Fractal Dimensions of Leaf Shapes

Introduction

Analyzing leaf shapes in terms of fractal geometry allowed me to apply and extend my knowledge of fractal dimension. Two different methods were used determine fractal dimension of two differently shaped leaves. Self-similar structured leaves found in ferns and herbs and simple, single shaped leaves were subjected to different methods of calculating fractal dimension, the box-counting and the perimeter methods. Actual leaves were scanned by a HP Photosmart C6380 All-in-One in color of varying sizes. All images used in the box-counting method were resized into square images with lengths being a power of 2, ranging from 256 to 1024 pixels. The images were then converted into binary images and the number of "boxes" containing as least one "black" pixel was counted. The *ln*(number of boxes) versus *ln*(pixel-size of box) points was plotted and the fractal dimension was determined by finding the slope of the best fit line for the plotted points. All images used in the perimeter method were converted into binary "outline" images, where the outline of the leaf was a "white" pixel and all other pixels were "black". The size of the image was not changed to any specific measurement. All pixel coordinates were then collected and used to determine the perimeter of the leaf. To determine the fractal dimension of the leaf, the *ln*(perimeter) versus *ln*(step size) were plotted, slope of best fit line was found and subtracted from 1.

Survey of Related Work

According to Iannaccone and Khokha (1996) there are two aspects of fractal geometry, synthesis and analysis. There are several free fractal generators available on the internet in which one is able to generate fractal art and in particular, there is a "fractal fern generator" that uses Excel (Meijer, STMicroeletronics). In general, simple linear transformations of rotations, mirror operations, multiplication and translations are used to recreate the leaf shape. Analysis quantifies given information that can be used to relatively describe the data. In particular, Iannaccone and Khokha state that shapes can be described quantitatively such that a number can be associated with the complexity of the shape. Because of the self-similarity (fractal property) found in leaves, fractal geometry and in particular, fractal dimension, serves as a quantifier of complexity.

Iannaccone and Khokha provide three different methods for determining fractal dimension: box-counting, perimeter, and dilation methods. Hartvigsen (2000) outlines an activity for finding fractal dimensions using the box-counting method on pressed Queen Anne's Lace leaves. The boxes occupied by the leaves are counted. Iannaccone's and Khokha's box-counting method counted only the boxes containing the outline of the leaves. The natural log of the boxes counted versus the natural log of the reciprocal of the box length is plotted and the slope of the best-fit line for these points is the fractal dimension. Kenkel and Walker (1996) list a variety of applications of fractal geometry in biology, including "plant and fungal structures", which mentioned Vlcek and Cheung (1986) measurements of leaf edges. Vlcek and Cheung use the perimeter method of determining the perimeter of the leaf with various "rulers", standardized units of length. The natural log of the perimeter versus the natural log of the ruler lengths is plotted and the slope of the best-fit line for these points is subtracted from 1 to determine the fractal dimension. The dilation method was not pursued.

In his notes, McAndrew (2004) presented and highlighted Matlab's image processing capabilities with many examples, explanations, and command sequences.

Derivation of Mathematics

Box-counting method:

In order to create "square" boxes, the image was resized to a square dimension such that the length, measured in number of pixels, was of a power of 2. This allows for the square image to be equally divided into four quadrants and each subsequent quadrant can be divided into four quadrants, and so on. After the color image was converted into grayscale, the intensity of grayness for each pixel was determined to be greater than a specified threshold or not, creating the binary, black and white, image. The number of boxes containing "black" pixels was noted as a function of the box-size, length of "box". The natural log of all these points were calculated and plotted. A "polyfit" of degree one was calculate and the slope of the line was noted as the fractal dimension of the leaf.

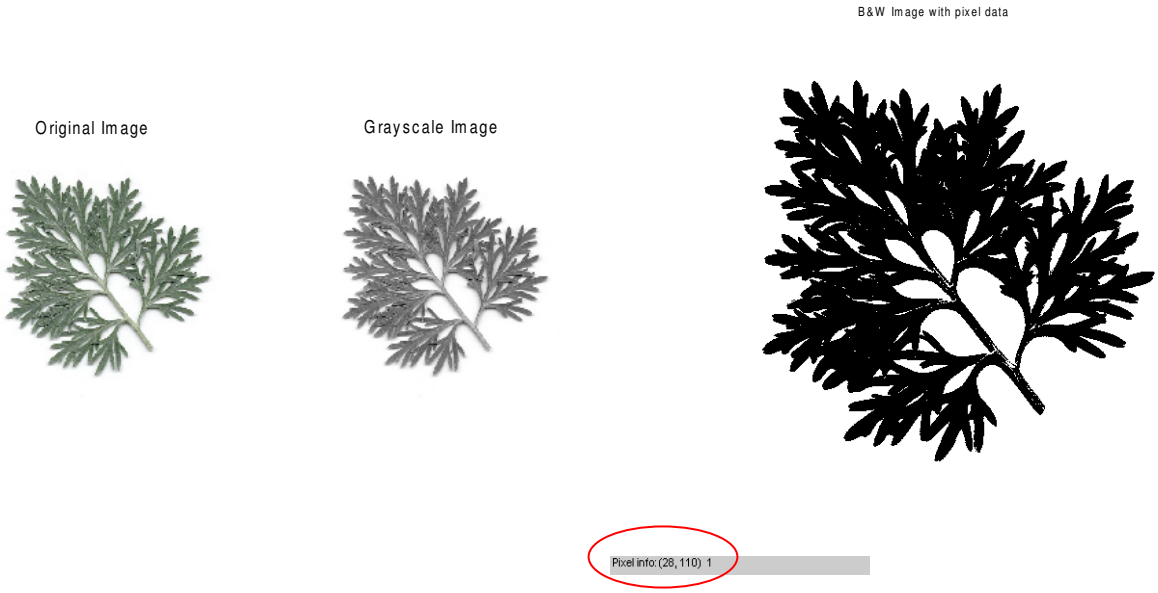
Perimeter method:

