

Computational study of dendritic solidification in
ammonium chloride:
A progress report for Summer 2002

Daniar Hussain¹
Institute of Theoretical Geophysics
DAMTP
Cambridge University
United Kingdom

14 August 2002

¹dhussain@mit.edu
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA
United States of America

Abstract

This brief article reports on the progress made during the summer of 2002 in the study of the dendritic solidification of aqueous ammonium chloride (NH_4Cl). The study consisted of taking digital images of growing dendrites with a microscopic lens and performing image analysis on the pictures. The experimental setup is described and diagrammed, including special procedures to ensure quality images. Next the methods of image analysis that were performed are presented, with results and recommendations for improvement. This article ends with a discussion of future directions for this line of research. A list of references and key words generated during the literature search are presented to aide the parties continuing this work in the future.

Contents

1	Introduction	2
2	Experimental methods	2
3	Data analysis	3
4	Directions for future work	7
5	References and key words	13
6	Appendix: Source code	14

1 Introduction

Solidification is an important process for many industrial and natural phenomena. The phase-change from liquid to solid state releases large quantities of latent heat. When a pure liquid is cooled below its freezing point, a solid is formed; the boundary between solid and liquid phase is generally flat. The evolution of the boundary with time is controlled by the dynamics of heat transfer.

The solidification of binary alloys (that is, two-component liquid systems) exhibit several interesting physical phenomena, with applications in many areas of physical science and engineering. In the case of a binary alloy (i.e., a salt dissolved in water), the boundary between the two phases is morphologically unstable, and creates an intermediate mixture at the boundary of solid and liquid called a mushy-layer. In this case, the evolution of the boundary with time is controlled by the dynamics of both heat and mass transfer, since the solid state is usually of a different composition than the original liquid (i.e., salty water freezes into pure ice).

2 Experimental methods

In this experiment the mushy layer that is formed from the solidification of an aqueous solution of ammonium chloride (NH_4Cl) was studied. Aqueous ammonium chloride exhibits especially branched crystals during crystallization, it is easy to prepare in the laboratory, and behaves in many ways like metallic alloys.

The experimental setup consisted of a small (5cm x 5cm) solidification chamber, filled with NH_4Cl solution. The chamber width can be varied by using different sized spacers. A 1mm width provides better images because this restricts the crystals from forming two layers in the chamber. The chamber is connected to a refrigeration unit with a pump, to cool the lower boundary of the chamber down to varying levels (approx. -15 to -20°C).

A blue dye (Rayner's cochineal substitute food color: acetic acid, E122 carmoisine) was used to increase the contrast of the specimens. Assuming the dye has a density of 1 g/mL like water, to create 26 wt% NH_4C the following quantities were mixed:

13.01 \pm 0.01 g NH_4C
33.0 \pm 0.5 mL distilled water
4.0 \pm 0.1 mL dye

Total weight was 49.50g, and water was added until the weight was 50.09 g.

(Red dye does not dissolve in ammonium chloride solution and leaves large globules, and green dye was not tested.)

A front light source was used to illuminate the specimen. A CCD camera attached to a microscopic lens was used to take the measurements. The CCD camera fed into a VCR which captured the images on tape. A framegrabber card

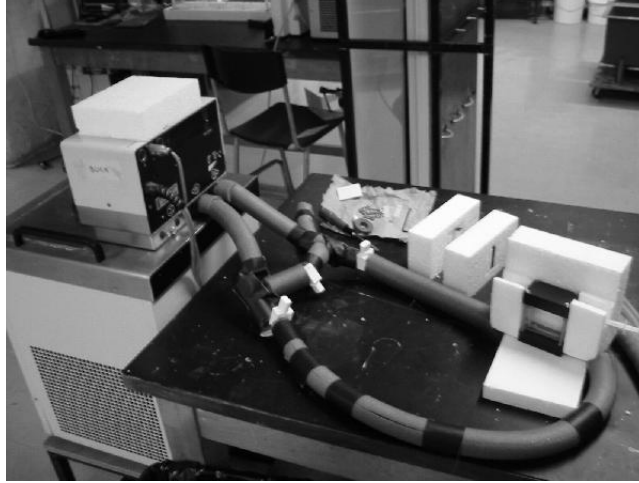


Figure 1. Experimental setup. Note the small solidification chamber, the tubing, and the refrigerator/pump unit.

attached to the computer could be used to grab digital frames from the analogue video with the software DigImage. The lens magnification ranges from 0.75x to 4.5x. A fine grid (1mm), printed on a transparency, can be used to calibrate the camera. By placing the grid in the chamber, and acquiring pictures at various magnifications, pixel coordinates can be converted to real-world coordinates by a simple transformation.

Because the chamber is cooled to -15°C , water condensation was originally a problem. The condensation on the glass chamber resulted in very poor images because it obstructed the view of the dendrites. To solve this problem, a high concentrated soap-water solution was applied in a thin layer to the surface of the chamber before the cooling was turned on. If condensation reappeared later in the experiment, the thin layer of soap-water was reapplied as necessary. This solved the main problems with condensation.

