An automated quantitative image analysis pipeline of *in vivo* oxidative stress and macrophage kinetics

Authors. Andre D. Paredes[1], David Benavidez[2], Jun Cheng[1], Steve Mangos[3], Michael Donoghue[4], Amelia Bartholomew[1,2]
Authors Affiliations. [1]Richard and Loan Hill Department of Bioengineering, University of Illinois at Chicago, [2]Department of Surgery, University of Illinois, [3]Department of Internal Medicine, Rush University, [4]Donoghue Chiropractic, Lincolnshire, Illinois
Full Postal and email of corresponding author(s). Andre Paredes; apared3@uic.edu
Short running title (40 characters). Zebrafish cell kinetics and ROS image analysis
Abbreviations used: (3D+t), three-dimensional time-lapse imaging; CTF, Corrected Total Fluorescence; DICE, Dice similarity coefficient; DHE, Dihydroethidium; dpf, days post-fertilization; FOV, Field of View; IntDen, Integrated Density; MPI, minutes post injury; ROI, Region of Interest; ROS, reactive oxygen species; µm, micrometer

Table of Contents

Table S1. User-defined parameters and their corresponding functionalities in *Zirmi* to execute an image analysis

User Input	Default	Module	Functionality in Zirmi
Minutes Post Injury (MPI)	30 minutes	1	initial image time stamp in reference to injury; used to standardize time points across different imaging batches
Sampling Frequency	1 minute/frame	1	the rate at which images are taken; used to standardize time points across different imaging batches
Lateral Resolution	1.64 µm / pixel	1	used to compute quantitative measures
Z-stack resolution	10 µm	1	used to compute quantitative measures
Bits per pixel (BPP)	16	1	Bit depth; used to compute quantitative measures
Parameter 1	Otsu value	2	used to threshold pixel intensities for image segmentation
Parameter 2	NA	2	outline of wound perimeter
Parameter 3	65 µm	2	radial distance extended from wound perimeter; used to reproduce wound region of interest
Parameter 4	NA	2	outline of background regions
Parameter 5	70%	3	percent used to select cell tracks based on distinguishable centroid positions relative to time domain
Parameter 6	0.9 µm		used to threshold cell movements as static
Parameter 7	NA	3	used to outline a zero position, S_o position; epicenter of wound
Parameter 8	NA	3	defines spatial domain, S1

Parameter 9	150 µm	3	defines the successive spatial domain radial extensions to define S2, S3, and S4
Parameter 10	NA	4	defines central directory used to save databases and visualizations
N/A, Not applicable		*Zirmi* is open-source permitting customization	

Table S2. Comparison of single wounded zebrafish macrophage velocity and static ratio by time frame interval total cell averages

Time frame [min]	Velocity [µm /min]	Respective P value	Static Ratio (0 to 1)	Respective P value
	Interval		***Interval***	
Zirmi 30-59, (Interval 1, Z-I)	3.03 ± 1.2	*vs. Z-II, **p=0.009* *vs. Z-III, ***p=0.0009*	0.43 ± 0.37	*vs. Z-II, n.s., p=0.14* *vs. Z-III, *p=0.01*
Zirmi 60-89 (Interval II, Z-II)	4.00 ± 2	*vs. Z-III, *p=0.03,*	0.35 ± 0.35	*vs. Z-III, n.s. p=0.1*
Zirmi 90-120 (Interval III, Z-III)	5.08 ± 1.5	*vs. PhagoSight *p=0.01*	0.16 ± 0.13	*vs. PhagoSight 30-120 ***p=0.0004*
	Cumulative		***Cumulative***	
PhagoSight 30-120	4.01 ± 1.77	*vs. Z-I, **p=0.004* *vs. Z-II, n.s, p=0.9*	0.40 ± 0.39	*vs. Z-I, n.s. p=0.96* *vs. Z-II,n.s. p=0.6*
Zirmi 30-120 (Z -I,II,III)	4.02 ± 1.33	*vs. PhagoSight 30-120, n.s. p=0.34*	0.32 ± 0.26	*vs. PhagoSigh t30-120, n.s. p=0.42*

Zirmi 30-59, data taken during the interval of 30-59 minutes, *Zirmi* 60-89 reflects the 60-89 minute interval, *Zirmi* 90-120 reflects 90-120 minute interval. *PhagoSight* 30-120 the 30-120 minute interval as determined by the *PhagoSight* software
n.s., Not Significant, Wilcoxon signed-rank test (*p<0.05,* *p<0.01, ***p<0.001, n.s. > 0.05);