In order to determine the perimeter of a leaf, the outline of the leaf needed to be identified. After the image was converted into a binary image, an edge detector (using the Sobel approximation to the derivative, returning points where the gradient of the image is maximum) and boundary maker (tracing the exterior boundaries of objects) were used to create an image that is completely black with a pixel wide white outline of the leaf. This process did not ensure a continuous boundary thus a morphing application was used to "bridge" pixel gaps in the outline. With noted pixel locations outlining the leaf, "rulers" of standard measure were created by locating pixels "x" apart and the distance between these pixels was accumulated to determine the perimeter. The last distance that closed the outline was either calculated as stated or was determined by finding the distance between the last "x" away point and the first point used and added to the accumulation of distances to determine the perimeter. Vlcek and Cheung (1986) state that "Mandelbrot defined the fractal dimension, D, to be an inverse of a least-squares estimate of the slope of a line given in the equation: $\log(N\lambda) = (1-D)\log\lambda + b$" where λ is the curve segment or ruler, and ($N\lambda$) is the total length of the curve, and (1-D) is the slope of the line. Therefore the fractal dimension is found by subtracting the slope of the line from 1.

Graphics

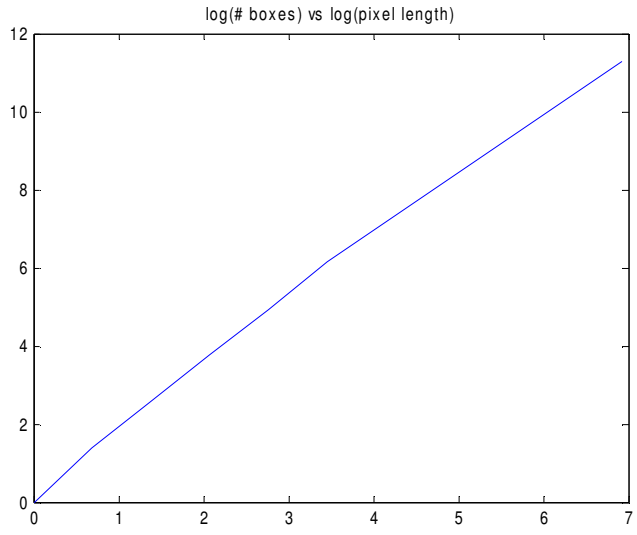
Box-counting method:

The box-counting method was used on leaves exhibiting a fractal structure, namely ferns and herbs found at the local nursery. After the leaves were scanned, the image was resized to measure 512x512 and then were converted into gray scale and displayed:



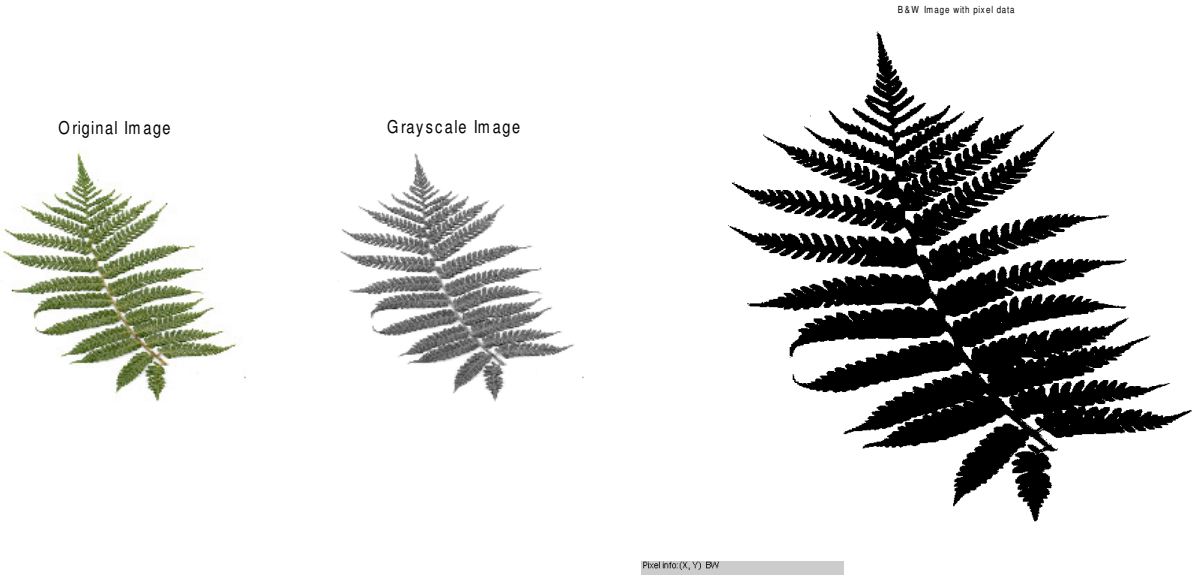
Grayscale images became binary images with a threshold of 190, meaning that if the pixel had a gray value greater than 190 (on a 0 to 255 scale) the pixel became "black", or assigned the value of 0, otherwise the pixel became "white", assigned a value of 1. Moving the cursor over the black and white image displays the pixel location and value, as indicated in the red oval above.

```
EDU>> boxdim('herb1512.jpg')

boxes =

           1           1
           2           4
           4          13
           8          44
          16         139
          32         465
         512       78872

Fractal dimension is

ans =

    1.7899
```

