MATLAB code to calculate the curvature of interface contours

Curvature of a digital curve

Jaya Kumar Alageshan *

Indian Institute of Science, Bangalore, India.
http://jkumarres.github.io

Abstract

This document describes the algorithm used to find the curvature of interface contours in an image, using the accompanying MATLAB code. We use a gradient descent based iterative scheme to estimate the best fit circle for local data points and find the curvature vector. The approach described here leads to a smoother varying curvature field in comparison to some of the methods available in literature.

MathWorks: https://in.mathworks.com/matlabcentral/fileexchange/93175-interface-curvature
Git: https://github.com/jkumarres/InterfaceCurvature

Keywords: Curvature. Interface. Digital curve.

1 Algorithm

A schematic of the different steps involved are given in Fig. 1.

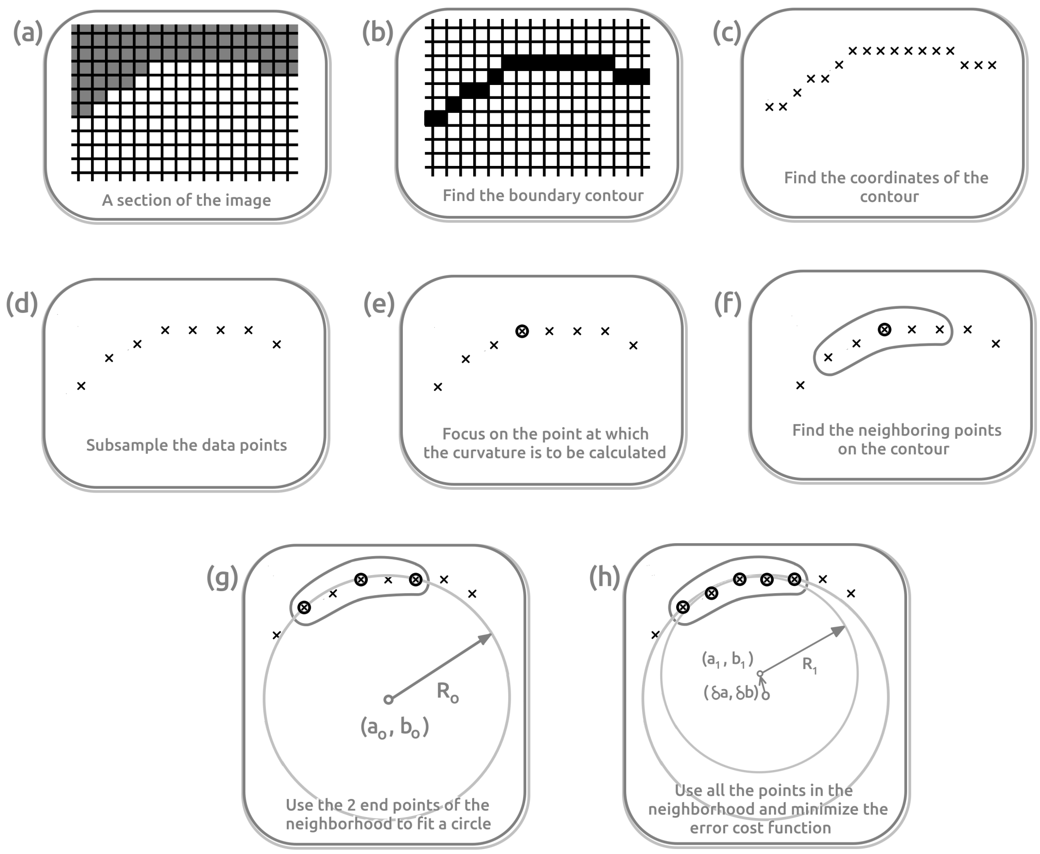


Figure 1. A schematic of sequence of operations performed to find the curvature at a point on the interface.