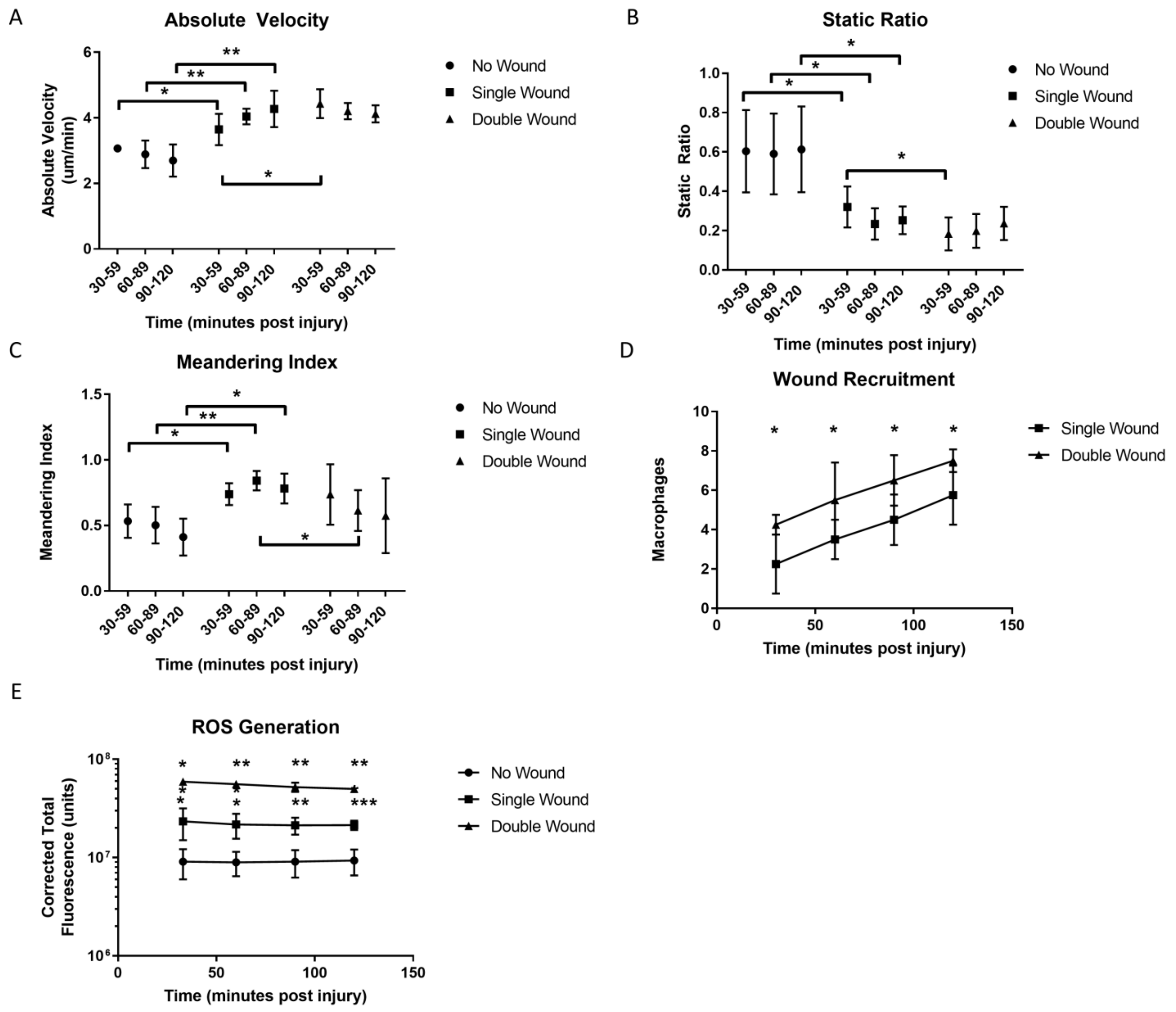

Figure S1. Wound severity comparative analysis. Macrophage absolute velocities (A), static ratios (B), and meandering (C) at three intervals: 30-59, 60-89, and 90-120 minutes post injury in unwounded, singly wounded, and doubly wounded fins. Macrophage cumulative wound recruitment in fish with single wounds (D) was compared to those with double wounds; ROS generation shown as Corrected Total Fluorescence (CTF) in unwounded fish (n=4) was compared to singly wounded (n=4) fish, and singly wounded CTF was compared to doubly wounded (n=3) fish at user defined time points (E). All values are shown as mean ± SD in GraphPad, Prism.