3 Data analysis

Once the pictures are acquired, the next task is image analysis. Figure 3 shows the contour of a magnified (via camera) region of Figure 2. This contour was acquired using DigImage to threshold those pixels within a range of intensity values. Notice the contour this gives has no guarantee of being smooth or continuous.

There was a choice of several data processing environments, including C, Fortran, Visual C++, Visual Basic, IDL, and Matlab. The ideal environment for high-level data analysis such as this would be in an environment tailored towards scientific analysis, such as IDL or Matlab. Matlab 6 Release 12.1 running on a Linux machine was selected as the best environment, because the simplicity

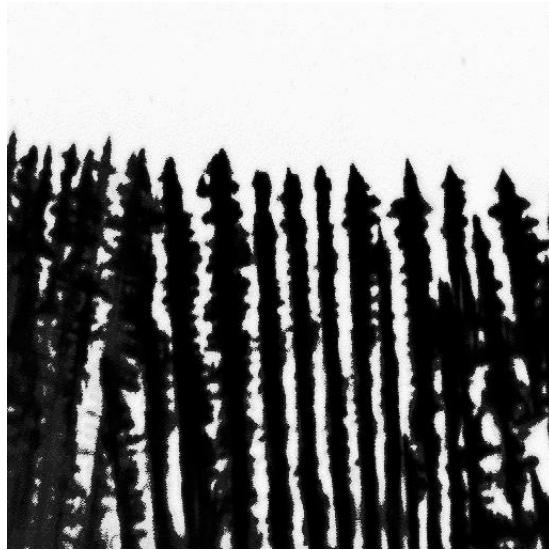


Figure 2. Picture of dendrites acquired with DigImage

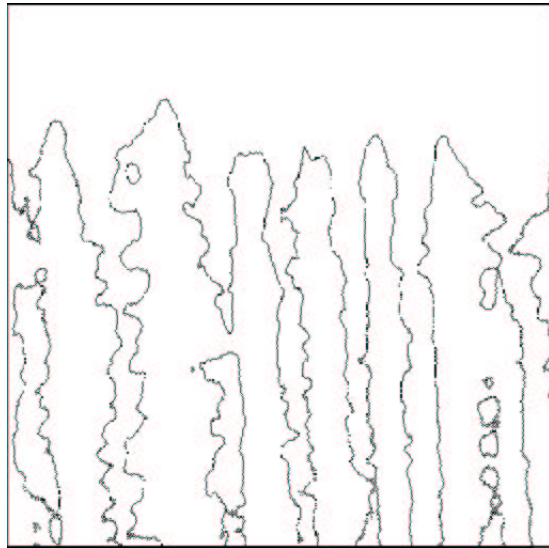


Figure 3. Outline of dendrites acquired with DigImage

and flexibility of Matlab leads to fast application development, the power of its built-in mathematical routines, and the wide acceptance, use, and support of it in the scientific community. Ideally, access to the Signal Processing Toolbox and the Image Processing Toolbox for Matlab would have made development a little easier.

The goal was to quantify the 'roughness' of the dendrites with depth. Fractal dimension and curvature were chosen as two measures of the roughness.

The first approach taken to measure the curvature was to fit the best circle to a small window of points along the contour. This approach resulted in extremely unreliable curvature estimates. The first problem with this approach is conceptual: what is needed is the osculating circle at a point to compute the curvature, not the best-fit circle. The second challenge is computational: the noise in the data points is sufficiently large to cause convergence to the true curvature using this method to be very poor. The challenge is to find the right window size, since a small window size will give a value corrupted too much by noise, while a large window size will result in an inaccurate estimate of the curvature because it will average out many details.

The first approach to address this problem was to approximate the discrete data in the original image by a continuous curve, and then to compute the curvature from this curve. The approach taken was with the Radial Basis Function Network. Using this approach a set of radial basis functions are fit to a decimation of the original image. Then, the functions may be evaluated at any arbitrary accuracy to give more data points in the contour. This approach was abandoned because the convergence of the RBF network was too slow for large images.

The second approach to address the problem of signal-to-noise in the contour is to find a better technique to extract the contour. Luckily, Matlab provides such a built-in function, **contour**. The contour function computes contour levels for an image represented as a matrix. The following code reads in an image and computes one contour:

```
img = imread( '11.TIF' );           % read in image
imagesc(img);                       % display image
C = contour( img, 1 );              % compute and display contour
axis ij;                             % reorient axis to match pixel coordinates
```

The contour is returned in the matrix C; for a description of the form of C, see the Matlab function reference. Figure 4 reports the result of loading an image and the corresponding contour.