*Email: jkumar.res@gmail.com

2 Input-Output

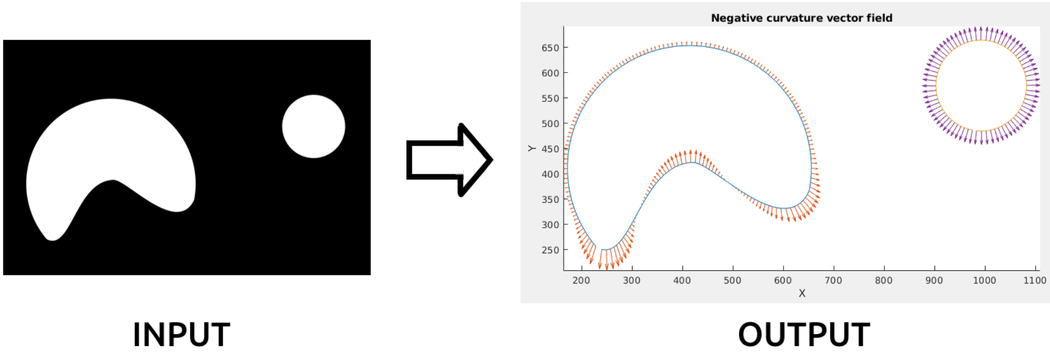


Figure 2. Typical input and output.

3 Function variables

The I/O format of the function is:

function [xy , K] = InterfaceCurvature(I , v , NN , SR)

where the inputs are

$$
\begin{aligned}
I &\longrightarrow \text{Image data in the form of 2D, 3D or 4D array} \\
v &\longrightarrow \text{Value of the level-set} \in (0,1) \\
SR &\longrightarrow \text{Sub-sample rate to go from Fig. 1(c) to (d) (here } SR = 2) \\
NN &\longrightarrow \text{To specify the neighborhood in Fig.1(f) (here } NN = 2)
\end{aligned}
$$

and the outputs are

$$
\begin{aligned}
xy &\longrightarrow \text{X- and Y- coordinates of the sub-sampled contour points in} \\
& \text{Fig. 1(d) of } N_c \times 1 \text{ cells} \\
K &\longrightarrow \text{The curvature vector of size } N_c \times 1 \text{ cells}
\end{aligned}
$$

where, N_c is the number of distinct contours.

4 Circle through three points (Fig. 1(g))

Now to perform the operation given in Fig.1(g), we need to fit a circle through three points. The equation of a circle is given by

$$(x-a)^2 + (y-b)^2 = r^2 \tag{1}$$

where, (a, b) is the coordinate of the circle center, and r is the radius.

Now, if (x_1, y_1), (x_2, y_2), and (x_3, y_3) are three given points, then the corresponding circle through them has

$$
\begin{aligned}
r &= \frac{1}{2}\frac{\sqrt{\left((x_1-x_2)^2+(y_1-y_2)^2\right)\left((x_2-x_3)^2+(y_2-y_3)^2\right)\left((x_3-x_1)^2+(y_3-y_1)^2\right)}}{|x_1(y_2-y_3)+x_2(y_3-y_1)+x_3(y_1-y_2)|} && (2) \\
a &= \frac{1}{2}\frac{x_1^2(y_2-y_3)+x_2^2(y_3-y_1)+x_3^2(y_1-y_2)-(y_1-y_2)(y_2-y_3)(y_3-y_1)}{x_1(y_2-y_3)+x_2(y_3-y_1)+x_3(y_1-y_2)} && (3) \\
b &= \frac{1}{2}\frac{y_1^2(x_2-x_3)+y_2^2(x_3-x_1)+y_3^2(x_1-x_2)-(x_1-x_2)(x_2-x_3)(x_3-x_1)}{y_1(x_2-x_3)+y_2(x_3-x_1)+y_3(x_1-x_2)} && (4)
\end{aligned}
$$

5 Circle fit (Fig. 1(h))

We use the gradient descent method to find the best value of $\{a, b, r\}$ that fits a given set of points. A convenient positive-definite cost function, on which to implement gradient descent is

$$\mathcal{E}(a, b, r) := \sum_{i \in \mathcal{N}} \left[(x_i - a)^2 + (y_i - b)^2 - r^2\right]^2 \tag{5}$$

where $\mathcal{N}$ are the neighboring points.

The gradient descent is an iterative algorithm that updates the variables, as follows

$$r' \longrightarrow r - \eta \frac{\partial \mathcal{E}}{\partial r} = r + 4r\,\eta \sum_{i \in \mathcal{N}} \left[(x_i - a)^2 + (y_i - b)^2 - r^2\right] \tag{6}$$

$$a' \longrightarrow a - \eta \frac{\partial \mathcal{E}}{\partial a} = a + 4\eta \sum_{i \in \mathcal{N}} (x_i - a) \left[(x_i - a)^2 + (y_i - b)^2 - r^2\right] \tag{7}$$

$$b' \longrightarrow b - \eta \frac{\partial \mathcal{E}}{\partial b} = r + 4\eta \sum_{i \in \mathcal{N}} (y_i - b) \left[(x_i - a)^2 + (y_i - b)^2 - r^2\right] \tag{8}$$

where, η is the descent rate, which we fix at 0.01. Starting from the values of $\{a, b, r\}$ estimated using the three extreme points in Sec. 4 (Fig. 1(g)), the above iteration is performed 100 times to arrive at the final solution.