Script S1

```
%% Supplementary_ScriptS1
%   Zirmi Source Information can be found at:
%   <https://github.com/ADParedes/Zirmi>
%   Written By: Andre Daniel Paredes | email: andre.paredes@ymail.com
%   MATLAB Source Information can be found at:
%   <https://www.mathworks.com/help/images/image-analysis.html>
%   Description:  Single Image Processing Workflow commonly performed through available
MATLAB tool. Requires  manual modifications of thresholding and morphological function
values per image.
%% Define the following Directories appropriately
cd          (dirScript_Z_Supplementary)
addpath     (dirScript_E_Functions)
%% Section A: Read Image Data and Preprocess
%-Step A1: Read Image data into workspace from preselected directory
fluorFish                   = imread('fluorescent_fish.tif');
%-Step A2: Format fluorescen_fish image to a standardized bit format to streamline processing
bytes                       = 2^16-1;
I               = im2uint16(fluorFish);
%-Step A3: adjust Image to allow for more rigid image segmentations
background              = imopen(I,strel('disk',15));
I2                      = I - background;
I3                      = imadjust(I2);
figure                  ();imshow(I3);
%% Section B:  Threshold Fish Tail
% Step B1. Detect entire Fish Tail
[~, threshold]      = edge(I, 'sobel');
fudgeFactor         = .55;
BWs                 = edge(I,'sobel', threshold * fudgeFactor);
 figure(), imshow(BWs), title('binary gradient mask');
% Step B2. Morphological Operate Binary Image
se90                = strel('line', 3, 90);
se0                 = strel('line', 3, 0);
BWsdil              = imdilate(BWs, [se90 se0]);
BWdfill             = imfill(BWsdil,'holes');
BW2                 = bwareaopen(BWdfill, 1000);
seD                 = strel('diamond',1);
BWfinal             = imerode(BW2,seD);
BWfinal             = imerode(BWfinal,seD);
BWfinal             = bwareaopen(BWfinal,1000);
%-Step B3.  Show fish tail image segmentation
BWoutline           = bwperim(BWfinal);
thickBWperim        = bwmorph(BWoutline,'thicken',1);
Segout              = I;
Segout(thickBWperim)= bytes;
```

```
imgFinalPerim        = imoverlay(I,thickBWperim,[1 1 0]);
figure();
imshow(imgFinalPerim);
title('outlined original image');
%% Section C:  Extrapolate Binary Image Components for further Analysis
binaryImgComp        = bwconncomp(BWfinal,4);
```

Script S1 Description: Example of a single image processing pipeline commonly performed through available MATLAB tools; Requires custom user-defined modifications of thresholding and morphological function values per image.

