Appendix A

nimage Code

function[out] = nimage19(hotthresh,minframe, maxframe, changedefault)

%% Data to be examined by this function must be a series of separate 8-bit grayscale tiff image files, numbered sequentially (000.tif to XXX.tif) and saved in the same folder.

%% This code has been optimixed for MATLAB 6.5 and 7.0 running on the Windows XP operating system. For matlab 7.0 it is necessary to first type in the command: feature('javafigures',0). Note that MAC OS X10 cannot run this function because of unresolved MATLAB java bugs.

%% Recommended system: 2.0GH or faster pentium 4 processor with a minimum of 1GB RAM.

%%——— Explanation of Variables ——————-
%% HOTTHRESH: sets a threshold for pixel intensity. Any pixel with intensity greater than HOTTHRESH will be used as a starting pixel to search. If no pixels are wanted this way, set HOTTHRESH = 256.

%% MINFRAME and MAXFRAME define the size of the stack (and should be numbers).

%% CHANGEDEFAULT = 1 allows user to change all default values for SD, search box size, comparison cube size, interactively.

```
%% LAST UPDATED: 2/02/05, 4:48 pm

[filename, pathname] = uigetfile(‘.tif’, ‘Select an image from input stack’);
cd(pathname);
thresh = 3;
S = 3;
%% search square radius - when a pixel P is being checked, S pixels in each
direction away from P are also examined.

N = 10;
%% outer square radius (defines the outer limit of x-y shell for generating mean and
SD)

M =9;
%% inner square radius (defines the inner limit of x-y shell for generating mean and
SD)

R1 = 10;
%% outer depth radius (defines the outer limit of z shell for generating mean and
SD)

R2 = 9;
%% inner depth radius (defines the inner limit of z shell for generating mean and
SD)

sizeallhotpixold = [];

% ———- Allows for interactive changing of parameters for search ————

if changedefault == 1

    disp(‘Current parameters: ’)

    disp(‘Threshold, thresh: ’); disp(thresh)

    disp(‘Search square radius, S: ’)
    disp(S)

    disp(‘Outer square radius, N: ’)

    disp(N)

    disp(‘Inner square radius, M:’)

    disp(M)

    disp(‘Outer depth radius, R1:’)

    disp(R1)

    disp(‘Inner depth radius, R2:’)

    disp(R2)

    changevar = input(‘Would you like to change any of these parameters?
    y/n: ’, ‘s’);

    while (changevar == ‘y’)

        vartochange = input(‘Enter the letter(s) that correspond to the
        parameter you would like to change: ’, ‘s’);

        switch vartochange

            case (‘S’)
            S = input(‘Enter new value for search square radius: ’)
            changevar = input(‘Would you like to change another
            parameter? ’, ‘s’);

            case (‘N’)

                N = input(‘Enter new outer square radius: ’)
                changevar = input(‘Would you like to change
                another parameter? ’, ‘s’);

            case (‘M’)

                M = input(‘Enter new value for inner square
                radius: ’)
```

```
                changevar = input(‘Would you like to change
                another parameter? ’, ‘s’);
            case (‘R1’)
                R1 = input(‘Enter new value for outer depth
                radius: ’)
                changevar = input(‘Would you like to change
                another parameter? ’, ‘s’);
            case (‘R2’)
                R2 = input(‘Enter new value for inner depth
                radius: ’)
                changevar = input(‘Would you like to change
                another parameter? ’, ‘s’);
            end
        end
    end

% ——— Calculate sdx ————————

%% sdx is the value by which number of pixels in shell will be divided by. Use