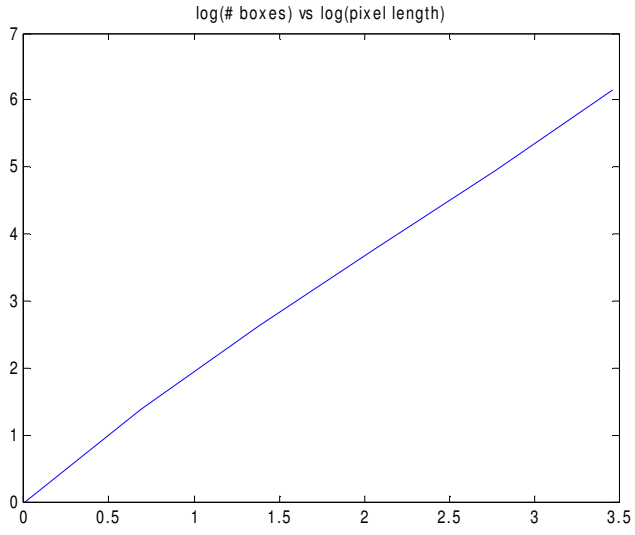


The first column of "boxes" is the reciprocal of the box length, thus the 2 represents a box of dimensions 256x256, or ½ of the original box length. After the image was divided into quadrants and the quadrants containing "black" pixels were counted, the values were placed in the second column of the array, "boxes". After the natural logs of all values in "boxes" were plotted and the best-fit line calculated, the fractal dimension was determined as the slope of the best-fit line.

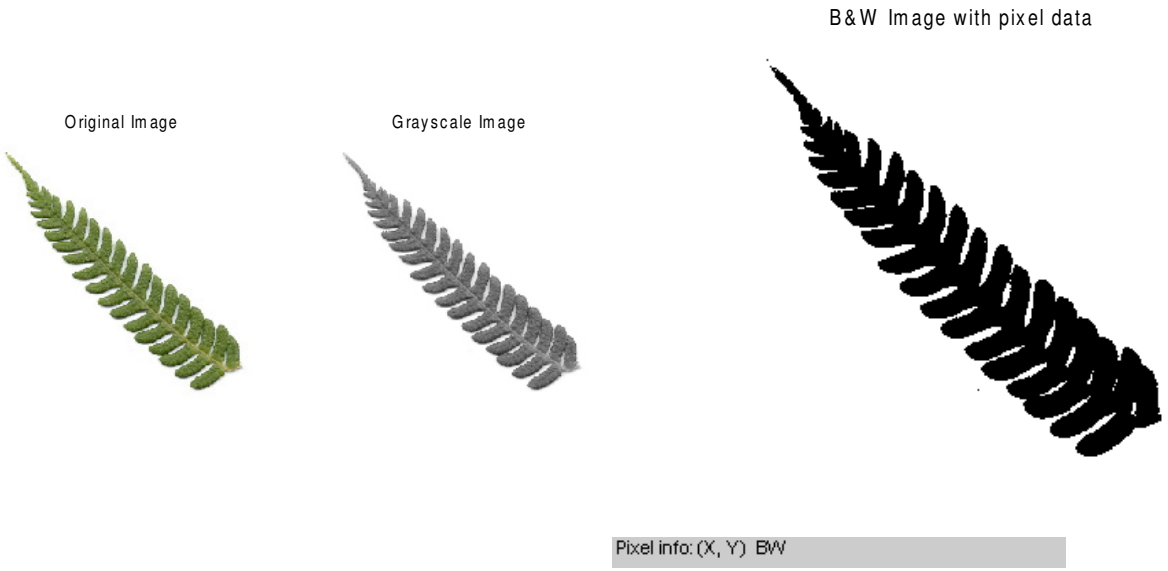
The same box-counting method was used on a fern whose image was 1024x1024, thus changes to the m-file was made to reflect the size.