Script S2

```
%% Supplementary_ScriptS2
%   Zirmi Source Information can be found at:
%   <https://github.com/ADParedes/Zirmi>
%   Written By: Andre Daniel Paredes | email: andre.paredes@ymail.com
%   Description: This uses “mindthegap” function incorporated into Zirmi as
%   a means to reproduce a wound region, acquire raw pixel intensity values, and eliminate
%   confounding fluorescence within the wound gap that is difficult and time consuming
%   to perform manually.
%%   Define the following directories appropriately:
   cd      (dirScript_Z_Supplementary)
   addpath (dirScript_E_Functions)
%% Section A:  Read Image Data and Preprocess
%-Step A1: Read Image data into workspace from preselected directory
fluorFish                   = imread('fluorescent_fish.tif');
%-Step A2: Format fluorescen_fish image to a standardized bit format to streamline processing
bytes                       = 2^16-1;
fluorFishImg                = im2uint16(fluorFish);
%-Step A3: adjust Image to allow for more rigid image segmentations
image_background            = imopen(fluorFishImg,strel('disk',15));
fluorFishImg_v2             = fluorFishImg - image_background;
fluorFishImg_v3             = imadjust(fluorFishImg_v2);
%% Section B:  Threshold Fish Tail
%-Step B1: Threshold Fish Fluoresence to ascertain binary image
[~, threshold_fluor]        = edge(fluorFishImg, 'sobel');
fudgeFactor                 = .55;
binaryFishImg               = edge(fluorFishImg,'sobel', threshold_fluor * fudgeFactor);
%-Step B2. Apply Morphological Operations to enhance image segmentation
structElement_90            = strel('line', 3, 90);
structElement_0             = strel('line', 3, 0);
binaryFishImg_dil           = imdilate(binaryFishImg, [structElement_90 structElement_0]);
binaryFishImg_fill          = imfill(binaryFishImg_dil,'holes');
binaryFishImg_open          = bwareaopen(binaryFishImg_fill, 1000);
structElement_D             = strel('diamond',1);
binaryFishImg_erode_v1      = imerode(binaryFishImg_open,structElement_D);
binaryFishImg_erode_v2      = imerode(binaryFishImg_erode_v1,structElement_D);
binaryFishImg_Final         = bwareaopen(binaryFishImg_erode_v2,1000);
%-Step B3.  Show fish tail image segmentation
fishImg_perim               = bwperim(binaryFishImg_Final);
fishImg_perim_v2            = bwmorph(fishImg_perim,'thicken',1);
Segout                      = fluorFishImg;
Segout(fishImg_perim_v2)    = bytes;
disp_fish_image_seg         = imoverlay(fluorFishImg,fishImg_perim_v2,[1 1 0]);
figure1                     = figure();
imshow(disp_fish_image_seg);
title ('flourescent fish tail image segmentation');
```

```
%% Section C:  Standardize image segmentation of "V-Shaped" injury
%-Step C1: Determine Image Size
[width_x,height_y]         = size(fluorFishImg);
%-Step C2: Trace Wound Gap
disp    ('Please trace a "V" along te wound perimeter')
warning ('Be sure to extend "V" line to open water')
warning ('DO NOT close the V or make any closed shape')
f2                         = figure(2);
imagesc (disp_fish_image_seg);
title   ('Trace Wound Perimeter');
%-Step C3: Identify and check wound gap region in reference to image
[nodes_xPos, nodes_yPos]                  = getline(f2);
binary_woundgap_seg        = poly2mask(nodes_xPos,nodes_yPos,width_x,height_y);
exist_woundgap             = sum(sum(binary_woundgap_seg));
if exist_woundgap          >  0
    exist_woundgap             = 1;
    bw_woundgap_outline        = bwperim(binary_woundgap_seg);
    Segout_woundgap            = imoverlay(fluorFishImg,...
                               binary_woundgap_seg,[1 0 0]);
    f3                         = figure(3);
    imshow(Segout_woundgap);
    title   ('Trace Wound Perimeter');
else
    warning ('Wound Perimeter was not Properly traced')
    disp    ('Please rerun script')
    disp    ('Supplementary_ScriptS2 Terminated')
    return
end;
%% Section D:  Automate wound Region ROI image segmentation
clc; close all
%-Step D1:  Predefine variables
%           User-determiend radial Distance from wound Gap
roi_rad_dist               = 15; % user based
%           Arrays  used to store computations
Arr_posCirCent             ={0};
Arr_xCirCent               ={0};
Arr_yCircCent              ={0};
Binary_circMASK                      ={0}; %  = combo;
Binary_perimROI                      ={0};%  = raw_combo;
point_dists                =(0);
%-Step D2-D6:  Employ for loop to address each node
lensNode                   = length(nodes_xPos);
countTraceNodes            = 0;
for countTraceNodes        = 1:(lensNode-1)
%-Step D2: Calculate Line equation between traced nodes
    temp_cirCenter                 = 0; % clear variable
    %       First or iniial node position
    temp_xPos_i                     = nodes_xPos(countTraceNodes);
    temp_yPos_i                     = nodes_yPos(countTraceNodes);
    %       Second or final node position
    if countTraceNodes == lensNode
       temp_xPos_f                  = nodes_xPos(1);
       temp_yPos_f                  = nodes_yPos(1);
    else
       temp_xPos_f                  = nodes_xPos(countTraceNodes+1);
       temp_yPos_f                  = nodes_yPos(countTraceNodes+1);
    end;
    %       Display trace line between the two nodes
    f2                              = figure(2+countTraceNodes);
    imagesc (fluorFishImg);
    hold    on
    plot    (nodes_xPos,nodes_yPos,'LineWidth',2,'Color','r')
    plot    (temp_xPos_i,temp_yPos_i,'c*','LineWidth',2)
    plot    (temp_xPos_f,temp_yPos_f,'y*','LineWidth',2)
    hold    on
    %       Calculate Euclid Distance Between Nodes
    point_dists(countTraceNodes)    = pdist([temp_xPos_i,...
                                    temp_yPos_i;temp_xPos_f,...
                                    temp_yPos_f]);
    temp_point_dist                 = point_dists(countTraceNodes);
    %       Number of circle centers for every pixel between nodes
