Automated Measurement of the Areas of Bivalve Shells

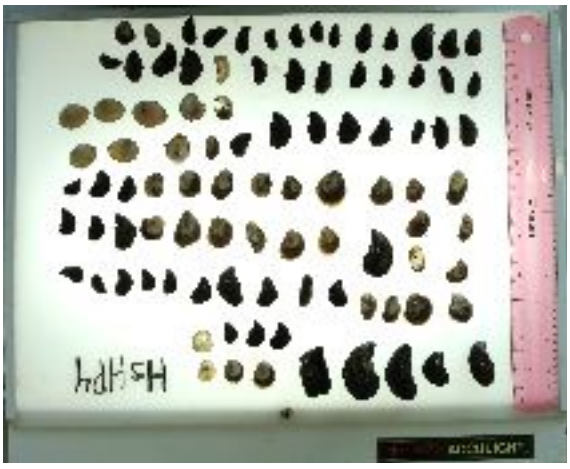
(For questions or assistance, contactTeresa Vaillancourt svmehitabel@gmail.com)

ImageJ from the NIH <https://fiji.sc> can be used to automate measurements of shell areas. Measurements that can take hours when done manually take only seconds if performed automatically.

Optimize original images

Properly photographed original images are required for optimal automation. Objects should be photographed with a balance of back-lighting and front-lighting. Proper back-lighting has a uniform intensity throughout the photographed field; an adjustable intensity source will help optimize the balance of background and object lighting. Front-lighting should also be uniform; it should not produce glare or shadows. Glare from overhead lights and windows should be avoided.

The following images illustrate a typical image and an optimal one. The image on the left was backlit using a simple light box; it was front-lit with ambient light from windows and overhead fluorescents. The background intensity is variable from edge to edge, and it is not much different from the intensity of the lighter objects in the image. The image on the right illustrates a uniform background and a good balance between background and object illumination; a digital tablet computer was used to produce the background and the flash from a smartphone camera lit the objects without producing much glare -- the image was produced in a room without overhead lighting.



Using macros to automate image analysis

A macro consists of a series of commands that direct the ImageJ program to perform specific functions. Four macros have been created to automate shell area measurements.

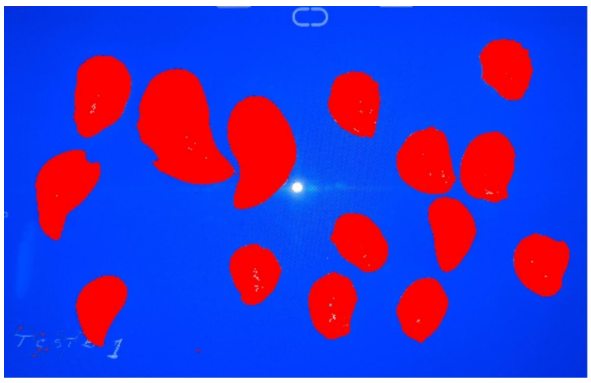
SERC_Shell_Area.ijm	Measures areas of shells in a single image.
SERC_Shell_Area_Batch.ijm	Measures areas of shells in all images in a directory.
SERC_Shell_Threshold.ijm	*Digitally Isolates shell objects from the background.
SERC_Shell_Threshold_Macro_Instructions.ijm	Instructions for creating an optimal threshold macro.

*The Shell_Area and Shell_Area_Batch macros use the Shell_Threshold macro to isolate objects of interest from the image background.

Typical workflow to prepare for automated shell area measurements.

1. Create optimal images.
2. Clean images: use ImageJ brush/pen/eraser tools to separate shell boundaries from mussel byssal threads, attached barnacles, adjacent shells, etc. -- the tool color should match the background.
3. Open a typical image and obtain a length calibration value, i.e., the number of image pixels per millimeter -- do not use the calibration function of ImageJ. Also, at this time you should estimate the area, in pixels-squared, of the smallest object of interest.
4. Place the four SERC_Shell macros in the ...\Fiji.app\macros directory.
5. Open a text window for running the macros: File > New > Text Window {ctrl-shift-N}.
    a. Open the SERC_Shell_Threshold_Macro_Instructions.ijm macro, and follow the directions to update the SERC_Shell_Threshold.ijm macro for the specific conditions used to photograph the shells for the current study.