The estimation of digital curvature is a difficult problem because the curvature is related to the second derivative of a function. Differentiation tends to amplify the presence of noise in a signal. Thus we wish to maximize the signal-to-noise ratio in the contour calculation. Technically speaking, a digital curve is nowhere

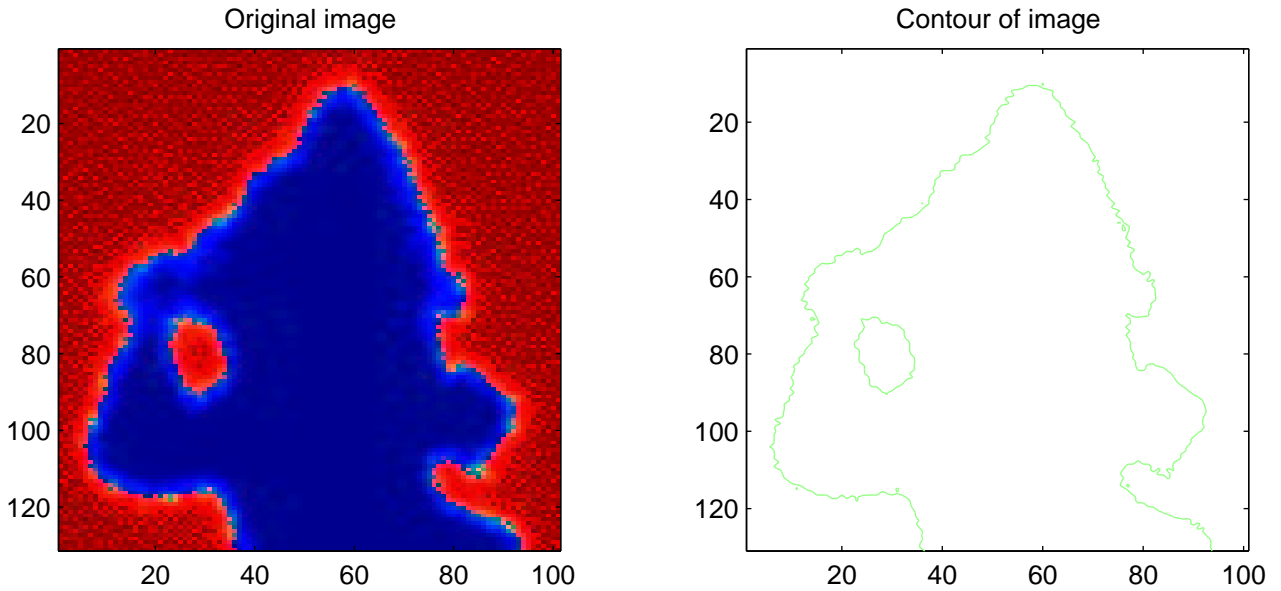


Figure 4. Using MATLAB contour command

continuous and nowhere differentiable, so the curvature technically does not exist anywhere in the strict analytical sense. What is really meant by digital curvature extraction is to estimate the curvature of the original pre-image object using the digital points present in the image.

Worring and Smeulders (1993)[3] report on several curvature estimation methods and report errors of between 1% to over 1000%. They present several techniques to reduce the error to between 1%–60%. This is an unacceptably large error for curvature estimation for this application.

Cesar and Costa (1996)[4] introduce a method of multiscale curvature estimation based on the Fourier transform, with reported error ranges from 0.1–1%. From experience with the dendrites, it is clear that curvature can vary vastly depending on the scale at which it is measured. This feature is true of fractals in general, since further magnification simply reveals more detail and hence leads to higher curvature. The problem becomes more complicated because the limited resolution of the camera leads to a smallest scale on the order of a pixel in size. The solution to this problem is to perform gaussian smoothing on the image before the curvature is computed. Because of the convolution theorem, applying gaussian smoothing in the spatial domain is equivalent to multiplying the FT by a gaussian in the spacial-frequency domain. This is the method implemented by Cesar and Costa, in what they call the “curvegram.” Furthermore, since the FFT algorithm allows very fast computation of the FT, the computation of the curvegram is extremely fast.

Figure 5 shows the results of applying the curvature algorithm with $\sigma = 8$

(reprinted in the Appendix) to a test image consisting of a circle attached to a rectangle. (Note: all σ values are given in the frequency domain; the standard deviation is proportional to $1/\sigma$ in the spatial domain.) Note the constant curvature along the contour of the semi-circle, the zero curvature along the edges of the rectangle, and the two high peaks corresponding to the edges in the rectangle. Notice that there is some noise in the curvature of the circular region and in the location of the edges. Using a larger σ (in the frequency domain), which corresponds to smoothing with a smaller σ in the spatial domain, leads to less smoothing, and thus reveals more details but also generates more noise. If a $\sigma = 16$ is applied to the image in Figure 5, then the location of the corners is sharper, but the noise in the circular region becomes greater.