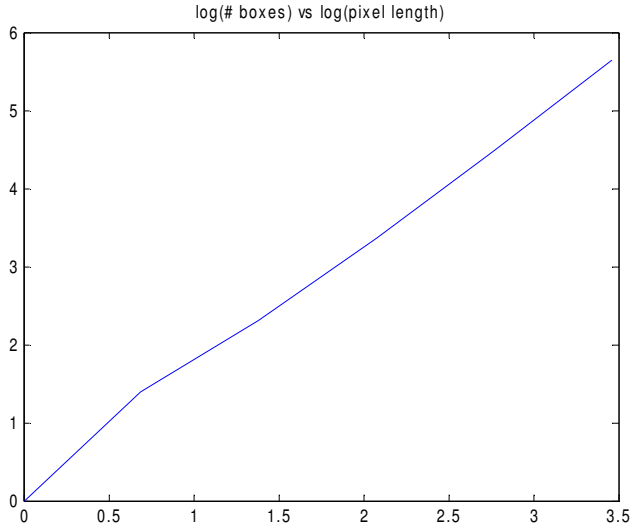

```
EDU>> boxdim('fern1024.jpg')
Warning: Image is too big to fit on screen; displaying at 67%
> In imuitools\private\initSize at 90
  In imshow at 234
  In boxdim at 20

boxes =

           1           1
           2           4
           4          14
           8          45
          16         141
          32         472
        1024      278353

Fractal dimension is

ans =

    1.7576
```



The same box-counting method was used on leaflet of the fern leaf above whose image was 256x256. Since the image measured 256x256, line 47 of boxdim.m (Appendix A) was edited for these results:




```
EDU>> boxdim('fern1256.jpg')

boxes =

           1           1
           2           4
           4          10
           8          29
          16          88
          32         281
         256       12754

Fractal dimension is

ans =

    1.5882
```

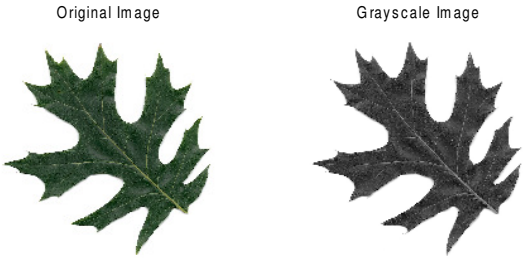


Perimeter method:

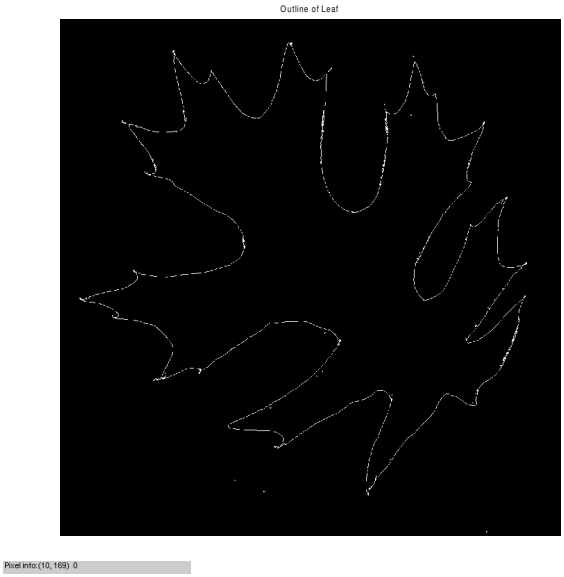
The perimeter method was used on leaves found on plants and trees. The color images, regardless of size, were converted to grayscale. With the use of several Matlab image processing functions, the outline of the leaf was obtained; white outline on black background. Another imaging procedure was used to obtain the pixels of the outline, in order, so that the edgedetector.m (Appendix B) file could calculate its perimeter using various "rulers".

Red oak leaf:

```
EDU>> edgedetector('redoak.jpg')
Warning: Image is too big to fit on screen; displaying at 67%
> In imuitools\private\initSize at 90
  In imshow at 234
  In edgedetector at 24
Using the figure, find a starting pixel on the outline of the leaf
Enter y-coordinate of starting pixel:
186

row =

   186

Enter x-coordinate of starting pixel:
379

col =

   379
```

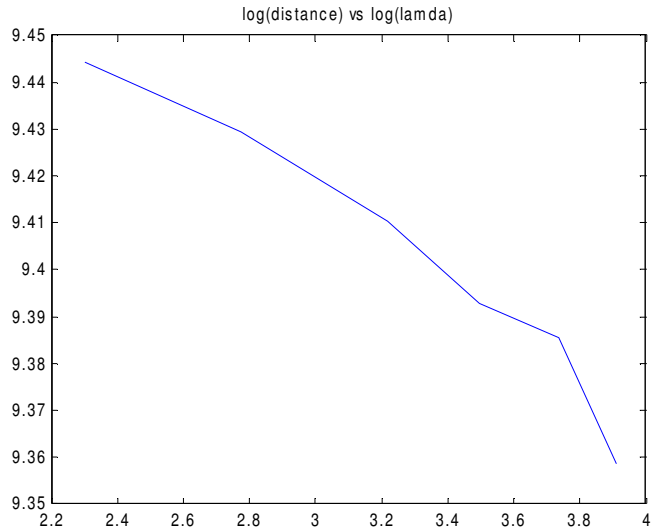


After plotting the outline of the leaf, the user is prompted to locate a pixel on the outline by using the outline figure and pixel information. For this particular example, the pixel (379, 186) was found to have a value of 1, "white" pixel, therefore on the outline. The point selected is noted in blue and the outline is "traced" in green westward from the initial point, indicating the continuous pixels identified as part of the outline.