6 Curvature vector

Once we have calculated (a, b, r) for a point (x_i, y_i), then the curvature vector is defined as

$$\mathbf{K}(x_i, y_i) := \frac{1}{r} \frac{(a - x_i, b - y_i)}{\sqrt{(a - x_i)^2 + (b - y_i)^2}} \tag{9}$$

so that the magnitude of $\mathbf{K}$ is the curvature, and the vector points towards the center of the circle.

7 Visualization

To visualize the curvature field along the contours, we plot the vector field of $-\mathbf{K}(x_i, y_i)$. Note that we plot the negative of the curvature vector, as this ensures that there is no crowding of vectors or formation of cusps.

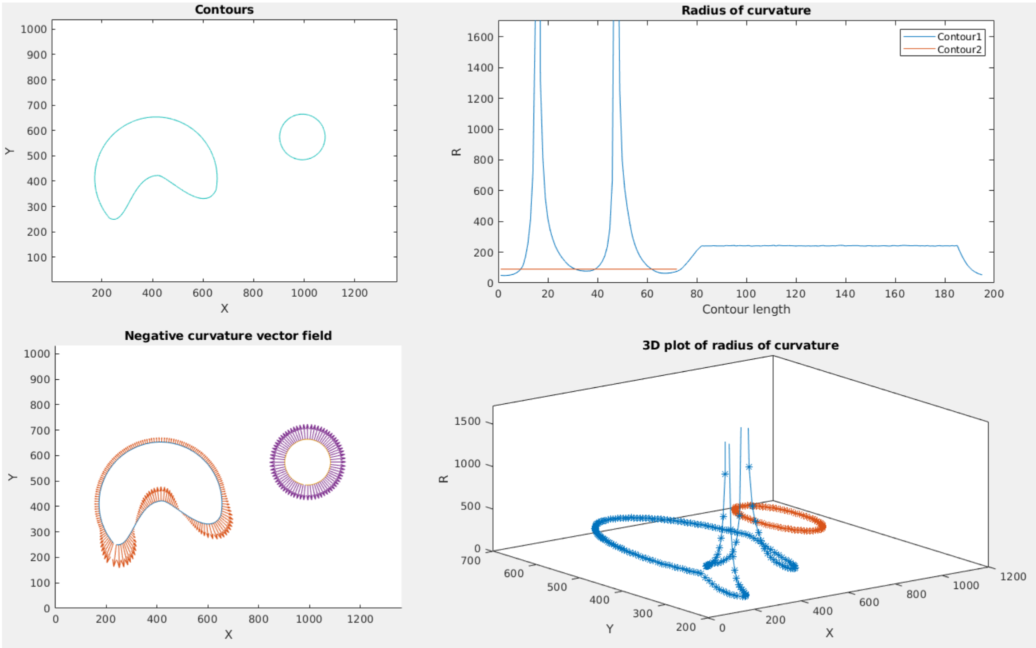


Figure 3. The top-left plot shows the extracted contours; bottom-left plot shows the corresponding curvature vector field; top-right plot shows the variation in radius of curvature along the contour length; bottom-right figure shows the spatial dependence of the radius of curvature.

A Appendix: Radius of curvature

Given a smooth curve in 2D, that is twice differentiable, the curvature is defined as

$$\mathbf{K} := \frac{d\hat{\mathbf{t}}}{ds}, \tag{10}$$

$$\text{where, } \hat{\mathbf{t}} = \frac{d\mathbf{R}(s)}{ds} \tag{11}$$

$\hat{\mathbf{t}}$ is the unit tangent vector, and $\mathbf{R}(s)$ is the position vector of the curve, parameterized by the arc-length parameter s, such that $ds = \sqrt{dx^2 + dy^2}$.