Figure 6 shows the results of applying the curvature algorithm to a portion (the tip) of a real dendrite from Figure 2. Figure 6a shows $\sigma = 10$, while Figure 6b shows $\sigma = 40$. Note how a smaller value of σ in the frequency domain corresponds to more smoothing in the spatial domain. Notice also how smoothing enlarges the image slightly, leading to an *underestimate* of the curvature. Note also how smoothing removes the noise in the image, but also how it masks the finer details. Not enough smoothing reveals the details, but may be highly contaminated with noise.

Figure 7 shows the result of applying the algorithm to an entire dendrite. The rest of the dendrites are documented separately (see dendrites.ps).

Figure 8 shows the curvature along the contour of the dendrite in Figure 7 for four layers. The layers are ordered (left to right, top to bottom) as going from the top of the dendrite down. The rest of the dendrites are documented separately (see layers.ps). There appears to be no consistent differences in curvature with depth in the dendrites sampled.

4 Directions for future work

The technical capacity developed during this summer may be utilized by another future UROP student to begin a scientific study of dendritic mushy layers. It is hoped that a Cambridge University student would continue this project next year under the new UROP scheme being modelled after MIT. It is hoped that this document has conveyed the information that will be necessary for the next student to pick up where the present author left off, without hinderance. The next student is invited to write to the author if there are any questions, misunderstandings, or if there is any missing information.

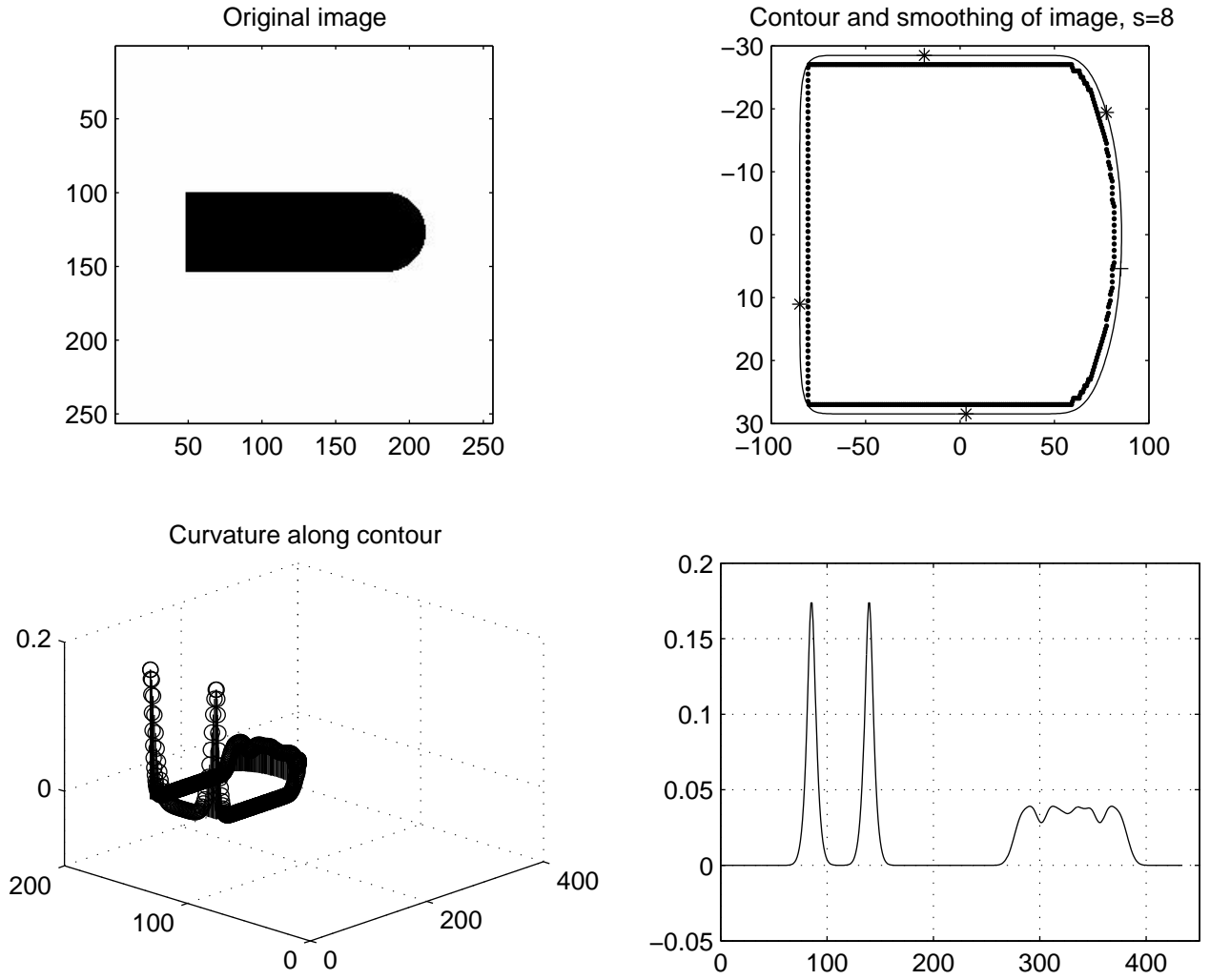


Figure 5. Curvature along the test image using method in Cesar and Costa (1997)[5]