The outline pixels are in a p x 2 array, where p is the number of "points", pixels, used the make the outline of the leaf. The first and last "points" are the same, to enclose the leaf. The "ruler" is set to 50 pixels such that the distance from the first point (initial point) to the 51st point is calculated. The 51st point becomes the "initial" point and the distance from this initial point to the 51st point (the 101st point in the original list) is calculated, and so on. All these distances are accumulated. If the last initial point is not the last point on the list, the distance between the last initial point and the last point is found and added to the accumulated distances.

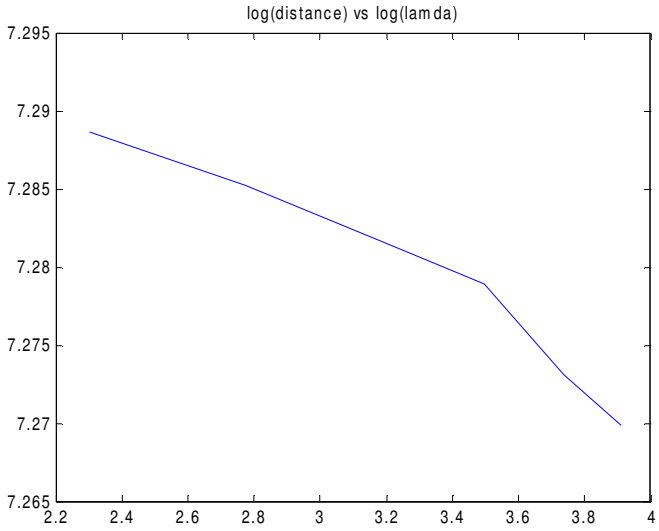
```
data =

  1.0e+004 *

    0.0050    1.1599
    0.0042    1.1915
    0.0033    1.2002
    0.0025    1.2214
    0.0016    1.2450
    0.0010    1.2638

     lamda, distance

log_data =

    3.9120    9.3587
    3.7377    9.3856
    3.4965    9.3929
    3.2189    9.4103
    2.7726    9.4295
    2.3026    9.4444

Fractal dimension is

ans =

    1.0496
```



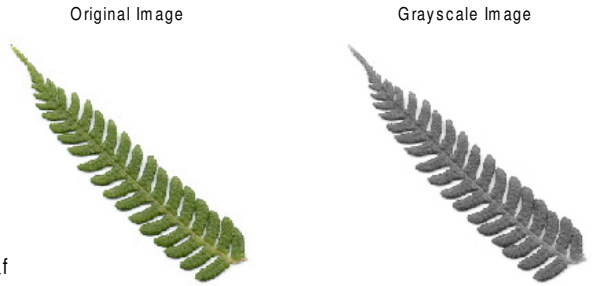
Column one of "data" contains the "ruler" sizes (50, 42, 33, 25, 16, 10 pixel count) while column two contains the perimeter the leaf using such rulers. The log of perimeter versus log of ruler size is plotted. The slope of the best fit line is subtracted from 1 to obtain the fractal dimension.

Live oak leaf:

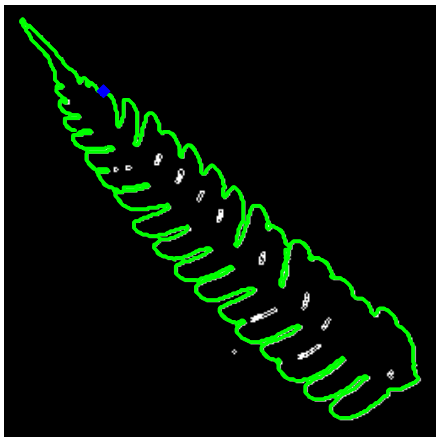
```
EDU>> edgedetector('liveoak.jpg')
Using the figure, find a starting pixel on the outline of the leaf
Enter y-coordinate of starting pixel:
444

row =

   444

Enter x-coordinate of starting pixel:
209

col =

   209


data =

  1.0e+003 *

    0.0500    1.4364
    0.0420    1.4410
    0.0330    1.4494
    0.0250    1.4530
    0.0160    1.4586
    0.0100    1.4636

     lamda, distance

log_data =

    3.9120    7.2699
    3.7377    7.2731
    3.4965    7.2789
    3.2189    7.2814
    2.7726    7.2852
    2.3026    7.2887

Fractal dimension is

ans =

    1.0113
```