But from a computation point of view, a better representation of curvature is in terms of the inverse radius of the best fit (osculating) circle, as shown in Fig. 4

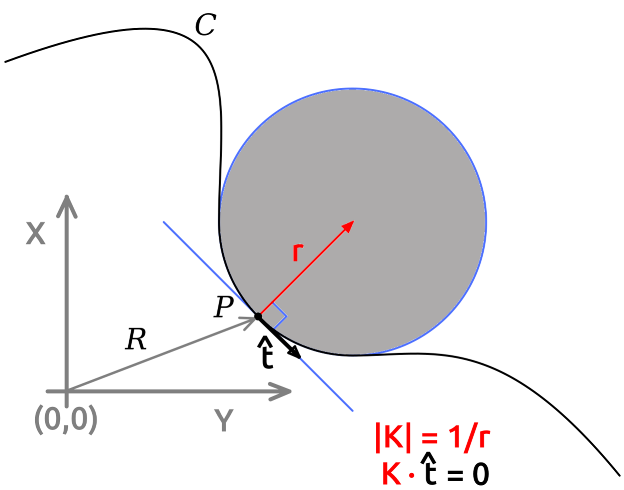


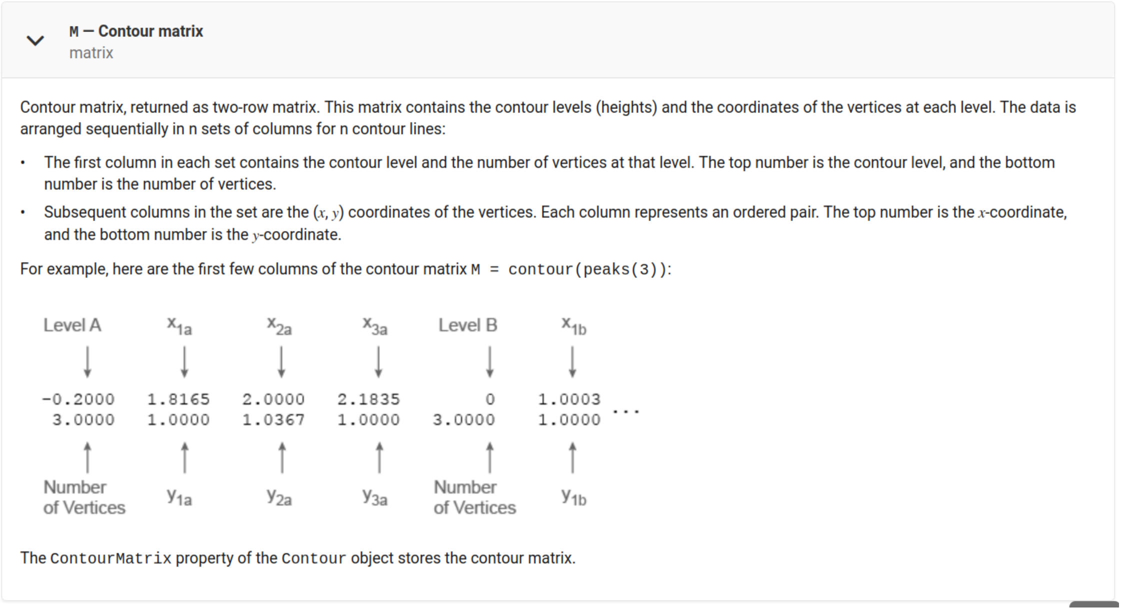
Figure 4. Schematic of curvature calculation

B Appendix: Note on MATLAB *contour*() function format

In MATLAB, to get the coordinates of the contour, corresponding to the level-set value of v, we use

M = contour(I , [v v]);

where the format of M is as described in Fig. 5 (from mathworks documentation)



M — Contour matrix
matrix

Contour matrix, returned as two-row matrix. This matrix contains the contour levels (heights) and the coordinates of the vertices at each level. The data is arranged sequentially in n sets of columns for n contour lines:

- The first column in each set contains the contour level and the number of vertices at that level. The top number is the contour level, and the bottom number is the number of vertices.
- Subsequent columns in the set are the (x, y) coordinates of the vertices. Each column represents an ordered pair. The top number is the x-coordinate, and the bottom number is the y-coordinate.

For example, here are the first few columns of the contour matrix `M = contour(peaks(3))`:

The `ContourMatrix` property of the `Contour` object stores the contour matrix.