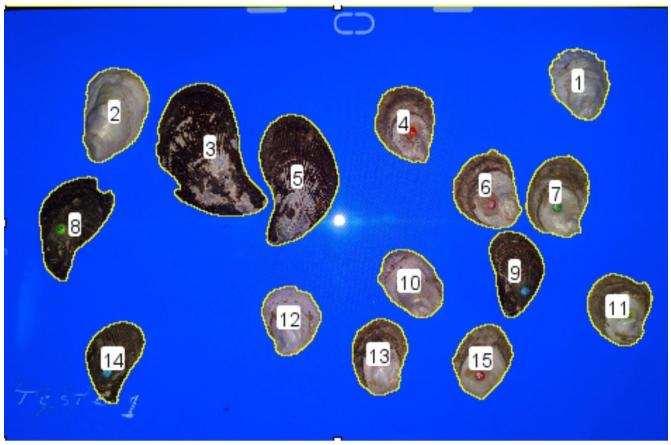
An optimal threshold setting will result in a thresholded image like the one here. The objects of interest will be entirely covered by a red mask. Unmasked regions in the center of a shell image are fine, as long as the entire periphery of the shell is masked.



Measure areas automatically.

- Open a text window for running the macros: File > New > Text Window {ctrl-shift-N}.
    - Open SERC_Shell_Area.ijm if analyzing one image at a time, or SERC_Shell_Area_Batch.ijm if analyzing an entire directory of images.
        - If analyzing one image at a time, open the image to be analyzed, then RUN the SERC_Shell_Area.ijm macro. NOTE: this macro does not automatically save the overlaid image or the results table.
        - If analyzing an entire directory of images, RUN the SERC_Shell_Area_Batch.ijm macro. NOTE: all overlaid image files and the cumulative area file are saved automatically.

The following figure illustrates the results of running the SERC_Shell_Area.ijm macro on a single image. The original image is overlaid with yellow outlines that indicate the object areas that were isolated using the SERC_Shell_Threshold.ijm macro that had been optimized for this type of image. Each object is also overlaid with a number that corresponds with one in the "OverlayNumber" column of the results table. The "Area" column is in raw units of pixels-squared.



Results_01_Feb_2020_15_03

File Edit Font

Label	Area	Area(mm^2)	OverlayNumber
blueOriginal.jpg	99949	9.995	1
blueOriginal.jpg	150727	15.073	2
blueOriginal.jpg	306399	30.640	3
blueOriginal.jpg	112540	11.254	4
blueOriginal.jpg	239192	23.919	5
blueOriginal.jpg	125037	12.504	6
blueOriginal.jpg	127454	12.745	7
blueOriginal.jpg	151277	15.128	8
blueOriginal.jpg	106552	10.655	9
blueOriginal.jpg	99971	9.997	10
blueOriginal.jpg	111127	11.113	11
blueOriginal.jpg	95173	9.517	12
blueOriginal.jpg	104232	10.423	13
blueOriginal.jpg	100575	10.057	14
blueOriginal.jpg	99189	9.919	15

The SERC_Shell_Area_Batch.ijm macro generates an overlaid image for each image in the processed directory and it generates and saves a cumulative results table in which the "Label" column corresponds to the file name of the original image.

Images with overlays are best viewed in ImageJ. Results tables are saved in a ".csv" format that can be opened directly by a spreadsheet program such as Microsoft Excel.

