ImageProcessor.RED_LUT);
    int options = ParticleAnalyzer.SHOW_RESULTS|
                  ParticleAnalyzer.EXCLUDE_EDGE_PARTICLES|
                  ParticleAnalyzer.CLEAR_WORKSHEET|
                  ParticleAnalyzer.SHOW_MASKS|
                  ParticleAnalyzer.INCLUDE_HOLES;
    int measurements = Measurements.RECT|Measurements.AREA;
    ResultsTable rt = new ResultsTable();
    int minSize=5; int maxSize=100;
    ParticleAnalyzer pa = new ParticleAnalyzer(options,
                          measurements, rt,minSize,maxSize,0.2,1.0);
    pa.setHideOutputImage(false);
    pa.analyze(imp2);
    imp2.close();
    return rt;
  }
}
```