Figure 5. Caption

C MATLAB code

C.1 main.m

```
clear
close all

I = imread('Test1.png');
%I = rgb2gray(I);

[xy,K] = InterfaceCurvature( I , 0.5 , 10 , 10 );

figure;
hold on;
[Nc,¬] = size(xy);  % Nc: number of closed contours

for i=1:Nc
    plot(xy{i}(:,1),xy{i}(:,2));
    quiver(xy{i}(:,1),xy{i}(:,2),K{i}(:,1),K{i}(:,2));
end

axis equal;
axis tight;

title('Negative curvature vector field');
xlabel('X');
ylabel('Y');
```

C.2 InterfaceCurvature.m

```
function [ xy , K ] = InterfaceCurvature( I , v , NN , SR )

    Flag = 1;  % Flag = 1 --> Display plots
               % Flag = 0 --> Do not display plots

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Description of I/O variables:                                        %
    %                                                                      %
    % INPUT:-                                                              %
    %                                                                      %
    % I    : image matrix, can be 2D or 3D (RGB)                           %
    % v    : contour / level-set value, which should be in the range [0,1] %
    % NN   : number of neighboring points used to fit a circle            %
    % SR   : sub-sampling rate (should be integer > 0)                     %
    %                                                                      %
    % OUTPUT:-                                                             %
    %                                                                      %
    % K    : curvature vector (cell of dim equal to number of contours)    %
    % xy   : coordinates of the points on the contour                      %
    %                                                                      %
    %                                                                      %
    % NOTE: For reliable outputs, the image must be atleast 300x300        %
    %       and each contour must have atleast 500 points.                 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Author: Jaya Kumar. A
    % E-mail: jkumar.res@gmail.com
    %
    % See the accompanying "doc.pdf" for algorithm details.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%% Formating the data %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

% Converting a 3D color image to 2D grayscale image

if length(size(I))==3
    I = rgb2gray(I);
end

I = double(I);
Imin = min(I(:));
Imax = max(I(:));

I = (I-Imin) / (Imax-Imin);  % Normalizing the data [0,1]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Padding the data to take care of   %%%
%%% interfaces that hit the boundaries %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Lx,Ly] = size(I);

I2 = zeros(Lx+6,Ly+6);
I2( 4:Lx+3 , 4:Ly+3 ) = I;

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Contour extraction %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

N = NN;
figure;
subplot(2,2,1)
xy = contour( I2 , [v v] );
title('Contours');
xlabel('X');
ylabel('Y');
axis equal;
axis tight;
%set(gca,'YDir','reverse');

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Conttour splitting %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

Np = length( xy(1,:) );
Lc = xy(2,1);   % Number of points on the 1st contour
ll = Lc+1;
iLc = 1;

while( ll < Np )
    Lc = [Lc ; xy(2,ll+1)];
    iLc = [iLc ; ll+1];
    ll = sum( Lc+1 );
end

Nc = length( Lc );    % Number of contours


% Cells to handle multiple contours with different sets of points

x = cell(Nc,1);     % X-coordinates of the contour
y = cell(Nc,1);     % Y-coordinates of the contour
r = cell(Nc,1);     % Radius of curvature
rr = cell(Nc,1);    %
x2 = cell(Nc,1);    % Sampled X-coordinates
y2 = cell(Nc,1);    % Sampled Y-coordinates
```

```
    a = cell(Nc,1);     % X-coordinate of the circle center
    b = cell(Nc,1);     % Y-coordinate of the circle center

    for i=1:Nc
        x{i} = xy( 1 , iLc(i)+1 : iLc(i)+Lc(i) )';
        y{i} = xy( 2 , iLc(i)+1 : iLc(i)+Lc(i) )';
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% Calculation of curvature %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for i=1:Nc
        %%%%%%%%%%%%%%%%%%%%
        %%% Sub-sampling %%%
        %%%%%%%%%%%%%%%%%%%%

        Np = length(x{i});

        ll = floor( Np/SR );
        xx = reshape( x{i}(1:SR*ll) , [SR ll] );
        yy = reshape( y{i}(1:SR*ll) , [SR ll] );

        xx = xx(1,:)';
        yy = yy(1,:)';
        Np = ll;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%% Locally fitting circles %%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        r{i} = zeros(Np,1);
        a{i} = zeros(Np,1);
        b{i} = zeros(Np,1);

        for j=1:Np
            %%% Maintain contour periodicity
            j1 = mod( j-NN-1 , Np ) + 1;
            j2 = mod( j+NN-1 , Np ) + 1;

            if( j1 < j2 )
                [a{i}(j),b{i}(j),r{i}(j)] = FitCircle(xx( j1:j2 ),yy( j1:j2 ));
            else
                [a{i}(j),b{i}(j),r{i}(j)] = FitCircle( [xx( j1:end ) ; xx(1:j2)]...
                                                      ,[yy( j1:end ) ; yy(1:j2)]);
            end
        end

        rr{i} = abs(r{i});

        x2{i} = xx;
        y2{i} = yy;
        a{i} = real(a{i});
        b{i} = real(b{i});
    end

    if( Flag==1 )
        subplot(2,2,2);
        plot(rr{1});
        title('Radius of curvature');
        hold on;
        leg = cell(Nc,1);
        leg{1} = 'Contour1';

        for i=2:Nc
            plot(abs(rr{i}));
```

```
            leg{i} = sprintf('Contour%d',i);
        end

        ylim( [0 , sqrt(Lx^2+Ly^2)] );
        xlabel('Contour length');
        ylabel('R');
        legend( leg );

        subplot(2,2,4);
        plot3(x2{1},y2{1},rr{1},'-*');
        title('3D plot of radius of curvature');
        hold on;

        for i=2:Nc
            plot3(x2{i},y2{i},rr{i},'-*');
        end

        box on;
        zlim( [0 , sqrt(Lx^2+Ly^2)] );
        xlabel('X');
        ylabel('Y');
        zlabel('R');
    end

    v = cell(Nc,1);
    K = cell(Nc,1);

    for i=1:Nc
        v{i} = [(x2{i}-a{i}) (y2{i}-b{i})];
        vv = sqrt( v{i}(:,1).^2 + v{i}(:,2).^2 );
        v{i} = v{i} ./ [vv vv];

        K{i} = v{i} ./ [rr{i} rr{i}];
    end

    if( Flag==1 )
        subplot(2,2,3);
        [m,n] = size(I2);
        I3 = zeros(m,n,3);
        I3(:,:,1) = I2;
        I3(:,:,2) = I2;
        I3(:,:,3) = I2;
        %imagesc(I3);
        hold on;

        for i=1:Nc
            plot(x2{i},y2{i});
            h = quiver(x2{i},y2{i},K{i}(:,1),K{i}(:,2));
            set(h,'AutoScale','on', 'AutoScaleFactor', 2)
        end

        axis equal;
        axis tight;
        xlim([0 Ly]);
        ylim([0 Lx]);
        title('Negative curvature vector field');
        xlabel('X');
        ylabel('Y');
    end

    xy = cell(Nc,1);

    for i=1:Nc
        xy{i} = [x2{i} y2{i}];
    end