SERC_Shell_Area.ijm macro code

```
//Shell area analysis macro for a single image of one or more cells.  It is
//assumed that the image has not been calibrated by using "Set Scale" function
//of ImageJ.
#@ Float (label = "Pixels per millimeter (enter 0 if unknown)", value = 0) pixels_per_mm
#@ Float (label = "Lower limit of shell size in units of pixels-squared", value = 0) shell_area_lower_limit
#@ Float (label = "Upper limit of shell size in units of pixels-squared", value = 1000000) shell_area_upper_limit

run("Set Measurements...", "area display add redirect=None decimal=3");
//copy the original image
run("Select All");
run("Copy");
runMacro("SERC_Shell_Threshold.ijm");

//analyze particles, resulting in numbered outline overlays
shell_size_range = "size="+ toString(shell_area_lower_limit, 0)+toString(-shell_area_upper_limit,0);
optionString = shell_size_range + " show=[Overlay Masks] display exclude include";
run("Analyze Particles...", optionString);
//make the overlay image RGB, set label font, paste back the original image, then apply the overlay
run("RGB Color");
run("Labels...", "color=black font=14 show draw");
run("Paste");
run("Overlay Options...", "stroke=yellow width=5 fill=none set apply show");
//Write shell overlay number to the results table for the active image,
rowStart = Table.size - Overlay.size;
rowEnd = Table.size;
for (i=rowStart; i<rowEnd; i++) {
		area = Table.get("Area", i);
 if (pixels_per_mm != 0) {
  calibratedArea= (1/(pixels_per_mm*pixels_per_mm))*area;
  Table.set("Area(mm^2)", i, calibratedArea);
 } else {
  Table.set("Area(px^2)", i, area)
 }

Table.set("OverlayNumber", i,i-rowStart+1);
}
Table.showRowNumbers(0);
//Results Table: rename using date+time stamp
MonthNames = newArray("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec");
DayNames = newArray("Sun", "Mon","Tue","Wed","Thu","Fri","Sat");
getDateAndTime(year, month, dayOfWeek, dayOfMonth, hour, minute, second, msec);
TimeString = "";
if (dayOfMonth<10) {TimeString = TimeString+"0";}
 TimeString = TimeString+dayOfMonth+"_"+MonthNames[month]+"_"+year+"_";
if (hour<10) {TimeString = TimeString+"0";}
 TimeString = TimeString+hour+"_";
if (minute<10) {TimeString = TimeString+"0";}
 TimeString = TimeString+minute;
tableName = "Results_"+TimeString;
Table.rename("Results", tableName);
```

SERC_Shell_Area_Batch.ijm macro code

```
//SERC_Shell_Area_Batch.ijm macro code

// This performs shell area processing for all images in a specified directory
// Processed images (original image + shell outline overlays) are
// saved with a tif format in a specified directory.
// If a non-zero calibration variable is specified, a calibrated area column
// will be included in the results table.
//Note: the #@ Script parameters are a feature of ImageJ2;
//they will not work in plain ImageJ1.
//The Fiji distribution of ImageJ is built on ImageJ2, so they also work in Fiji.

#@ File (label = "Input directory", style = "directory") input
#@ File (label = "Output directory", style = "directory") output
#@ Float (label = "Pixels per millimeter (enter 0 if unknown)", value = 0) pixels_per_mm
#@ Float (label = "Lower limit of shell size in units of pixels-squared", value = 10000) shell_area_lower_limit
#@ Float (label = "Upper limit of shell size in units of pixels-squared", value = 1000000) shell_area_upper_limit

//shell_areas function: this function runs the SERC_Shell_Threshold.ijm macro
//for a specified image, the runs the Analyze Particles routine.
function shell_areas(input, output, filename, pixels_per_mm) {
open(input + File.separator + filename);
run("Set Measurements...", "area display add redirect=None decimal=3");
//copy the original image
run("Select All");
run("Copy");

runMacro("SERC_Shell_Threshold.ijm");

//analyze particles, resulting in numbered outline overlays
shell_size_range = "size="+ toString(shell_area_lower_limit, 0)+toString(-shell_area_upper_limit,0);
optionString = shell_size_range + " show=[Overlay Masks] display exclude include";
run("Analyze Particles...", optionString);
//make the overlay image RGB, set label font, paste back the original image, then apply the overlay
run("RGB Color");
run("Labels...", "color=black font=14 show draw");
run("Paste");
run("Overlay Options...", "stroke=yellow width=5 fill=none set apply show");
//Write shell overlay number to the results table for the active image,
rowStart = Table.size - Overlay.size;
rowEnd = Table.size;
for (i=rowStart; i<rowEnd; i++) {
        area = Table.get("Area", i);
 if (pixels_per_mm != 0) {
  calibratedArea= (1/(pixels_per_mm*pixels_per_mm))*area;
  Table.set("Area(mm^2)", i, calibratedArea);
 } else {
  Table.set("Area(px^2)", i, area)
 }

Table.set("OverlayNumber", i,i-rowStart+1);
}
saveAs("Tiff", output + File.separator+filename);
close();
}
```

```
//Batch process: get file names from input directory, then call
//shell_areas function for each file until finished.
setBatchMode(true);
list = getFileList(input);
for (i = 0; i < list.length; i++)
        shell_areas(input, output, list[i], pixels_per_mm);
setBatchMode(false);
//.............................
Table.showRowNumbers(0)
//Results Table: rename using date+time stamp and save in output directory
    MonthNames = newArray("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec");    DayNames =
newArray("Sun", "Mon","Tue","Wed","Thu","Fri","Sat");
    getDateAndTime(year, month, dayOfWeek, dayOfMonth, hour, minute, second, msec);
            TimeString = "";
    if (dayOfMonth<10) {TimeString = TimeString+"0";}
    TimeString = TimeString+dayOfMonth+"_"+MonthNames[month]+"_"+year+"_";
    if (hour<10) {TimeString = TimeString+"0";}
    TimeString = TimeString+hour+"_";
    if (minute<10) {TimeString = TimeString+"0";}
    TimeString = TimeString+minute;
tableName = "Results_"+TimeString;
Table.rename("Results", tableName);
fileName = tableName + ".csv";
Table.save(output + File.separator+fileName);
```