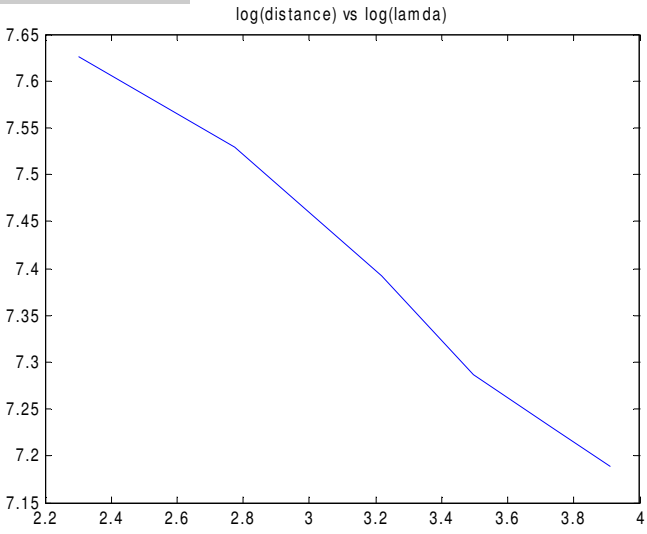


See what happens if this procedure is used on the fern leaflet:




```
EDU>> edgedetector('fern1256.jpg')
Using the figure, find a starting pixel on the outline of the
Enter y-coordinate of starting pixel:
52

row =

    52

Enter x-coordinate of starting pixel:
60

col =

    60


data =

  1.0e+003 *

    0.0500    1.3243
    0.0420    1.3796
    0.0330    1.4603
    0.0250    1.6224
    0.0160    1.8628
    0.0100    2.0496

     lamda, distance

log_data =

    3.9120    7.1886
    3.7377    7.2295
    3.4965    7.2864
    3.2189    7.3917
    2.7726    7.5299
    2.3026    7.6254

Fractal dimension is

ans =

    1.2834
```





Summary and Conclusions

Box-Counting method:

As the leaves become dense to the point of "filling in" a shape, the fractal dimension approaches the value of 2. One issue with this procedure is the possibility of "noise" in the image acquired during the scanning process. A "box" could contain a "black" pixel due to noise and the current procedure would count the box containing the noise, calculating an inaccurate fractal dimension. Images larger than 512x512 took longer computation time however, resizing images to these dimensions distorts the original image as alternate rows and columns are removed from the original image.

Perimeter method:

Though the size of the original images were not altered, smaller images (less than 1000x1000 pixels) were displayed to size and the cursor movement was able to identify a pixel point on the outline with more ease. The magnification of the outline is accessible to help identify a pixel in smaller size images (256x256). The location of the leaf is critical as the entire leaf needs to be placed in the image so that the closed, continuous boundary can be traced accurately. For larger "rulers", the perimeter was not accurately measured as the ruler may not have accounted for a *curve* in the leaf (as seen in the plot) so that the distance between the two points is smaller than the perimeter edge between the two points. Calibrations for the "ruler" size are needed and dependent on the complexity of the leaf-edge.