    numCircles                      = round(temp_point_dist);
    %       Calculate line equation in reference to image
    lineCoeffs                      = polyfit([temp_xPos_i, temp_xPos_f],...
                                    [temp_yPos_i, temp_yPos_f],1);
    aCoeff                          = lineCoeffs (1);
    bCoeff                          = lineCoeffs (2);
    min_xPos    = min(nodes_xPos(countTraceNodes:countTraceNodes+1));
    max_xPos    = max(nodes_xPos(countTraceNodes:countTraceNodes+1));
    min_yPos    = min(nodes_yPos(countTraceNodes:countTraceNodes+1));
    max_yPos    = max(nodes_yPos(countTraceNodes:countTraceNodes+1));
%-Step D3: Map circle centers for wound region ROI
    %      Determine circle center positions respective to image
    cirCent_xPos                    = linspace(min_xPos,max_xPos,...
                                    numCircles);
    cirCent_yPos                    = aCoeff*cirCent_xPos+bCoeff;
    temp_cirCenter                  =[cirCent_xPos',cirCent_yPos'];
    Arr_posCirCent{countTraceNodes} = temp_cirCenter;
    Arr_xCirCent{countTraceNodes}   = cirCent_xPos;
    Arr_yCircCent{countTraceNodes}  = cirCent_yPos;
    %     Define Variables for sub for loop
    arr_combomask                   ={0};
    binary_indyCirImages            ={0};
    mean_maskPixInten               =(0);
%-Step D4-D6: Sub for loop defining finite circles comprising ROI
    for countCirc                   = 1:numCircles
%-Step D4: Create circle boundaries at center position
       indy_xCirCent               = cirCent_xPos(countCirc);
       indy_yCirCent               = cirCent_yPos(countCirc);
       THETA                       = linspace(0, 2 * pi, 1000);
       RHO                         = ones(1, 1000) * roi_rad_dist;
       [xBounds,yBounds]           = pol2cart(THETA, RHO);
       pos_xBounds                 = xBounds + indy_xCirCent;
       pos_yBounds                 = yBounds + indy_yCirCent;
       plot(pos_xBounds, pos_yBounds, '-',...
           'linewidth',2,'color','g');
%-Step D5: Create binary mask of circle ROI in fish tissue
       temp_xPos                   = indy_xCirCent  ;
       temp_yPos                   = indy_yCirCent;
       [xMesh,yMesh]               = meshgrid(-(temp_xPos-1):(width_x-temp_xPos),...
                                   -(temp_yPos-1):(height_y-temp_yPos));
       binary_circMask             =((xMesh.^2+yMesh.^2)<=roi_rad_dist^2);
       %       Corrected circle mask to be inside fish tissue
       binary_corrMask       = binary_circMask & binaryFishImg_Final;
%-Step D6: Calculate mean mask pixel intensity of combined circle masks
       maskPixNum                  = sum(sum(binary_corrMask));
       maskPixInten                =
sum(sum(uint16(binary_corrMask).*fluorFishImg))/maskPixNum;
       mean_maskPixInten(countCirc)    = maskPixInten;
       binary_indyCirImages{countCirc} = binary_corrMask;
       %  Combine Individual Circle Masks in fish issue to complete ROI
       if countCirc==1
           binary_circComb     = binary_circMask;
           binary_corrCombo    = binary_corrMask;
       else
           binary_circComb     = binary_circComb + binary_circMask;
           binary_circComb     = im2bw(binary_circComb);
           binary_corrCombo    = binary_corrCombo + binary_corrMask;
           binary_corrCombo    = im2bw(binary_corrCombo);
       end;
       arr_combomask{countCirc}= binary_corrCombo;
       hold on
    end;
    Binary_circMASK{countTraceNodes}   = binary_corrCombo;
    Binary_perimROI{countTraceNodes}   = binary_circComb;


end;
```

```
%% Section E:  Display and Check Final wound perimeter Image Segmentation
clc; close all;
%-Step E1:  Combine ROI masks
dispImg                  = fluorFishImg;
blankImg                 = binaryFishImg_Final;
for countROIs            = 1:countTraceNodes
    if countROIs        == 1
       woundPerimMask        = Binary_circMASK{countROIs};
       circRawMask           = Binary_perimROI{countROIs};
    else
       woundPerimMask        = woundPerimMask + Binary_circMASK{countROIs};
       woundPerimMask        = im2bw(woundPerimMask);
       circRawMask           = circRawMask + Binary_perimROI{countROIs};
       circRawMask           = im2bw(circRawMask);
    end;
    blankImg            = imoverlay(blankImg,...
                        bwperim(Binary_circMASK{countROIs}),...
                        0.3.*rand(1,3));
    dispImg             = imoverlay(dispImg,...
                        bwperim(Binary_perimROI{countROIs}),...
                        0.6.*rand(1,3));
 end;
%-Step E2:  Display ROI masks
circImgPerim        = imoverlay(fluorFishImg,bwperim(circRawMask),[1 1 1]);
finalImgPerim       = imoverlay(fluorFishImg,bwperim(woundPerimMask),[1 1 0]);
figure();
imshow(circImgPerim);
title('Complete Circles Final Image');
hold on
plot(nodes_xPos,nodes_yPos,'LineWidth',2,'Color','r')
hold off
figure(); imshow(finalImgPerim); title('Wound Region Final Image'); hold on
plot(nodes_xPos,nodes_yPos,'LineWidth',2,'Color','r')
hold off
%% Section F: Display and Check Final wound ROI Image Segmentation
%-Step F1: Remove Confounding fluorescent debris between wound gap
if exist_woundgap       >0
   exist_woundgap         = 1;
   binaryFinal            = ~woundPerimMask | binary_woundgap_seg;
   binaryFinal            = ~binaryFinal;
else
end;
%-Step F2: Display final ROI Image Segmentation
FinalSegout             = imoverlay(binaryFishImg_Final,...
                      bwperim(binaryFinal),[1 0 1]);
figure  ();
```

```
imshow  (FinalSegout);
title   ('Fish Mask');
hold    on
plot    (nodes_xPos,nodes_yPos,'LineWidth',2,'Color','r')
hold    off
figure();
imshow  (binaryFinal);
title   ('only regions of interest');
hold    on
%% Section G:  Compute Fluorescent Intensity in ROI
areaROI                 = sum(sum(binaryFinal));
meanPixelIntensity      = sum(sum(uint16(binaryFinal).*fluorFishImg))/areaROI;
```