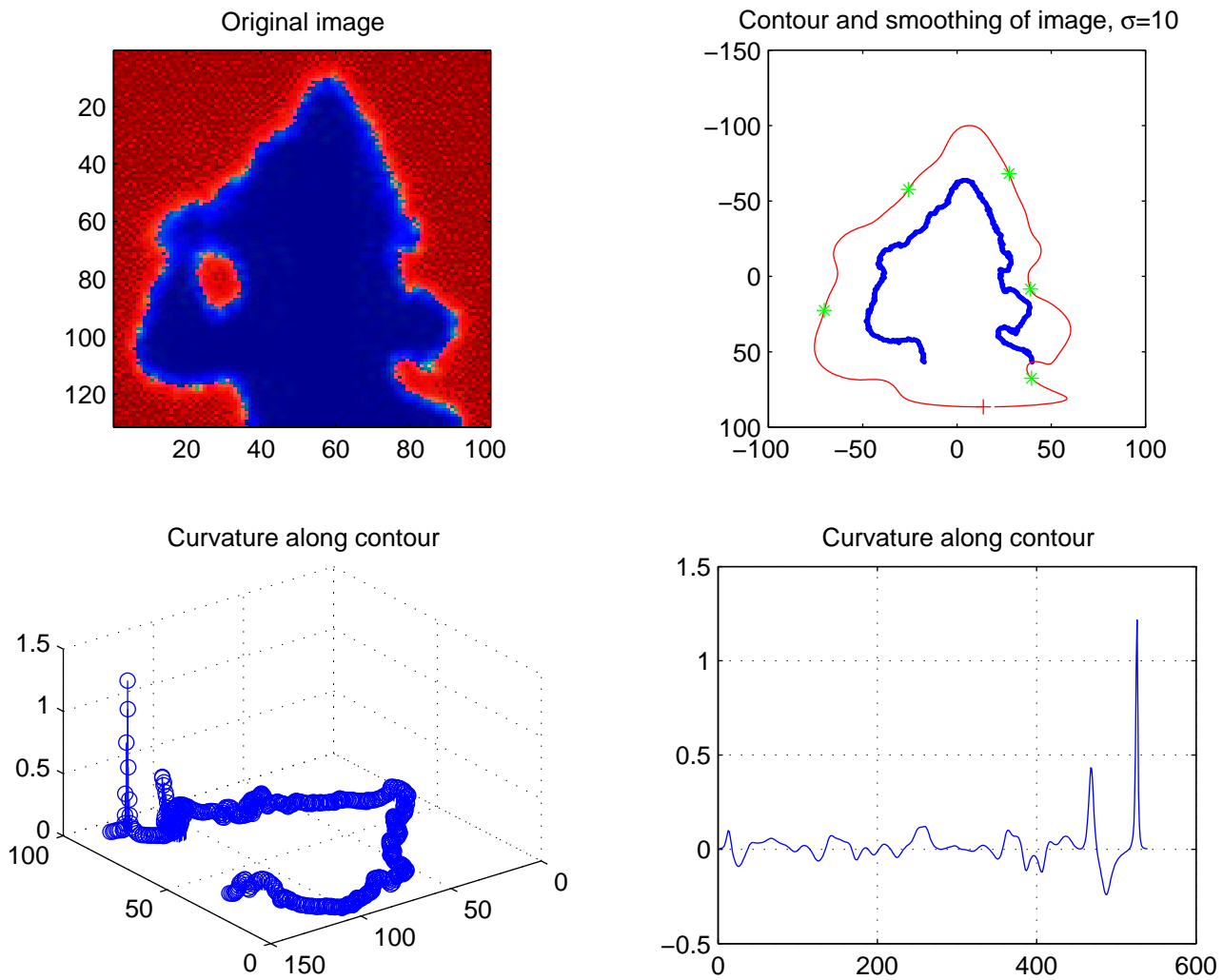


Figure 6a. Curvature along dendrite tip, $\sigma = 10$.