higher sdx for larger N, M, R1 and R2.

  if R1>N
    tempvar = R1;
else tempvar = N;
  end
  if tempvar<6
    sdx = 2;
elseif (tempvar>=6 & tempvar <10)
```

```
        sdx = 4;
elseif (tempvar >=10)
        sdx = 8;
end
    sdx


% ———————————————————
tn3 = (2*N+1)*(2*N+1)*(2*R1+1); %%sizes of the search boxes.
tm3 = (2*M+1)*(2*M+1)*(2*R2+1);


% ——— determine filename ————–
[filenameout] = rmtif(filename); %% remove '.tif' from end of filename.
[filenameout2] = rmnum(filenameout); %% remove 'xxx' from end of filename.

fileplace = strcat(pathname, filenameout2);
sampleimagename = strcat(fileplace, '-001.tif');
sampleimage = imread(sampleimagename, 'tif');
[realSIZEy realSIZEx] = size(sampleimage);

SIZEx = realSIZEx + 2*(N+S); %% size in pixels of individual images.
SIZEy = realSIZEy + 2*(N+S);

mf = maxframe;

matdepth = maxframe-minframe+1+2*R1; %% calculates the depth of the
processing matrix (with the padding).
out = zeros(SIZEy,SIZEx,matdepth); %%initialize output stack
out = uint8(out);
```

```
exnow = []; %% vector where the indeces of pixels currently being examined are
recorded.
exd = []; %% vector where the indices of pixels already examined are recorded.
toex=[]; %% vector where the indices of pixels to examine is recorded.
raw = out;

C = floor(mean([minframe maxframe])); %% search starting frame - starting pixel
defined on this frame.
A= floor(realSIZEy/2); %%Y coordinate
B = floor(realSIZEx/2); %%X coordinate

startind = [A B C]
C = C - minframe + R1; %% converting original coordinates to padded coordinates.
A = A + N+S;
B = B+N+S;
startpix = sub2ind([SIZEy SIZEx matdepth], A, B, C);

testout = out; %% generate offset box

testout(A-S:A+S, B-S:B+S, C-1:C+1) = 1;
boxoff = find(testout) - startpix;
size(boxoff)
exnow = [];

testout = out; %% generate offset box
testout(A-M:A+M, B-M:B+M, C-R2:C+R2) = 1;
Mboxoff = find(testout) - startpix;

testout = out; %% generate offset box
```

```
testout(A-N:A+N, B-N:B+N, C-R1:C+R1) = 1;
Nboxoff = find(testout) - startpix;

%% make the shell (mean comparison pixels) smaller by taking only 1/sdxth of all
pixels in the defined shell.
shellidx = setdiff(Nboxoff,Mboxoff);
shellidx = subsample(shellidx',sdx)'; % uses 1/sdxth the values, use less if possible.

testout = out;
testout(N+S+1:N+S+realSIZEy, N+S+1:N+S+realSIZEx,
R1+1:R1+maxframe-minframe) = 1;
hotbox = testout;
testout = testout == 0;
exd = find(testout);

out(startpix) = 1; %%records the staring pixel(s) in output stack.

figure

%% image input loop

for ii = minframe-R1:maxframe+R1 %% loading original images into 'raw', and
recording all pixels above threshold, TT.

    if (ii > maxframe)

        irt = 2*maxframe-ii; %% adds padding to the stack.

    elseif (ii < minframe)

        irt = 2*minframe-ii;

    else
        irt = ii;
    end
    if (irt < 10)
        fname = sprintf(‘%s-00%d.tif’,fileplace, irt);
    elseif (irt < 100)
        fname = sprintf(‘%s-0%d.tif’, fileplace, irt);
    else
        fname = sprintf(‘%s-%d.tif’,fileplace, irt);
    end
    picture = imread(fname); %% read the image
    picture = padarray(picture, [N+S N+S], ‘symmetric’, ‘both’);

    oi = ii-(minframe-R1)+1; %%this is the number of the image in process-
    ing coordinates (with padding) of the image that was just read in. irt is
    the actual number of the original image.

    raw(:,:,oi) = picture; %%add original image just loaded to a stack, called
    ‘raw’ (already initialized above), as image number ‘oi’.

    %% 1st processing.
    temp = wiener2(raw(:,:,oi),[3 3]); %% low pass filter images
    if (minframe <= ii & ii <= maxframe)
        out(:,:,oi) = uint8(temp >= hotthresh); %% first output stack