```

```
end
```

C.3 FitCircle.m

```
function [a,b,r] = FitCircle(x,y)

N = length(x);
NN = 10;
eta = 0.001;

n = floor(N/2);

x0 = sum(x)/N;
y0 = sum(y)/N;

x = x - x0;
y = y - y0;

x1 = x(1);
y1 = y(1);
x2 = x(n);
y2 = y(n);
x3 = x(N);
y3 = y(N);

x12 = x1-x2;
x23 = x2-x3;
x31 = x3-x1;

y12 = y1-y2;
y23 = y2-y3;
y31 = y3-y1;

Num = (x12^2 + y12^2) * (x23^2 + y23^2) * (x31^2 + y31^2);
Den = (x1 * y23 + x2 * y31 + x3 * y12);

r = abs( 0.5 * sqrt(Num) / Den );

Num = x1^2 * y23 + x2^2 * y31 + x3^2 * y12 - y12 * y23 * y31;
a = 0.5 * Num / Den;

Num = y1^2 * x23 + y2^2 * x31 + y3^2 * x12 - x12 * x23 * x31;
b = 0.5 * Num / (-Den);

s = r;

r = r/s;
x = x/s;
y = y/s;
a = a/s;
b = b/s;

for i=1:NN
    E = 0;
    dEda = 0;
    dEdb = 0;
    dEdr = 0;

    for j=1:N
        dE = (x(j)-a)^2 + (y(j)-b)^2 - r^2 ;
        E = E + dE^2;

        dEda = dEda - 4 * dE * (x(j)-a);
        dEdb = dEdb - 4 * dE * (y(j)-b);
```

```
        dEdr = dEdr - 4 * dE * r;
    end

    r = r - eta * dEdr;
    a = a - eta * dEda;
    b = b - eta * dEdb;
end

a = s * a;
b = s * b;
r = s * r;

a = a + x0;
b = b + y0;

return;
```