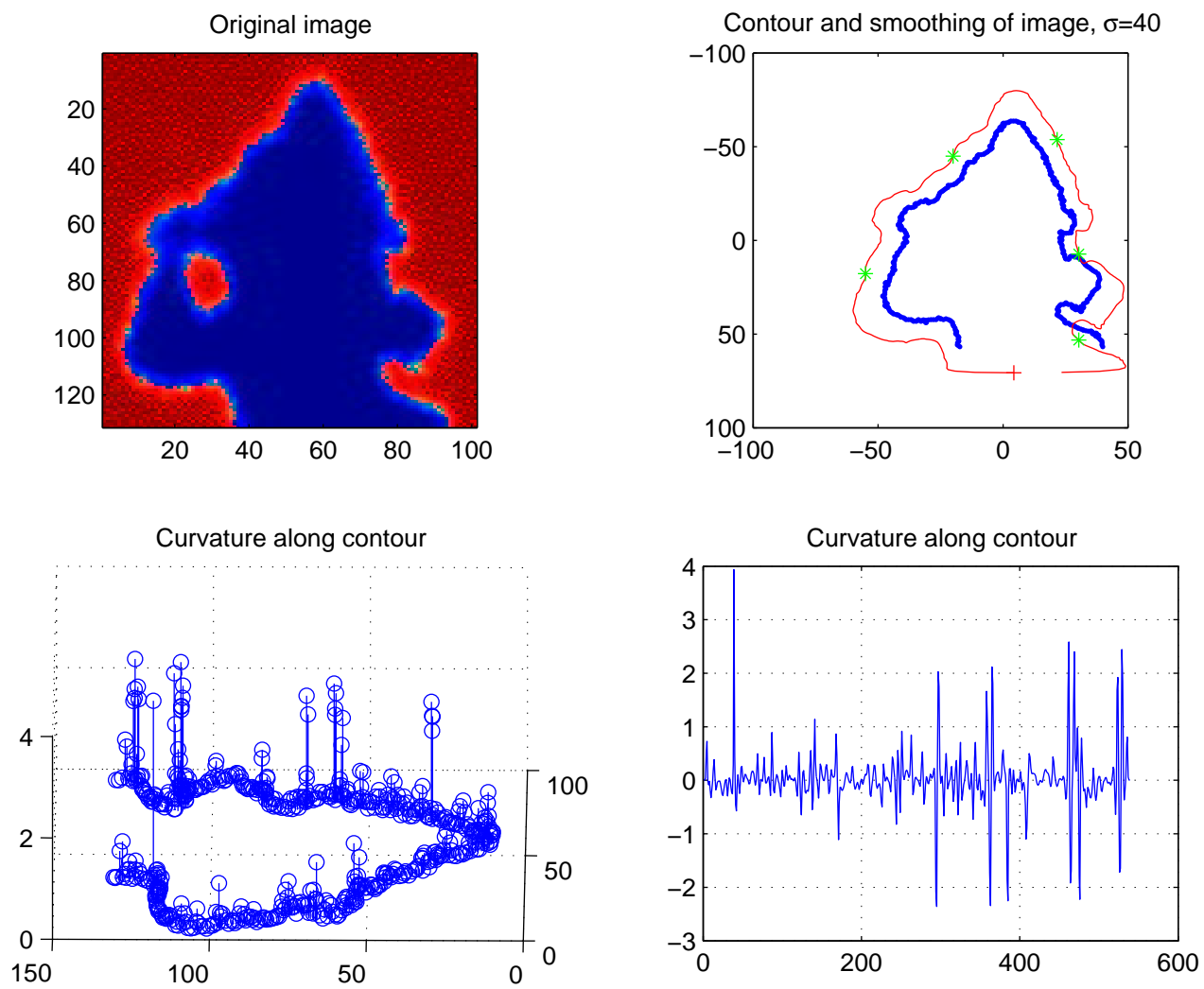


Figure 6b. Curvature along dendrite tip, $\sigma = 40$.