        is all pixels above threshold value, TT, from low passed image
```

```
        oi. TT specified in the function call. (‘out’ already initialized
        above.)

    end

end

raw = imsubtract(raw, uint8(raw==255));

imshow(out(:,:,10),[0 1])

tempstring = input (‘Do you want to load in a previous out array? (y/n): ’,‘s’);
if (tempstring == ‘y’)

    ‘yes’

    saveoutprev = 0;

    [filename_p, pathname_p] = uigetfile(‘.tif’, ‘Select an image in the previous
    nimage stack’);

    [filenameout_p] = rmtif(filename_p); %% remove ‘.tif’ from end of file-
    name.

    [filenameout2_p] = rmnum(filenameout_p); %% remove ‘xxx’ from end of
    filename.

    fileplace_p = strcat(pathname_p, filenameout2_p);


    %% previous image input loop

    for ii = minframe-R1:maxframe+R1 %% loading previously created out
    array

        if (ii > maxframe)

            irt = 2*maxframe-ii; %% adds padding to the stack.
    elseif (ii < minframe)
        irt = 2*minframe-ii;
    else
        irt = ii;
    end

    if (irt < 10)
        fname = sprintf('%s-00%d.tif',fileplace_p, irt);
    elseif (irt < 100)
        fname = sprintf('%s-0%d.tif', fileplace_p, irt);
    else
        fname = sprintf('%s-%d.tif',fileplace_p, irt);
    end

    picture = imread(fname); %% read the image
    picture = padarray(picture, [N+S N+S], 'symmetric', 'both');

    oi = ii-(minframe-R1)+1; %%this is the number of the image
    in processing coordinates (with padding) of the image that was
    just read in. irt is the actual number of the original image.

    out(:,:,oi) = picture; %%add original image just loaded to a
    stack, called 'raw' (already initialized above), as image number
    'oi'.

end
```

```
    out = (out == 254);

    exd = [exd; find(out)];

else

    'no'

end


nimageoutnum = 1;
%% main loop flag
PROCESS = 1;

%% this is main look-process loop
while(PROCESS == 1)

    current_char = 0;

    minoi = 1+R1;

    maxoi = R1+maxframe-minframe+1;

    oi = minoi;


    tempout1 = immultiply((raw(:,:,oi)), (out(:,:,oi)==0));

    tempout2 = immultiply((out(:,:,oi)==1), uint8(ones(SIZEy, SIZEx).*255));

    outtemp = imadd(tempout1, tempout2);

    imshow(outtemp);

    rcmap = colormap; %% make a colormap that prints 256 as red

    rcmap(256,2) = 0;

    rcmap(256,3) = 0;
```

```
pixval('ON');


while(current_char ~= 113)

    keyinput = waitforbuttonpress;

    current_char = double(get(gcf,'CurrentCharacter'));


    if ( isempty(current_char))

        switch current_char

            case 1 % Ctrl + A

                disp('Ctrl+A pressed');


            case 28

                %% left key
                oi = oi-10;
                if (oi < minoi)
                    oi = minoi;
                end
                tempout1 = immultiply((raw(:,:,oi)), (out(:,:,oi)==0));
                tempout2 = immultiply((out(:,:,oi)==1),
                uint8(ones(SIZEy, SIZEx).*255));
                outtemp = imadd(tempout1, tempout2);
                imshow(outtemp, rcmap);
                pixval('ON');


            case 29

                %% right key
                oi = oi+10;
                if (oi > maxoi)
    oi = maxoi;
  end
  tempout1 = immultiply((raw(:,:,oi)), (out(:,:,oi)==0));
  tempout2 = immultiply((out(:,:,oi)==1),
  uint8(ones(SIZEy, SIZEx).*255));
  outtemp = imadd(tempout1, tempout2);
  imshow(outtemp, rcmap);
  pixval('ON');

case 30
  %% up key
  oi = oi+1;
  if (oi > maxoi)
    oi = maxoi;
  end
  tempout1 = immultiply((raw(:,:,oi)), (out(:,:,oi)==0));
  tempout2 = immultiply((out(:,:,oi)==1),
  uint8(ones(SIZEy, SIZEx).*255));