Appendix A – Matlab Code

```
function boxdim(filename)

% This file calculates fractal dimension of an image
% using the box-counting method
% Usage:  boxdim('filename')
% where 'filename' is a 512x512 image

% load the image
image=imread(filename);
subplot(1,2,1),imshow(image); title('Original Image'); %show color image side by
side with grey image

% convert the image into grayscale
image=rgb2gray(image);
subplot(1,2,2),imshow(image); title('Grayscale Image'); %show grey image side by
side with color image

% show image as true black and white
% resolution of b&w can be adjusted, but set to 190
resol=190;
image=(image>resol);
figure,imshow(image),impixelinfo; title('B&W Image with pixel data')   %show black
and white image, pixel info too

[m,n]=size(image);

% create matrix to hold data, x and y
% where x is 1/length of the box and y is the number of boxes counted to cover the
% leaf
pc=6; %number of (iterations)times counting boxes takes place
boxes=zeros(pc+1,2);

% Establish first column of matrix boxes, the "length of box"
for row=1:pc
    boxes(row,1)=2^(row-1);
end
% Establish boxes(1,2) = 1, the number of boxes used to cover the leaf
% since the box is 512x512 or 1x1
boxes(1,2)=1;
% Establish the boxes(6,2)= # of black pixels of the 512x512 boxes of
% length 1/512
count=0;
for i=1:m
    for j=1:n
        if image(i,j)==0, count=count+1;end
    end
end

%%%%%This needs to change if the dimension is NOT 512X512
boxes(pc+1,1)=256;

boxes(pc+1,2)=count;
```

```
boxes;

% Divide each side of box by 2....to get four quadrants and check each
% quadrant for 0's, parts of the leaf, and count the number of quadrants as
% the number of "boxes" used to cover the leaf
boxcounter=0;

% Divide each side by 2
m=m/2;
n=n/2;

% Create quadrants and Scan quadrant for 0's, black pixels, leaf part
iteration=1;
k=iteration;
quadrants=2^(2*iteration);
    for i=1:m
    for j=1:n
        quad(i,j,k)=image(i,j);
        quad(i,j,k+1)=image(i,j+n);
        quad(i,j,k+2)=image(i+m,j);
        quad(i,j,k+3)=image(i+m,j+n);
    end
    end

count=0;    % Set the counter to 0
% search each quadrant for 0's
for k=1:4
    for i=1:m
        for j=1:n
            if quad(i,j,k)==0, count=count+1; end
        end
    end
    if count>0, boxcounter=boxcounter+1; end
    count=0;
end
boxcounter;
boxes(iteration+1,2)=boxcounter;
boxes;


iteration=2;
boxcounter=0;

% Divide each side by 2
m=m/2;
n=n/2;

% Create quadrants and Scan quadrant for 0's, black pixels, leaf part
quadrants=2^(2*iteration);

s=1;
    for k=1:4
    for i=1:m
    for j=1:n
        quad(i,j,k,s)=quad(i,j,k);
```

```
        quad(i,j,k,s+1)=quad(i,j+n,k);
        quad(i,j,k,s+2)=quad(i+m,j,k);
        quad(i,j,k,s+3)=quad(i+m,j+n,k);
    end
    end
    end

count=0;    % Set the counter to 0
% search each quadrant for 0's
for k=1:4
    for s=1:4
    for i=1:m
        for j=1:n
            if quad(i,j,k,s)==0, count=count+1; end
        end
    end
    if count>0, boxcounter=boxcounter+1; end
    count=0;
    end
end
boxcounter;
boxes(iteration+1,2)=boxcounter;
boxes;

iteration=3;
boxcounter=0;

% Divide each side by 2
m=m/2;
n=n/2;

% Create quadrants and Scan quadrant for 0's, black pixels, leaf part
quadrants=2^(2*iteration);

t=1;
    for s=1:4
    for k=1:4
    for i=1:m
    for j=1:n
        quad(i,j,k,s,t)=quad(i,j,k,s);
        quad(i,j,k,s,t+1)=quad(i,j+n,k,s);
        quad(i,j,k,s,t+2)=quad(i+m,j,k,s);
        quad(i,j,k,s,t+3)=quad(i+m,j+n,k,s);
    end
    end
    end
    end

count=0;    % Set the counter to 0
% search each quadrant for 0's
for t=1:4
    for k=1:4
    for s=1:4
    for i=1:m
        for j=1:n
            if quad(i,j,k,s,t)==0, count=count+1; end
```

```
        end
    end
    if count>0, boxcounter=boxcounter+1; end
    count=0;
    end
    end
end
boxcounter;
boxes(iteration+1,2)=boxcounter;
boxes;

iteration=4;
boxcounter=0;

% Divide each side by 2
m=m/2;
n=n/2;

% Create quadrants and Scan quadrant for 0's, black pixels, leaf part
quadrants=2^(2*iteration);

u=1;
    for t=1:4
    for s=1:4
    for k=1:4
    for i=1:m
    for j=1:n
        quad(i,j,k,s,t,u)=quad(i,j,k,s,t);
        quad(i,j,k,s,t,u+1)=quad(i,j+n,k,s,t);
        quad(i,j,k,s,t,u+2)=quad(i+m,j,k,s,t);
        quad(i,j,k,s,t,u+3)=quad(i+m,j+n,k,s,t);
    end
    end
    end
    end
    end

count=0;    % Set the counter to 0
% search each quadrant for 0's
for u=1:4
    for t=1:4
    for k=1:4
    for s=1:4
    for i=1:m
        for j=1:n
            if quad(i,j,k,s,t,u)==0, count=count+1; end
        end
    end
    if count>0, boxcounter=boxcounter+1; end
    count=0;
    end
    end
    end
end
boxcounter;
boxes(iteration+1,2)=boxcounter;
```

```
boxes;

iteration=5;
boxcounter=0;

% Divide each side by 2
m=m/2;
n=n/2;

% Create quadrants and Scan quadrant for 0's, black pixels, leaf part
quadrants=2^(2*iteration);

v=1;
for u=1:4
    for t=1:4
    for s=1:4
    for k=1:4
    for i=1:m
    for j=1:n
        quad(i,j,k,s,t,u,v)=quad(i,j,k,s,t,u);
        quad(i,j,k,s,t,u,v+1)=quad(i,j+n,k,s,t,u);
        quad(i,j,k,s,t,u,v+2)=quad(i+m,j,k,s,t,u);
        quad(i,j,k,s,t,u,v+3)=quad(i+m,j+n,k,s,t,u);
    end
    end
    end
    end
    end
end

count=0;    % Set the counter to 0
% search each quadrant for 0's
for v=1:4
    for u=1:4
    for t=1:4
    for k=1:4
    for s=1:4
    for i=1:m
        for j=1:n
            if quad(i,j,k,s,t,u,v)==0, count=count+1; end
        end
    end
    if count>0, boxcounter=boxcounter+1; end
    count=0;
    end
    end
    end
    end
end
boxcounter;
boxes(iteration+1,2)=boxcounter;
boxes
```

```
% find log of boxes to create points on the ln(count) vs ln(1/side)
points=zeros(pc,2);

for i=1:pc
    points(i,1)=log(boxes(i,1));
    points(i,2)=log(boxes(i,2));
end


points;

% Graph points (ln(boxcounter) vs ln(1/length of box))
x=points(:,1);
y=points(:,2);
figure,plot(x,y);title('log(# boxes) vs log(pixel length)');% give it a title

% fractal dimension, D, is the slope of plot
[a,b]=polyfit(x,y,1);
fprintf('Fractal dimension is \n'),a(1,1)
```