Script S2 Description: Demonstrates the V-shaped image segmentation functionality within *Zirmi* as a reproducible means to acquire raw pixel intensity values and eliminate confounding fluorescence automatically within the wound gap. This is difficult and time consuming to perform manually.

Script S3

```
%% Supplementary_ScriptS3
%   Zirmi Source Information can be found at:
%   <https://github.com/ADParedes/Zirmi>
%   Written By: Andre Daniel Paredes | email: andre.paredes@ymail.com
%   Matlab Source Information can be found at:
%   <https://www.mathworks.com/help/images/image-analysis.html>
%   Description:  Manual tracing technique to image segment wound region and acquire
%   raw pixel intensity
%% Define the following Directories appropriately
cd          (dirScript_Z_Supplementary)
addpath     (dirScript_E_Functions)
%% Section A: Read Image Data and Preprocess
%-Step A1: Read Image data into workspace from preselected directory
fluorFish                   = imread('fluorescent_fish.tif');
%-Step A2: Format fluorescen_fish image to a standardized bit format to streamline processing
bytes                       = 2^16-1;
fluorFishImg                = im2uint16(fluorFish);
%% Section B: Manually Image Segment Wound Region and Display
disp        ('Outline the Wound Region')
close all;
imagesc     (fluorFishImg);
colormap    ('gray');
movegui     ('northeast');
title       ('ROS REGION');
RosRegion                   = roipoly();
imshow      (RosRegion)
%% Section C:  Compute Fluorescent Intensity in ROI
areaROI                     = sum(sum(RosRegion));
meanPixelIntensity          = sum(sum(uint16(RosRegion).*fluorFishImg))/areaROI;
display(strcat('Raw Pixel Intensity (abu):',num2str(meanPixelIntensity)))
display(strcat('Region area (pixel^2):',num2str(areaROI)));
```

Script S3 Description: Manual tracing technique to image segment wound region and acquire raw pixel intensity