  outtemp = imadd(tempout1, tempout2);
  imshow(outtemp, rcmap);
  pixval('ON');

case 31
  %% down key
  oi = oi-1;
  if (oi < minoi)
    oi = minoi;
  end
  tempout1 = immultiply((raw(:,:,oi)), (out(:,:,oi)==0));
  tempout2 = immultiply((out(:,:,oi)==1),
  uint8(ones(SIZEy, SIZEx).*255));
  outtemp = imadd(tempout1, tempout2);
  imshow(outtemp, rcmap);
  pixval('ON');

case 113
  %% q
  %% quit input loop

case 115
  %% s
  %% save current out array

  disp ('Saving nout file...')
  if (tempstring == 'y' & saveoutprev ==
  0)
    [nimageoutnum] = retnoutnum(filenameout2_p);
    nimageoutnum = nimageoutnum + 1;
    nimageoutnums = num2str(nimageoutnum);
    noutpathname = strcat(pathname, 'nout',
    nimageoutnums);
    [sucess, message, messageid]= mkdir(noutpathname);
    noutfilename = strcat(noutpathname,'\',
    'nout', nimageoutnums)
    saveoutprev = 1;
  else
    nimageoutnums = num2str(nimageoutnum);
    noutpathname = strcat(pathname, 'nout',
    nimageoutnums);
    [sucess, message, messageid]= mkdir(noutpathname);
    noutfilename = strcat(noutpathname,'',
    'nout', nimageoutnums)
  end

  for ii = minframe:maxframe %% write the
  out array images to file.
    oi = ii - minframe + 1+R1;
    if (ii < 10)
     fname2 = sprintf('%s-00%d.tif',noutfilename,ii);
    elseif (ii < 100)
     fname2 = sprintf('%s-0%d.tif',noutfilename,ii);
    else
     fname2 = sprintf('%s-%d.tif',noutfilename,ii);
    end

    blah = raw(:,:,oi) > 253; %%this pre-
    pares the images for scion image's -
    now a logical.
    blah = uint8(blah); %back to image.

    % % These lines of code get rid of all
    255, 254 and 0 pixels and replace them
    with either 253, 252, 1 respectively.
    bb = raw(:,:,oi) == 0;

    blah2 = double(raw(:,:,oi)) - double(blah)
    - double(blah)+double(bb);
```

```
        blah3 = blah2.*(double(out(:,:,oi))==0)
        + double(out(:,:,oi)).*254; %%so the
        hot pixels come out as 254.

        imwrite(uint8(blah3((N+S+1):(SIZEy-
        N-S),(N+S+1):(SIZEx-N-S))), fname2,
        ‘TIFF’, ‘Compression’, ‘none’);

    end
    nimageoutnum = nimageoutnum + 1;
    disp(‘Nout file saved.’)

case 116
    %% t
    % terminate processing loop
    PROCESS = 0;

case 102
    %% f
    %% move to first frame
    oi = minoi;

    tempout1 = immultiply((raw(:,:,oi)), (out(:,:,oi)==0));
    tempout2 = immultiply((out(:,:,oi)==1),
    uint8(ones(SIZEy, SIZEx).*255));
    outtemp = imadd(tempout1, tempout2);
    imshow(outtemp, rcmap);
    pixval(‘ON’);
```

```
case 108
    %% l
    %% move to last frame
    oi = maxoi;

    tempout1 = immultiply((raw(:,:,oi)), (out(:,:,oi)==0));
    tempout2 = immultiply((out(:,:,oi)==1),
    uint8(ones(SIZEy, SIZEx).*255));
    outtemp = imadd(tempout1, tempout2);
    imshow(outtemp, rcmap);

case 32
    %% <space>
    pt = get(gca,‘CurrentPoint’);
    xpt = pt(1,1);
    ypt = pt(1,2);
    chosenpix = sub2ind([SIZEy SIZEx mat-
    depth], ypt, xpt, oi);
    out(chosenpix) = 1;
    tempout1 = immultiply((raw(:,:,oi)), (out(:,:,oi)==0));
    tempout2 = immultiply((out(:,:,oi)==1),
    uint8(ones(SIZEy, SIZEx).*255));
    outtemp = imadd(tempout1, tempout2);
    imshow(outtemp, rcmap);
    pixval(‘ON’);

otherwise
    current_char;
end
```

```
    end
end
‘out of processing loop’
pixval(‘OFF’);
clear tempout1
clear tempout2
clear outtemp
‘about to find newpix’

%%% – examine around all new pixels turned hot
newpix = find(out);
‘found newpix’
newpix = setdiff(newpix,exd);
sizenew = size(newpix)
for counter = 1:sizenew(1)
    exnow = [exnow; boxoff+ newpix(counter)];
end
exnow = setdiff(exnow, exd);
out = out & hotbox;
```