Appendix B – Matlab Code

```
function edgedetector(filename)

% This file uploads an color image, converts it to grayscale and finds the
% edges of the image.  All images are displayed with the edge image
% (B&W with pixel information).

% load the image
image=imread(filename);
subplot(1,2,1),imshow(image); title('Original Image');%show color image side by
side with grey image

% convert the image into grayscale
image=rgb2gray(image);
subplot(1,2,2),imshow(image); title('Grayscale Image'); %show grey image side by
side with color image

% find the edge of the leaf
% show image as true black and white
% resolution of b&w can be adjusted, but set to 215
resol=215;
image=(image>resol);
image = double(image) + 1; %edge cannot use logical image
image=edge(image,'sobel');
[B,image]=bwboundaries(image,'noholes');
image=bwmorph(image,'bridge'); % makes bridges to enclose the outline
figure,imshow(image),impixelinfo,title('Outline of Leaf')

%find the coordinates of the pixels, in order,
% by using the bwtraceboundary

disp('Using the figure, find a starting pixel on the outline of the leaf')
row=input('Enter y-coordinate of starting pixel: \n')
col=input('Enter x-coordinate of starting pixel: \n')

   contour = bwtraceboundary(image, [row, col], 'W',8,inf,'clockwise');
   if(~isempty(contour))
      hold on;
      plot(contour(:,2),contour(:,1),'g','LineWidth',2);
      hold on;
      plot(col, row,'bx','LineWidth',5);
   else
      hold on; plot(col, row,'rx','LineWidth',5);
   end

% Find the number of pixels that make the outline
 n=length(contour);
% there are n-1 distinct pixels as the first and last pixels are the same

% if a point on the outline is not chosen....
if n<1
    disp('The pixel you have chosen is not on the outline of the leaf.'),
end
```

```
% initialize variables
data=zeros(1,2);

% Find the number of steps to "walk around" leaf with a step size, lambda

% Determine step-size, lambda
for number=1:6
    if number==1
        lambda=50;
    elseif number==2
        lambda=42;
    elseif number==3
        lambda=33;
    elseif number==4
        lambda=25;
    elseif number==5
        lambda=16;
    elseif number==6
        lambda=10;
    else lambda=1;
    end

% initialize variables
steps=0;
remainder=0;
nextpt=zeros(1);
step_dist=0;
distance=0;
initialpt=zeros(1);
delta=zeros(2);

% find pixels exactly lambda away from each other and calculate step
% distance then sum the step distances
initialpt(1)=contour(1,2);
initialpt(2)=contour(1,1);
initialpt;

for i=(lamda+1):lamda:n
    nextpt(1)=contour(i,2);nextpt(2)=contour(i,1);
    nextpt;
    delta(1)=initialpt(1)-nextpt(1);
    delta(2)=initialpt(2)-nextpt(2);
    step_dist=sqrt((delta(1))^2+(delta(2))^2);
    steps=steps+1;
    distance= distance + step_dist;
    initialpt=nextpt;
    initialpt;
end
lambda;
steps;
initialpt;
% Find the distance btwn last pixel used above and the last pixel that
% completes the outline and add to the distance if there are left over
% pixels
remainder=n-steps*lambda;
```

```
if remainder>0
    nextpt(1)=contour(n,2);
    nextpt(2)=contour(n,1);
    nextpt;
    delta(1)=initialpt(1)-nextpt(1);
    delta(2)=initialpt(2)-nextpt(2);
    delta;
    r_dist=sqrt((delta(1))^2+(delta(2))^2); %remainder distance
    distance=distance+r_dist;
end

% Store data
data(number,1)=lamda;data(number,2)=distance;

end

data, disp('     lambda, distance') % Displays lambda size with corresponding
distance

% plot the log steps versus log of lambda

% find log of  values

log_data(:,1)=log(data(:,1));log_data(:,2)=log(data(:,2));


log_data

% Graph points (ln(perimeter) vs ln(#of pixels skipped))
x=0; y=0;
x=log_data(:,1);
y=log_data(:,2);
figure,plot(x,y);title('log(distance) vs log(lambda)');% give it a title

% fractal dimension, D, is the slope of plot
[a,b]=polyfit(x,y,1);

fprintf('Fractal dimension is \n'),1-a(1,1)
```

Appendix C – References

Hartvigsen, G. (2000).The Analysis of Leaf Shape Using Fractal Geometry. *The American Biology Teacher*. *62*, 664-669.

Iannaccone, P. M. & Khokha, M. (1996). *Fractal Geometry in Biological Systems: An Analytical Approach*. CRC Press, pages 173-176.

Kenkel, N.C. & Walker, D.J. (1996). Retrieved April 20, 2009, from Fractals in the Biological Sciences Web site: http://www.umanitoba.ca/botany/LABS/ECOLOGY/FRACTALS/fractal.html

McAndrew, A. (2004). *An Introduction to Digital Images Processing with Matlab: Notes for SCM2511 Image Processing 1*. School of Computer Science and Mathematics: Victoria University of Technology. Retrieved April 30, 2009, from University of Texas at El Paso Web site: http://www.geo.utep.edu/pub/hurtado/5336/other_resources/DigitalImageProcessing_Matlab.pdf

Meijer, B. & STMicroelectronics Biological ESTEEM: Excel Simulations and Tools for Exploratory, Experiential Mathematics. Retrieved April 25, 2009, from BioQUEST Curriculum Consortium Web site: http://www.bioquest.org/esteem/esteem_details.php?product_id=249

Vlcek, J. and Cheung, E. (1986).Fractal analysis of leaf shapes. *Canadian Journal of Forest Research*. *16*, 124-127.