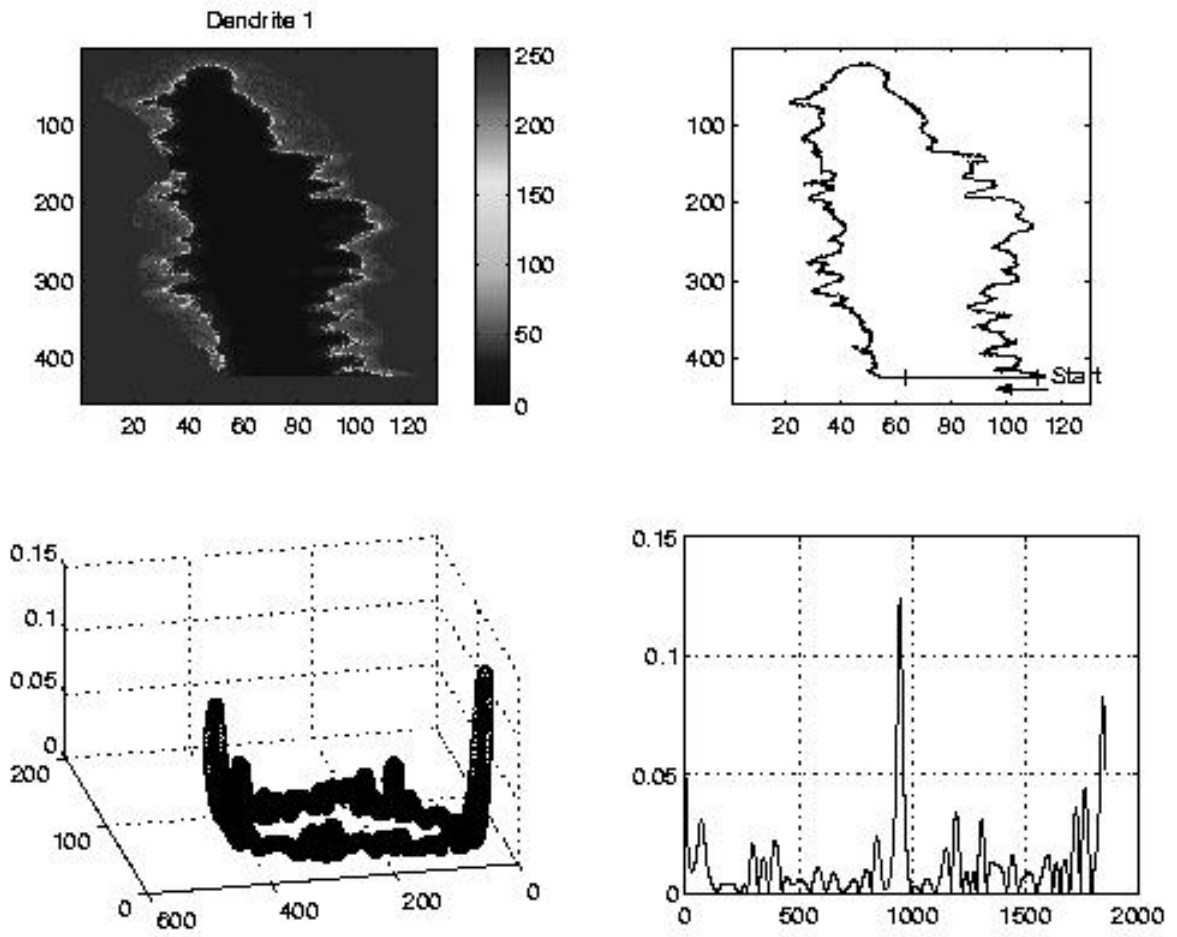


Figure 7. Curvature along dendrite 1.

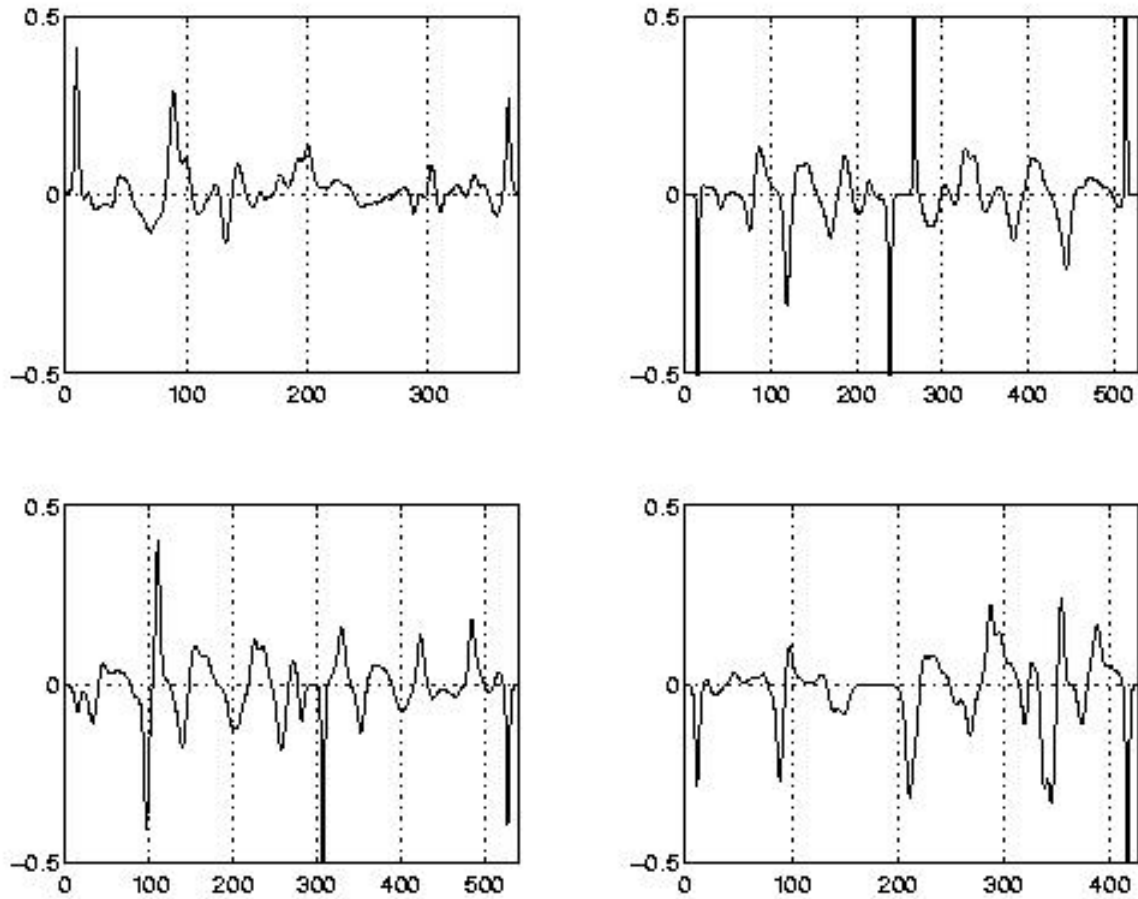


Figure 8. Curvature along dendrite in Figure 7, with the each figure representing going down in depth in the dendrite from left to right and top to down.