round = 0; %% Prints out a counter so that can tell what processing the program is on. Cycle until no changes in out - processing steps:

while(isempty(exnow) == 0); %% while ‘out’ and ‘prevout’ have at least one row different follow the alogrithm below.

```
    round = round + 1
    sizeexnow = size(exnow)
    for counter = 1:sizeexnow(1)
        temp1 = double(raw(shellidx + exnow(counter)));
        mt1 = mean(temp1);
        std1 = std(temp1);

        mt2 = double(raw(exnow(counter)));

        if ( mt2-mt1 > thresh*std1 — raw(exnow(counter))
        > hotthresh)
          out(exnow(counter)) = 1;
          toex = [toex; boxoff+ exnow(counter)];
        end
    end

    exd = [exd ;exnow];
    exnow = setdiff(toex, exd);
end
%% remove speckles
allhotpix = find(out);
sizeallhotpixold
sizeallhotpix = size(allhotpix)
sizeallhotpixold = sizeallhotpix;
[a, b, c] = ind2sub([SIZEy,SIZEx,maxframe-minframe+1+2*R1],
allhotpix);

for n = 1:sizeallhotpix(1)
    temp1 = double(reshape(out(a(n)-2:a(n)+2,b(n)-2:b(n)+2,c(n)-
    1:c(n)+1),75,1));
```

```
                if ( sum(temp1) < 4)
                      out(a(n),b(n),c(n)) = 0;
                end
           end
      end %% end of main look-process while loop


% ————————- SUBFUNCTION: pad the array, top, bottom, and sides ——

function [out] = pad(pixelstoadd, inputmatrix, sizeofinputmatrix) %% function to
add padding to stack. called above.

leftside = inputmatrix(:,2:pixelstoadd+1); leftside = fliplr(leftside);

rightside = inputmatrix (:, (sizeofinputmatrix -1 :-1:
(sizeofinputmatrix-pixelstoadd)));

outputmatrix1 = [leftside, inputmatrix, rightside];

top = outputmatrix1(2:pixelstoadd+1, :); top = flipud(top); bottom =
outputmatrix1 (sizeofinputmatrix-1:-1:

(sizeofinputmatrix-pixelstoadd), :);
out = [top; outputmatrix1; bottom];

% ———— SUBFUNCTION: remove '.tif' from end of filename —-

function [filenameout] = rmtif(filename);
```

```
size_filename = size(filename);

size_filename = size_filename(2);

sf = size_filename - 4;

filenameout = filename(1:sf);


% ———— SUBFUNCTION: remove '-xxx' from end of filename —-


function [filenameout] = rmnum(filename);

size_filename = size(filename);

size_filename = size_filename(2);

sf = size_filename - 4;

filenameout = filename(1:sf);


% ———- SUBFUNCTION: retrieve nout number ——————


function [noutnum] = retnoutnum(filename);

size_filename = size(filename, 2);

noutnum = filename(5:size_filename);

noutnum = str2num(noutnum);


% ——— SUBFUCNTION: get number of image from filename ——-


function [imnum] = fimnum(filename);

size_filename = size(filename, 2);

sf = size_filename - 2;

imnum = filename(sf: size_filename);

imnum =str2num(imnum);
```