SERC_Shell_Threshold.ijm macro code -- this will change, depending on specific imaging conditions

```
// Color Thresholder 2.0.0-rc-69/1.52s
// Autogenerated macro, single images only!
min=newArray(3);
max=newArray(3);
filter=newArray(3);
a=getTitle();
run("HSB Stack");
run("Convert Stack to Images");
selectWindow("Hue");
rename("0");
selectWindow("Saturation");
rename("1");
selectWindow("Brightness");
rename("2");
min[0]=0;
max[0]=255;
filter[0]="pass";
min[1]=0;
max[1]=255;
filter[1]="pass";
min[2]=0;
max[2]=168;
filter[2]="pass";
for (i=0;i<3;i++){
  selectWindow(""+i);
```

```
  setThreshold(min[i], max[i]);
  run("Convert to Mask");
  if (filter[i]=="stop")  run("Invert");
}
imageCalculator("AND create", "0","1");
imageCalculator("AND create", "Result of 0","2");
for (i=0;i<3;i++){
  selectWindow(""+i);
  close();
}
selectWindow("Result of 0");
close();
selectWindow("Result of Result of 0");
rename(a);
// Color Thresholding----------
```

SERC_Shell_~~Area_~~Threshold_Macro_Instructions.ijm macro code

```
  print("\\Clear");
  print("Follow these steps to create an optimized SERC_Shell_Threshold macro.\n");
  print("1. Open a representative image.\n");
  print("2. Menu: Image>Adjust>Color Threshold>.\n    Typically, the shells will have been photographed on a bright background. If so, uncheck the 'Dark background' box. \n    Adjust the 'Threshold Color' parameters so that the shells are mostly covered in Red, or whichever 'Threshold color' is selected.\n    A very good thresholding may be achieved simply by adjusting the second slider beneath the 'Brightness' histogram. \n    Excellent shell differentiation is possible as long as the shell margins  are completely thresholded.\n");
  print("3.  Menu: Plugins>Macros>Record. \n     Go to the 'Threshold Color' dialog window and press the 'Macro' button in the .\n Go to the 'Macro Recorder' window and press the 'Create' button.\n     A macro editor window opens with the 'Color Threshold' macro instructions. Go to the editor's File menu and\n     Save as...SERC_Shell_Threshold.ijm in the the FIJI_IMAGEJ>fiji-win64>Fiji.app>plugins>Macros>SERC directory.\n");
  print("4. Close image, Color Threshold, Macro Recorder, Editor and Log windows.");
```