5 References and key words

In the context of doing this research, a number of key words useful during literature search have been noted. These are listed here for reference:

Radial Basis Function Networks - neural networks used to compute the best radial basis function approximation to a set of data

Digital curvature estimation - useful for finding references on computing the digital curvature from a contour

morphology, morphometrics - the study of shape in biological sciences, the mathematical study of shape. Biologists have done quite a bit of work on shape analysis, shape detection, contour description, etc. in fields such as neuroscience and paleobiology. Although this work has been guided by their unique applications, many of the software solutions that they have found can be quite general and useful.

References

- [1] Mokhtarian F. and Mackworth A. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on pattern analysis and machine intelligence*, 14(8):789–805, August 1992.
- [2] Costa L.D.F. and Cesar R.M. *Shape Analysis and Classification*. CTC Press, 2000.
- [3] Worring M. and Smeulders A.W.M. Digital curvature estimation. *CVGIP: Image understanding*, 58(3):366–82, 1993.
- [4] Cesar R.M. and Costa L.D.F. Towards effective planar shape representation. *Pattern recognition*, 29(9):1559–69, 1996.
- [5] Cesar R.M. and Costa L.D.F. Application and assessment of multiscale bending energy for morphometric characterization of neural cells. *Rev. Sci. Instrum.*, 68(5):2177–86, May 1997.
- [6] Estrozi L. Campos A.G. Rios L.G. Cesar R.M. and Costa L.D.F. Comparing curvature estimation techniques. *Cybernetic vision research group*, 1999.

6 Appendix: Source code

```
/DAMTP/a/cortex/local/raid/itg/v0207a/src/curvegram2.m Page 1  
August 15, 2002 2:18:10 PM

---

  
function k = curvegram2( x, y, sigma )  
% Curvegram based on Costa & Cesar (1997)  
% Modified from Multiscale Curvature Analysis MATLAB Toolbox  
% by Roberto M. Cesar (2001)  
% Modification by Daniar Hussain on 2002/08/12  
%  
% Returns the curvature k along the contour specified in x,y  
% with a gaussian filter with standard deviation sigma  
  
Np = length(x); % Length variables  
Np1 = Np - 1;  
Np2 = floor(Np/2);  
  
u = x + j*y; % Complex representation  
U = fft(u); % Fourier transform  
U(1) = 0; % Remove DC component, no effect on curvature  
u=ifft(U); % Shifted original  
U = fftshift(U); % Shift frequencies  
E0 = sum( abs(U) ); % Total energy in signal used for energy renormalization  
  
s = -Np2:(Np1-Np2); % Used for derivature calculation  
  
% Create gaussian filter  
GF = exp( - (s.^2) * ((2*sigma).^(-2)) );  
  
U = U .* GF; % Apply filter  
dU = j * s .* U; % Compute derivature  
ddU = (j*s).^2 .* U; % Compute second derivative  
  
ru = ifft( ifftshift(U) ); % Get back time-domain signals  
du = ifft( ifftshift(dU) );  
ddu = ifft( ifftshift(ddU) );  
  
E1 = sum( abs(U) ); % New energy in signal after filtration  
ru = ru .* E0/E1; % Apply energy renormalization  
du = du .* E0/E1;  
ddu = ddu .* E0/E1;  
  
% Compute complex curvature  
k = (-imag( du .* conj(ddu) )) ./ (abs(du).^3);  
  
k = flipud(k); % Flip up-down so that scale increases downwards  
  
% Plotting  
plot(real(u),imag(u),'b. '); % Original data points  
axis square; axis ij; hold on; % Configure plot  
plot(real(ru),imag(ru),'r '); % Approximated contour  
plot(real(ru(1)),imag(ru(1)),'r+'); % Starting point in contour  
for i=101:100:size(ru,2) % Useful for comparisons between plots
```

Acknowledgements

The author is grateful to the Cambridge-MIT Institute for funding this project and for giving the author the opportunity to visit and study in England. The author is indebted to Herbert Huppert and Grae Worster for being so welcoming to the author in their research lab. The author is also thankful to Mark Hallworth, Dave Page-Craft, and other technicians for technical assistance. And to the rest of the staff, students, and faculty of the Fluid Dynamics Lab for a friendly and welcoming atmosphere during the summer.